8th Conference on Advances in Financial Technologies

AFT 2026, October 6–9, 2026, London, UK

Edited by

Aggelos Kiayias

Maria Kyropoulou



Editors

Aggelos Kiayias
University of Edinburgh and IOG, UK
aggelos.kiayias@ed.ac.uk

Maria Kyropoulou
University of Essex, UK
maria.kyropoulou@essex.ac.uk

ACM Classification 2012
Security and privacy → Distributed systems security; Theory of computation → Algorithmic game theory and mechanism design; Applied computing → Economics; Security and privacy → Economics of security and privacy; Applied computing → Electronic commerce; Security and privacy → Cryptography; Security and privacy → Security protocols; Applied computing → Digital cash; Theory of computation → Distributed algorithms

ISBN 978-3-95977-451-2

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at https://www.dagstuhl.de/dagpub/978-3-95977-451-2.

Publication date
October, 2026

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists all publications of this volume in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at https://portal.dnb.de.

License
This work is licensed under a Creative Commons Attribution 4.0 International license (CC-BY 4.0): https://creativecommons.org/licenses/by/4.0/legalcode.
In brief, this license authorizes each and everybody to share (to copy, distribute and transmit) the work under the following conditions, without impairing or restricting the authors' moral rights:
- Attribution: The work must be attributed to its authors.

The copyright is retained by the corresponding authors.

Digital Object Identifier: 10.4230/LIPIcs.AFT.2026.0

ISBN 978-3-95977-451-2 **ISSN 1868-8969** **https://www.dagstuhl.de/lipics**

LIPIcs – Leibniz International Proceedings in Informatics

LIPIcs is a series of high-quality conference proceedings across all fields in informatics. LIPIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

Editorial Board

- Patricia Bouyer-Decitre (Université Paris-Saclay, CNRS, ENS Paris-Saclay, FR)
- Roberto Di Cosmo (Inria and Université Paris Cité, FR)
- Faith Ellen (University of Toronto, CA)
- Holger Hermanns (Universität des Saarlandes, Saarbrücken, DE and Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Wadern, DE)
- Laura Kovács (TU Wien, AT)
- Daniel Král' (Leipzig University, DE and Max Planck Institute for Mathematics in the Sciences, Leipzig, DE – Chair)
- Sławomir Lasota (University of Warsaw, PL)
- Meena Mahajan (Institute of Mathematical Sciences, Chennai, IN)
- Chih-Hao Luke Ong (Nanyang Technological University, SG)
- Eva Rotenberg (IT University of Copenhagen, DK)
- Pierre Senellart (ENS, Université PSL, Paris, France)
- Alexandra Silva (Cornell University, Ithaca, US)

ISSN 1868-8969

https://www.dagstuhl.de/lipics

Contents

Preface
Aggelos Kiayias and Maria Kyropoulou 0:ix

Program Committee
..... 0:xi

Steering Committee
..... 0:xiii

External Reviewers
..... 0:xv

List of Authors
..... 0:xvii

Session 1: Cryptographic Protocols

BitVM: Quasi-Turing Complete Computation on Bitcoin
Lukas Aumayr, Zeta Avarikioti, Robin Linus Woll, Matteo Maffei, Andrea Pelosi, Christos Stefo, and Alexei Zamyatin 1:1–1:23

Privacy-Preserving Inclusion Lists
Zhengwei Tong, Saba Eskandarian, and Kartik Nayak 2:1–2:23

SoK: Cryptographic Key Recovery for Cryptoasset Custody and Financial Technologies
Francisco Javier Becerra Sanchez, Antonio Ken Iannillo, and Radu State 3:1–3:28

Session 2: Consensus Protocols

Gatling: Rapid-Fire Consensus from Parallel Composition
Giulia Scaffino, Max Resnick, and Joachim Neu 4:1–4:24

The Consensus Number of Untraceable Cryptocurrencies
Christian Cachin, David Lehnherr, Juan Villacis, and François-Xavier Wicht 5:1–5:24

Session 3: Proof-of-Stake Protocols

Reserve Depletion and Security Runway in Proof-of-Stake Systems
Paolo Penna and Manvir Schneider 6:1–6:22

The Cost of Secure Restaking vs. Proof-Of-Stake
Akaki Mamageishvili and Benny Sudakov 7:1–7:16

StakeNote: A Proof-Of-Stake Protocol for CryptoNote Payments
Bernardo David and Dimitris Karakostas 8:1–8:24

8th Conference on Advances in Financial Technologies (AFT 2026).
Editors: Aggelos Kiayias and Maria Kyropoulou
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Session 4: Parallel Execution and State Contention

Approximation and Hardness Results for the Parallel-Block Construction Problem
Arivarasan Karmegam, Alexandru Popa, Lucianna Kiffer, and Antonio Fernández Anta 9:1–9:22

Not All Reads Are Conflicts: A Write-Only Analysis of the Sui Blockchain
Haygen Tsoi, Alberto Sonnino, and Philipp Jovanovic 10:1–10:20

SAMM: Sharded Automated Market Maker
Hongyin Chen, Amit Vaisman, and Ittay Eyal 11:1–11:27

Session 5: Automated Market Makers and Liquidity

Defensive Rebalancing in Networks of Automated Market Makers
Sam Devorsetz and Maurice Herlihy 12:1–12:17

Multi-Currency AMMs for Decentralized FOREX Markets: Feasibility & Optimal Design
Reina Ke Xin Li, Andreas Park, Andreas Veneris, and Srisht Fateh Singh 13:1–13:23

Uniform-Loss Automated Market Making for Prediction Markets
Ciamac C. Moallemi, Dan Robinson, and Brian Zhu 14:1–14:3

ParlayMarket: Automated Market Making for Parlay-Style Joint Contracts
Ranvir Rana, Viraj Nadkarni, Niusha Moshrefi, and Pramod Viswanath 15:1–15:22

Session 6: Mechanism Design

On Sybil-Proof Mechanisms
Minghao Pan, Bruno Mazorra, Christoph Schlegel, and Akaki Mamageishvili 16:1–16:3

Belief-Aware Pivotal Mechanism for DAO Committees
Nuno Braz, Diogo Poças, and Miguel Correia 17:1–17:22

Credible Auctions via MPC Gadgets: Bounding Information Leakage Under Abort
Matheus V. X. Ferreira 18:1–18:22

Session 7: Auctions for Transaction Ordering

Competing Auctions in Intermediated Markets
Bruno Mazorra, Minghao Pan, and Christoph Schlegel 19:1–19:21

Just-In-Time Resale in an Ahead-Of-Time Auction for Faster Execution
Burak Öz, Akaki Mamageishvili, Christoph Schlegel, and Ali Taslimi 20:1–20:23

Towards Decentralized Searcher Competition in MEV Markets
Roozbeh Sarenche and Yunwen Liu 21:1–21:24

Session 8: MEV Incentives and Countermeasures

SoK: Encrypted Mempools Through the MEV Lens
Jonathan Passerat-Palmbach 22:1–22:25

RANDAO Manipulation in the Presence of MEV
Kaya Alpturer, Nicholas Hope, and S. Matthew Weinberg 23:1–23:22

To Wait or to Probe: Arbitrage Competition on High-Throughput Blockchains
Fei Wu and Burak Öz 24:1–24:24

Session 9: The Price of Fair Access

The Price of Decentralization in Block Building
Burak Öz, Fei Wu, Luis Correia, Sen Yang, Bruno Mazorra, and Stefanos Leonardos . 25:1–25:24

Price of Censorship: Censorship Resistance and Throughput Under Rational Concurrent Proposers
Aditya Saraf, Ioannis Kaklamanis, Sarisht Wadhwa, and Fatima Elsheimy 26:1–26:24

The Execution Dilemma in Pessimistic Blockchains: Profitability XOR Fair Ordering
Artjom Pugatsov, Can Umut Ileri, and Jérémie Decouchant 27:1–27:26

Session 10: Selfish Mining and Proof-of-Work Rewards

Inside Qubic's Selfish Mining Campaign on Monero: Evidence, Tactics, and Limits
Suhyeon Lee and Hyeongyeong Kim 28:1–28:19

When Does Being Selfish Pay Off? Temporal Composability and Profitability in Selfish Mining
Colin Finkbeiner, Connor Shaw, and Ghada Almashaqbeh 29:1–29:23

Optimal Reward Allocation via Proportional Splitting
Lukas Aumayr, Zeta Avarikioti, Dimitris Karakostas, Karl Kreder, and Shreekara Shastry 30:1–30:23

Session 11: Detecting Adversarial Behavior

Friend or Foe? Anomalous Peers in Monero's P2P Network
Yannik Kopyciok, Stefan Schmid, and Friedhelm Victor 31:1–31:22

Who Plays to Earn? Detecting Bots in Web3 Games via On-Chain Transactions Analysis
Woncheol Jeong, Sohee Choi, and Jiwon Yoon 32:1–32:22

Pre-Model Representation Failures in GNN-Based Smart Contract Vulnerability Detection
Birindwa Prisca Hondi, Chinonso Philip Nwishienyi, Charity Wanja Mwaura, Alia Teto, and Jema David Ndibwile 33:1–33:12

Session 12: Empirical Studies of Fees and Trading

EVM Workloads in the Wild: Evidence for Multi-Dimensional Gas Metering, State Growth, Delayed Execution, and Parallelism
Lioba Heimbach, Kushal Babel, and Jason Milionis 34:1–34:24

Price Elasticity of Gas Demand on L1 and L2: Evidence from Ethereum and Arbitrum
Pranay Anchuri and Akaki Mamageishvili 35:1–35:23

Who Loses and Why? Market Microstructure of Perpetual Futures
Ruizhe Jia, Ciamac C. Moallemi, Shouqiao Wang, and Xinmeng Zeng 36:1–36:3

Preface

The 8th International Conference on Advances in Financial Technologies (AFT '26) was held at the London School of Economics in London, UK, on October 6–9, 2026. This proceedings volume contains 36 papers selected from a record-high 185 submissions. This eighth edition of the conference also marks the fourth year in which its proceedings are published in LIPIcs – the Leibniz International Proceedings in Informatics series. The conference has been independently organized for the last four years; this year, AFT became an annual conference of the Stichting Financial Cryptography/IFCA, the International Financial Cryptography Association.

The paper selection process followed common practices in computer science. Each submission was considered and assessed by several Program Committee members (at least three) and external reviewers as part of a double-blind review process. Each accepted paper was presented in a 15-minute live talk, followed by a 5-minute question/answer session. The opening keynote was delivered by Dr. Raphael Auer, founder and president of the Central Bank Research Association (CEBRA). AFT'26 also featured an associated workshop, DeTaP – Workshop on Decentralisation: Theory and Practice, organised by Christina Ovezik and Mojtaba Tefagh.

We would like to thank all the members of the Program Committee as well as the external reviewers for their efforts in selecting the conference program, and all authors for submitting their work for consideration. We are also grateful to the AFT steering committee for their support and guidance throughout the process. We gratefully acknowledge the industry sponsors whose financial support was once again essential to running AFT:

Gold level

- a16z crypto
- Input Output

Silver level

- IC3
- Sui

Finally, we are very grateful to Andy Lewis-Pye for his service and invaluable contributions as General Chair. We would also like to thank Jasna Begic, of the LSE Department of Mathematics, for her help in organizing the event, as well as the Blockchain Technology Laboratory (BTL) at the University of Edinburgh for its support.

Edinburgh, UK
Essex, UK
October 2026

Aggelos Kiayias
Maria Kyropoulou

8th Conference on Advances in Financial Technologies (AFT 2026).
Editors: Aggelos Kiayias and Maria Kyropoulou
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Program Committee

Lukas Aumayr
University of Edinburgh & Common Prefix

Rainer Böhme
University of Innsbruck

Jing Chen
Tsinghua University

Jeremy Clark
Concordia University

Rachid Elazouzi
Avignon University

Matheus Venturyne Xavier Ferreira
University of Virginia

Aris Filos-Ratsikas
University of Edinburgh

Luciano Freitas
Nomadic Labs

Robin Fritsch
Offchain Labs

Arthur Gervais
UCL

Giacomo Giuliari
Mysten Labs

Bernhard Haslhofer
Complexity Science Hub

Ethan Heilman
Cloudflare

Lioba Heimbach
Category Labs

Aljosha Judmayer
University of Vienna

Ari Juels
Cornell Tech and IC3

Aniket Kate
Purdue University / Supra Research

Lucianna Kiffer
IMDEA Networks

Elias Koutsoupias
University of Oxford

Stefanos Leonardos
King's College London

Jiasun Li
GMU

Chen-Da Liu-Zhang
Lucerne University of Applied Sciences and Arts

Akaki Mamageishvili
Offchain Labs

Vangelis Markakis
Athens University of Economics & Business, Input Output

Francisco Marmolejo-Cossio
Boston College

Patrick McCorry
Arbitrum Foundation

Ciamac Moallemi
Columbia / Paradigm/ Uniswap Labs

Pedro Moreno-Sanchez
IMDEA Software Institute, MPI-SP

Joachim Neu
a16z Crypto Research

Paolo Penna
IOG

Georgios Piliouras
Google DeepMind & SUTD

Kaihua Qin
University of Warwick

Geoffrey Ramseyer
Stellar Development Foundation

Christian Rechberger
TU Graz and TACEO

Pierre-Louis Roman
Independent

8th Conference on Advances in Financial Technologies (AFT 2026).
Editors: Aggelos Kiayias and Maria Kyropoulou
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Tim Roughgarden
Columbia University & a16z crypto

Arnab Roy
Mysten Labs

Qihong Ruan
Agentic Sciences, San Francisco

Pietro Saggese
IMT School for Advanced Studies Lucca

Fahad Saleh
University of Florida

Clara Shikhelman
Chaincode Labs

Parwat Singh Anjana
Supra.com

Alberto Sonnino
Mysten Labs and University College London (UCL)

Nimrod Talmon
Ben-Gurion University, Input Output

Qiang Tang
The University of Sydney

Florian Tschorsch
TU Dresden

Sara Tucci Piergiovanni
CEA LIST, Paris-Saclay University

Marie Vasek
University College London

Andreas Veneris
University of Toronto

Kanye Ye Wang
University of Macau

Ke Wu
University of Michigan

Mengqian Zhang
Carnegie Mellon University

Dionysis Zindros
Common Prefix

Steering Committee

Ittai Abraham (co-chair)
Intel

Dan Boneh
Stanford University

Christian Cachin
University of Bern

Ittay Eyal (co-chair)
Technion

Giulia Fanti
CMU

Maurice Herlihy
Brown University

Tim Roughgarden
Columbia University

Eli Ben Sasson
Technion

Emin Gun Sirer (co-chair)
Cornell University

Rainer Böhme ('24 PC chair)
University of Innsbruck

Lucianna Kiffer ('24 PC chair)
IMDEA Networks

Nicolas Christin ('25 PC chair)
Carnegie Mellon University

Zeta Avarikioti ('25 PC chair)
TU Wien and Common Prefix

Aggelos Kiayias ('26 PC chair)
University of Edinburgh and IOG

Maria Kyropoulou ('26 PC chair)
University of Essex

8th Conference on Advances in Financial Technologies (AFT 2026).
Editors: Aggelos Kiayias and Maria Kyropoulou
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

8th Conference on Advances in Financial Technologies (AFT 2026).
Editors: Aggelos Kiayias and Maria Kyropoulou

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

External Reviewers

Rouwaida Abdallah
Paris-Saclay University, CEA LIST

Yackolley Amoussou Guenou
Panteon Assas University

Lorenz Breidenbach
Chainlink Labs

Eyal Briman
BGU

Martin Ceresa
IOG

Hongyin Chen
Technion

Georgios Chionas
University of Liverpool

Andrei Constantinescu
Category Labs

Bernardo David
IT University of Copenhagen
& Common Prefix

Alvaro Garcia Perez
Paris Saclay University, CEA LIST

Mathieu Gestin
CEA

Rohit Kapoor
Supra Research

Stefan Kitzler
Complexity Science Hub

Charalampos Kleitsikas
King's College London

Jichen Li
Tsinghua University

Yuhao Li
Columbia University

Yuan Lu
A*STAR Singapore

Zhenliang Lu
City U of Hong Kong

Florian Lugstein
TU Graz

Fredrik Meisingseth
TU Graz

Katherine Molinet
University of Edinburgh

Shibam Mukherjee
TU Graz and Know Center

Roman Oliynykov
Kharkiv Uni

Luca Pennella
SnT, University of Luxembourg

Maxence Perion
Universite Paris-Saclay, CEA LIST

Manaswini Piduguralla
Google

Alexandre Rapetti
CEA LIST, Paris-Saclay University

Fabian Schmid
TU Graz

Judith Senn
University of Innsbruck

Joel Van der Boo
BGU

Yann Vonlanthen
Ethereum Foundation

Fei Wu
King's College London

Jiajun Xin
The University of Sydney

Sen Yang
Yale University

Yuchen Ye
The University of Sydney

Xiaokuan Zhang
GMU

Brian Zhu
Columbia University

8th Conference on Advances in Financial Technologies (AFT 2026).
Editors: Aggelos Kiayias and Maria Kyropoulou

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

List of Authors

Ghada Almashaqbeh (29)
University of Connecticut, Storrs, CT, USA

Kaya Alpturer (23)
Princeton University, NJ, USA

Pranay Anchuri (35)
Offchain, Princeton, United States

Lukas Aumayr (1, 30)
University of Edinburgh, UK; Common Prefix, Edinburgh, UK

Zeta Avarikioti (1, 30)
TU Wien, Austria; Common Prefix, Vienna, Austria

Kushal Babel (34)
Category Labs, New York, NY, USA

Nuno Braz (17)
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal

Christian Cachin (5)
Institute of Computer Science, University of Bern, Switzerland

Hongyin Chen (11)
Technion, Haifa, Israel

Sohee Choi (32)
Korea University, Seoul, Republic of Korea

Luis Correia (25)
Flashbots, London, UK

Miguel Correia (17)
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal

Bernardo David (8)
IT University of Copenhagen (ITU), Denmark; Common Prefix, Copenhagen, Denmark

Jérémie Decouchant (27)
Delft University of Technology, The Netherlands

Sam Devorsetz (12)
Computer Science Department, Brown University, Providence, RI, USA

Fatima Elsheimy (26)
Yale University, New Haven, CT, USA

Saba Eskandarian (2)
University of North Carolina at Chapel Hill, NC, USA

Ittay Eyal (11)
Technion, Haifa, Israel

Antonio Fernández Anta (9)
IMDEA Software Institute, Madrid, Spain; IMDEA Networks Institute, Madrid, Spain

Matheus V. X. Ferreira (18)
University of Virginia, Charlottesville, VA, USA

Colin Finkbeiner (29)
University of Connecticut, Storrs, CT, USA

Lioba Heimbach (34)
Category Labs, Zurich, Switzerland

Maurice Herlihy (12)
Computer Science Department, Brown University, Providence, RI, USA

Birindwa Prisca Hondi (33)
Carnegie Mellon University Africa, Kigali, Rwanda

Nicholas Hope (23)
Talarion, San Francisco, CA, USA

Antonio Ken Iannillo (3)
University of Luxembourg, Luxembourg

Can Umut Ileri (27)
IOTA Foundation, Berlin, Germany; Delft University of Technology, The Netherlands

Woncheol Jeong (32)
Korea University, Seoul, Republic of Korea

Ruizhe Jia (36)
Stanford University, CA, USA

Philipp Jovanovic (10)
Mysten Labs, Palo Alto, CA, USA; University College London (UCL), UK

Ioannis Kaklamanis (26)
Yale University, IC3, New Haven, CT, USA

Dimitris Karakostas (8, 30)
Common Prefix, Athens, Greece

Arivarasan Karmegam (9)
IMDEA Networks Institute, Madrid, Spain; Universidad Carlos III de Madrid, Spain

Lucianna Kiffer (9)
IMDEA Networks Institute, Madrid, Spain

Hyeongyeong Kim (28)
Korea University, Seoul, Republic of Korea

Yannik Kopyciok (31)
Technische Universität Berlin, Germany

Karl Kreder (30)
Dominant Strategies, Austin, TX, USA

Suhyeon Lee (28)
ShardLab, Seoul, Republic of Korea; Hashed Open Research, Seoul, Republic of Korea

David Lehnherr (5)
Institute of Computer Science, University of Bern, Switzerland

Stefanos Leonardos (25)
King's College London, London, UK

Reina Ke Xin Li (13)
University of Toronto, Canada

Yunwen Liu (21)
COSIC, KU Leuven, Belgium

Matteo Maffei (1)
TU Wien, Austria

Akaki Mamageishvili (7, 16, 20, 35)
Offchain, Zurich, Switzerland

Bruno Mazorra (16, 19, 25)
Flashbots, Barcelona, Spain

Jason Milionis (34)
Category Labs, New York, NY, USA

Ciamac C. Moallemi (14, 36)
Decision, Risk, and Operations Division, Graduate School of Business, Columbia University, New York, NY, USA; Paradigm, New York, NY, USA; Uniswap Labs, New York, NY, USA

Niusha Moshrefi (15)
Princeton University, NJ, USA

Charity Wanja Mwaura (33)
Carnegie Mellon University Africa, Kigali, Rwanda

Viraj Nadkarni (15)
Princeton University, NJ, USA

Kartik Nayak (2)
Duke University, Durham, NC, USA

Jema David Ndibwile (33)
Carnegie Mellon University Africa, Kigali, Rwanda

Joachim Neu (4)
a16z Crypto Research, New York, NY, USA

Chinonso Philip Nwishienyi (33)
Carnegie Mellon University Africa, Kigali, Rwanda

Minghao Pan (16, 19)
California Institute of Technology, Pasadena, CA, USA; Flashbots, Pasadena, CA, USA

Andreas Park (13)
University of Toronto, Canada

Jonathan Passerat-Palmbach (22)
Flashbots, London, UK; Imperial College London, UK

Andrea Pelosi (1)
TU Wien, Austria

Paolo Penna (6)
IOG, Zurich, Switzerland

Alexandru Popa (9)
University of Bucharest, Romania

Diogo Poças (17)
Instituto de Telecomunicações, Instituto Superior Técnico, Universidade de Lisboa, Portugal

Artjom Pugatsov (27)
Delft University of Technology, The Netherlands

Ranvir Rana (15)
Kaleidoscope Blockchain Inc., Raleigh, NC, USA

Max Resnick (4)
Anza, New York, NY, USA

Dan Robinson (14)
Paradigm, New York, NY, USA

Francisco Javier Becerra Sanchez (3)
University of Luxembourg, Luxembourg

Aditya Saraf (26)
Cornell University, Ithaca, NY, USA

Roozbeh Sarenche (21)
COSIC, KU Leuven, Belgium

Giulia Scaffino (4)
TU Wien, Austria; Common Prefix, Vienna, Austria

Christoph Schlegel (16, 19, 20)
Flashbots, Zurich, Switzerland

Stefan Schmid (31)
Technische Universität Berlin, Germany

Manvir Schneider (6)
Cardano Foundation, Zurich, Switzerland

Shreekara Shastry (30)
Dominant Strategies, Austin, TX, USA

Connor Shaw (29)
University of Connecticut, Storrs, CT, USA

Srisht Fateh Singh (13)
University of Toronto, Canada

Alberto Sonnino (10)
Mysten Labs, Palo Alto, CA, USA; University College London (UCL), UK

Radu State (3)
University of Luxembourg, Luxembourg

Christos Stefo (1)
TU Wien, Austria

Benny Sudakov (7)
ETH Zurich, Switzerland

Ali Taslimi (20)
Entropy Advisors, London, UK

Alia Teto (33)
Carnegie Mellon University Africa, Kigali, Rwanda

Zhengwei Tong (2)
Duke University, Durham, NC, USA

Haygen Tsoi (10)
University College London (UCL), UK

Amit Vaisman (11)
Technion, Haifa, Israel

Andreas Veneris (13)
University of Toronto, Canada

Friedhelm Victor (31)
TRM Labs, San Francisco, CA, USA

Juan Villacis (5)
Institute of Computer Science, University of Bern, Switzerland

Pramod Viswanath (15)
Princeton University, NJ, USA

Sarisht Wadhwa (26)
Duke University, Durham, NC, USA

Shouqiao Wang (36)
Columbia Business School, Columbia University, New York, NY, USA

S. Matthew Weinberg (23)
Princeton University, NJ, USA

François-Xavier Wicht (5)
Institute of Computer Science, University of Bern, Switzerland

Robin Linus Woll (1)
Stanford University, CA, USA; ZeroSync, Zug, Switzerland

Fei Wu (24, 25)
King's College London, London, UK

Sen Yang (25)
Yale University, New Haven, CT, USA; IC3, New York, NY, USA

Jiwon Yoon (32)
Korea University, Seoul, Republic of Korea

Alexei Zamyatin (1)
BOB, London, UK

Xinmeng Zeng (36)
Stanford University, CA, USA

Brian Zhu (14)
Department of Industrial Engineering and Operations Research, Columbia University, New York, NY, USA

Burak Öz (20, 24, 25)
Flashbots, Berlin, Germany

BitVM: Quasi-Turing Complete Computation on Bitcoin

Lukas Aumayr
University of Edinburgh, UK
Common Prefix, Edinburgh, UK

Zeta Avarikioti
TU Wien, Austria
Common Prefix, Vienna, Austria

Robin Linus Woll
Stanford University, CA, USA
ZeroSync, Zug, Switzerland

Matteo Maffei
TU Wien, Austria

Andrea Pelosi
TU Wien, Austria

Christos Stefo
TU Wien, Austria

Alexei Zamyatin
BOB, London, UK

Abstract

A long-standing question in the blockchain community is which class of computations is efficiently expressible in cryptocurrencies with limited scripting languages, such as Bitcoin Script. Such languages expose a reduced trusted computing base, thereby being less prone to hacks and vulnerabilities, but have long been believed to support only limited classes of payments.

In this work, we confute this long-standing belief by showing for the first time that arbitrary computations can be encoded in today's Bitcoin Script without introducing any language modification or additional security assumptions, such as trusted hardware, trusted parties, or committees with an honest majority. We present BitVM, a two-party protocol that realizes a generic virtual machine by combining cryptographic primitives and economic incentives. We conduct a formal analysis of BitVM, characterizing its functionality, system assumptions, and security properties. We further demonstrate the practicality of our approach by implementing a prototype and performing an experimental evaluation: in the optimistic case (i.e., when parties agree), our protocol requires just three on-chain transactions, whereas in the pessimistic case, the number of transactions grows logarithmically with the size of the virtual machine. We exemplify the deployment potential of BitVM by building a Bitcoin-sidechain bridge application. This work not only solves a long-standing theoretical problem, but it also promises a strong practical impact, enabling the development of complex applications in Bitcoin.

2012 ACM Subject Classification Security and privacy → Security protocols; Security and privacy → Distributed systems security

Keywords and phrases Bitcoin, quasi-Turing completeness, off-chain protocols, bridge

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.1

Related Version *Full Version*: `https://eprint.iacr.org/2024/1995.pdf` [5]

Supplementary Material *Software (source code)*: `https://github.com/BitVM/bitvm-js` [2]
archived at `swh:1:dir:fd9022956a1774f59cf49dacf6e87d32f90863aa`

Funding This work was partially supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement 101141432-BlockSec), by the Austrian Science Fund (FWF) through the SFB SpyCode project (F8510-N and F8512-N), by the Vienna Science and Technology Fund (WWTF) through the projects 10.47379/ICT25056 and 10.47379/ICT22045, and by Input Output (`https://iohk.io`) through their funding of the Edinburgh Blockchain Technology Lab.



© Lukas Aumayr, Zeta Avarikioti, Robin Linus Woll, Matteo Maffei, Andrea Pelosi, Christos Stefo, and Alexei Zamyatin;
licensed under Creative Commons License CC-BY 4.0



1 Introduction

Smart contracts are a foundational component of modern blockchain systems, enabling decentralized applications (dApps) and programmable money without the need for trusted intermediaries. These self-executing programs have unlocked a wide range of use cases, spanning finance, governance, and supply chain management, by enforcing agreement logic directly on-chain.

While some blockchains, such as Ethereum, support quasi-Turing complete[1] execution through bytecode languages like the Ethereum Virtual Machine (EVM), others, most notably Bitcoin, opt for a minimalist approach. Bitcoin Script deliberately limits expressiveness to minimize complexity and attack surface, trading expressiveness for security and stability.

This trade-off has given rise to a long-standing open question: *Can general-purpose computation be supported on Bitcoin, using only its existing scripting language and consensus rules?* Unlocking such expressiveness could expand Bitcoin's utility for dApps and decentralized finance (DeFi), while preserving its conservative security model.

The prevailing belief in the blockchain community is that Bitcoin Script cannot support general-purpose computation in practice. This belief stems from several structural limitations of the language: it is stateless, lacks loops and recursion, and enforces strict constraints on script and transaction size. These properties make it well-suited for simple functionalities, such as multisignature payments or hashed timelock contracts, but appear to rule out the execution of complex logic on-chain.

Several proposals have attempted to overcome these constraints, either by extending Bitcoin with new opcodes (e.g., covenants [12]), encoding computations into low-level counter machines [13], or relying on trusted execution environments [18, 24] and oracles [33, 19]. However, these approaches either require consensus changes, incur prohibitive on-chain costs, or compromise Bitcoin's trust model. As a result, it has long been assumed that supporting arbitrary computation on Bitcoin would require trade-offs incompatible with its conservative design philosophy.

Contributions. In this work, we challenge this long-standing belief by showing that arbitrary (bounded) computations can be executed securely on Bitcoin in a practical manner. We introduce BitVM[2], a two-party protocol that enables expressive, verifiable off-chain computation using only existing Bitcoin features. BitVM requires no consensus changes, no new opcodes, and no trusted hardware or oracles.

The core idea behind BitVM is to shift computation off-chain while retaining on-chain verifiability through a fraud-proof mechanism. Specifically, a prover submits a claim about the output of a bounded computation. The verifier can either accept the result or, in case of disagreement, initiate an interactive dispute protocol. This protocol relies on a custom virtual machine that encodes computation as an execution trace, enabling the parties to identify a point of disagreement and verify a single step of computation using Bitcoin Script.

When both parties agree, the entire process completes with just three on-chain transactions. In case of dispute, BitVM guarantees that verification incurs only a logarithmic number of additional transactions. This makes BitVM both practical and expressive, enabling Bitcoin to support advanced smart contract functionality without compromising its trust model or requiring protocol changes.

---

[1] This term is adopted in the blockchain community to indicate Turing-complete languages that enforce termination by bounding the execution (e.g., via gas consumption in Ethereum) [35].

[2] This work extends and formalizes the original BitVM design, which was conceptualized and developed by Robin Linus [29].

To illustrate how BitVM operates in practice, consider a simple two-party wager: a prover claims to have solved a chess puzzle, and a verifier bets against them. Both parties lock funds into a BitVM contract that verifies the claimed solution. If the verifier agrees, the outcome is settled off-chain. If not, they can initiate an on-chain dispute. The contract then executes the verification step by step, and settles the funds accordingly, using only Bitcoin Script to enforce the outcome. While conceptually simple, this example demonstrates the core functionality of BitVM: enabling trustless agreement on arbitrary computation, with minimal on-chain footprint in the absence of disputes.

The same mechanism underpins more complex applications. A key example, which we outline in this work, is a *trust-minimized bridge between Bitcoin and a sidechain*. In typical designs, a committee of operators handles redemptions from the sidechain to Bitcoin, with security guaranteed only under an honest-majority assumption (t-of-n, where $t > n/2$), e.g., [9, 1]. With BitVM, this trust requirement is minimized to *existential honesty*: security holds as long as *at least one* operator behaves honestly ($t = 1$). Any operator can front coins to a user on Bitcoin and later claim reimbursement by proving correctness; if another committee member disputes the claim, they can initiate a BitVM instance to verify or refute it on-chain. This approach *reduces the trust assumption from honest majority of the bridge committee to existential honesty*, while remaining fully compatible with Bitcoin's existing scripting model. Crucially, such a bridge serves as a foundational building block for a broader ecosystem of decentralized applications: once assets can move between Bitcoin and external execution layers securely and trustlessly, DeFi protocols, from decentralized exchanges to lending platforms, can operate on top of Bitcoin without compromising its security guarantees.

The appeal of the BitVM paradigm is witnessed by the massive investments of the industry (more than 150M USD[3]) and the number of BitVM-based protocols already in development [23, 38, 17, 4].

This paper makes the following contributions:

- We present BitVM, the first protocol to encode quasi-Turing complete computations in Bitcoin Script, requiring no consensus changes or trusted third parties (Section 5).
- We provide a formal analysis of BitVM, characterizing its functionality, system assumptions, and security properties (Section 6).
- To evaluate BitVM's performance, we implement a prototype of BitVM in JavaScript.[4] Optimistic execution completes in three on-chain transactions, costing approximately $5,832$ satoshis[5], equivalent to \$4.26 (as of May 2026). In case of disputes, the on-chain footprint grows logarithmically with the computation size. For a virtual machine with 2^{32} memory cells and steps – comparable to a high-end 1990s workstation – settlement requires up to 81 transactions, at a cost of approximately $732,000$ satoshis, equivalent to \$534 (Section 7).
- We demonstrate the capabilities of BitVM by constructing a trust-minimized bridge protocol between Bitcoin and a sidechain, reducing the traditional honest-majority assumption to that of a single honest operator (Section 8).

Related work. Several works have attempted to overcome the limited expressiveness of Bitcoin Script and enable more complex smart contracts by combining UTXOs and scripts, effectively splitting functionality across multiple transactions. BitML [14] provides a high-level, domain-specific language for smart contracts with an associated compiler that produces

[3] `https://cryptorank.io`
[4] The prototype is available at [2].
[5] A *satoshi* is a fraction of a bitcoin, i.e., $1sat = 10^{-8}$₿.

Bitcoin transactions, illustrating Bitcoin's potential for intricate smart contract designs [11]. These methods, however, incur substantial on-chain costs, as compiled programs often result in numerous large transactions that must ultimately be recorded on-chain, and they support only a restricted class of contracts.

To mitigate these costs, some approaches leverage Trusted Execution Environments (TEEs). FastKitten [18] facilitates off-chain computation within a secure hardware enclave, but relies on collateral, rational adversaries, trusted TEE operators, and a limited contract duration. POSE [24] improves upon FastKitten by removing collateral requirements and time constraints, and by enhancing privacy, but it continues to rely on trusted TEE hardware.

A different approach uses Hashed Timelock Contracts (HTLCs) to shift computation off-chain by encoding outcomes in preimages of hash functions [8], similar in spirit to *state channels* on Ethereum [22, 20]. This model underpins constructions such as Discreet Log Contracts (DLCs) [19] and oracle-based conditional payments [33], which depend on (semi-) trusted oracles to attest to specific events. A key limitation of these approaches is that all possible outcomes must be known and encoded in advance. Consequently, they cannot support applications like the chess puzzle or the bridge example, where the correct outcome, such as the solution to a puzzle or the identity of the party holding funds on the sidechain, may not be known a priori to any participant.

Unlike prior works, BitVM enables quasi-Turing complete computation on Bitcoin without consensus changes or relying on external trust assumptions, such as TEEs and semi-trusted oracles. It is the first trustless protocol to allow arbitrary, bounded computation on Bitcoin, while incurring logarithmic on-chain cost in the worst case, unlocking a range of potential applications. Table 1 situates BitVM among prior Bitcoin smart contract approaches.

Table 1 Comparison of Bitcoin-based smart contract approaches. n denotes the upper bound on computational steps, and QT refers to Quasi-Turing completeness.

Approach	Expressiveness	Extra Assumptions	On-chain cost
BitML [14, 11]	QT	None	$O(n)$
TEEs [18, 24]	QT	TEE	$O(n)$
Gen. Channels [8]	Bitcoin	None	$O(1)$
Oracles [19, 33]	QT	Trusted oracle	$O(1)$
BitVM (our work)	QT	None	$O(\log(n))$

Follow-up work [30], informally referred to as BitVM2, targets bridges between Bitcoin and Layer-2 systems. It compiles a SNARK verifier into Bitcoin Script, splits it across transactions, and commits to intermediary states on-chain. This supports permissionless dispute resolution but incurs on-chain cost linear in the verifier size, requiring a transaction that fills a 4MB Bitcoin block in case of disputes. Our BitVM is instead better suited to permissioned settings and resolves disputes at logarithmic on-chain cost, with formal security guarantees. Beyond the asymptotic gap, it is also cheaper in practice: Table 2 shows on-chain costs an order of magnitude below BitVM2 in both the optimistic and dispute cases, where for a fair comparison we size BitVM at 2^{32} memory cells, sufficient to encode the SNARK verifier. Finally, BitVM runs RISC-V programs through off-the-shelf compilers, whereas BitVM2 requires compiling the verifier (around 4GB) into Bitcoin Script, a cumbersome and error-prone task.

A recent development, informally termed BitVM3 [40], pushes this paradigm further by trading on-chain cost for off-chain computation: it encodes the BitVM2 SNARK verifier as a garbled circuit. The operator garbles the verifier off-chain and commits to it during setup; a challenger recovers a fraud-proof secret off-chain iff the claim is incorrect, so a dispute is settled by revealing a single secret on-chain. BitPriv [6] and BABE [26] build on this

paradigm, the former adding input privacy and the latter reducing verification costs under a permissioned challenger set. The trade-off is steep off-chain cost: each garbled circuit reaches 40GB, yielding terabytes of storage and communication overhead.

Table 2 summarizes the three BitVM paradigms. BitVM is the first to introduce this paradigm; BitVM2 and BitVM3 are follow-ups that trade it off differently, the former for permissionless disputes at linear on-chain cost, the latter for lower on-chain cost at the expense of heavy off-chain garbling.

Table 2 On-chain and off-chain costs comparison between BitVM paradigms.

Approach	Optimistic ($)	Dispute ($)	Optimistic (vB)	Dispute (vB)	Off-chain cost
BitVM (our work)	4.26	534	1.9k	244k	Low
BitVM2 [31]	38.7	11,680	16.8k	496.7k	Low
BitVM3 [40]	5.26	5.48	2.4k	2.5k	High

Finally, we note that the optimistic verification paradigm behind BitVM is inspired by optimistic rollup protocols [27]. While optimistic rollups are prominent in the Ethereum ecosystem, adapting this principle to Bitcoin, which offers limited scripting flexibility and strict block size constraints, remained an open question until now.

2 Model

BitVM is a protocol between two mutually distrusting parties, the prover P and the verifier V, designed to enable P to prove on the Bitcoin blockchain that the outcome of a pre-agreed computation with V was performed correctly. Concretely, for an agreed-upon Turing-complete program Π, a BitVM instance secures collateral from both parties and it enables P to enforce a transaction on-chain based on the outcome $\Pi(x)$ for a specific input x. In other words, $\Pi(x)$ dictates the payout of the funds within the BitVM instance, typically allocating them to P and V[6]. If P or V stop collaborating during protocol execution, after a designated period all the funds are allocated to the other party.

2.1 System model

We assume time advances in discrete rounds $(1, 2, \dots)$. Protocol participants run in probabilistic polynomial time (PPT) in the security parameter κ. We assume synchronous communication, i.e., messages sent between parties arrive at the beginning of the next round, as well as authenticated communication channels. Our protocol employs a hash function modeled as a random oracle $H : \{0,1\}^* \to \{0,1\}^\kappa$ which maps an input of arbitrary length to a fixed κ-sized output. Moreover, our protocol builds upon a distributed ledger protocol (e.g., [25, 39, 7]).

▶ **Definition 1** (Distributed Ledger Protocol). *A distributed ledger protocol is an interactive Turing machine exposing the following functionality on each party.*

- *execute(): executes one protocol round and enables the machine to communicate with the network, invoked by the environment in every round;*
- *write(tx): takes as input a transaction from the environment;*
- *read(): outputs a finite, ordered sequence of transactions, known as* transaction ledger L.

[6] Note that P and V can also agree to allocate the funds to a third party or, more generally, make the funds spendable under any condition that can be expressed in Bitcoin Script.

We denote L_r^P as the output of invoking read() on party P at the end of round r. We restrict honest parties to only include *valid* transactions in their ledgers[7]. As we are interested in building BitVM on Bitcoin, when we present the construction, transactions are deemed (in)valid based on Bitcoin's validation rules (see Section 3.1). However, BitVM can be built on top of any distributed ledger protocol with validation rules as expressive as those of Bitcoin. We assume that our protocol participants have access to the functionality exposed by the distributed ledger protocol, either by being an active participant or by running some (light) client protocol. We are interested in distributed ledger protocols that are safe and live, as defined below (cf. [25, 39, 7]). Given two sequences A and B, we use $A \preceq B$ to mean that A is a prefix of B.

▶ **Definition 2** (Safety). *A distributed ledger protocol is safe, if (i) for any honest party P and any rounds $r_1 \leq r_2$, it holds that $\mathsf{L}_{r_1}^P \preceq \mathsf{L}_{r_2}^P$ and (ii) for any pair of honest parties P_1, P_2 and any pair of rounds r_1, r_2, it holds that $\mathsf{L}_{r_1}^{P_1} \preceq \mathsf{L}_{r_2}^{P_2} \vee \mathsf{L}_{r_2}^{P_2} \preceq \mathsf{L}_{r_1}^{P_1}$.*

▶ **Definition 3** (Liveness). *A distributed ledger protocol execution is live(u), if any transaction that is written to an honest party's ledger at round r, appears in the ledger of all honest parties by round $r + u$, denoted as $\mathsf{L}_{r+u}^{\bigcap}$.*

Throughout this paper, we say "publish a transaction tx (on L)" to denote calling the function write(tx). Furthermore, after publishing a valid transaction tx, we sometimes say "wait until tx appears (on L)", to denote calling the function read() every round until $tx \in \mathsf{L}$, which happens at most after u rounds due to liveness. When presenting the BitVM construction, we sometimes refer to the ledger as blockchain even though the distributed ledger protocol could be realized differently. We say something happens *on-chain* if there are one or more corresponding transactions in the ledger, and something happens *off-chain* if there are no corresponding transactions on the ledger.

There is a ledger state that is induced by a ledger L, denoted as $\mathsf{st}(\mathsf{L})$, by executing each transaction in order, starting with a genesis state. The execution of transactions is captured by a state transition function, taking a state and a transaction and outputting a new state. We denote $\mathsf{bal}_{\mathsf{L}}(P) \in \mathbb{R}_{\geq 0}$ as the balance of party P in the state induced by L. A party can use parts of their balance $\mathsf{in}_P \in [0, \mathsf{bal}_{\mathsf{L}}(P)]$ as *monetary input* for a transaction. For a given ledger L, we define the on-chain (monetary) utility of a transaction $tx \in \mathsf{L}$ for a party P as $w_{\mathsf{L}}(P, tx) := \mathsf{bal}_{\mathsf{L}_2}(P) - \mathsf{bal}_{\mathsf{L}_1}(P)$, where $\mathsf{L}_1 \prec \mathsf{L}$ is the ledger up to (not including) tx and $\mathsf{L}_2 := \mathsf{L}_1 || tx$. Usually, it is obvious which ledger we refer to, so we omit the subscript. In addition to balances of parties, a ledger state $\mathsf{st}(\mathsf{L})$ can include a string $s \in \{0,1\}^*$, denoted as $s \in \mathsf{st}(\mathsf{L})$, if there exists a transaction $tx \in \mathsf{L}$, such that tx contains the string s.

2.2 Threat model

We analyze BitVM in the presence of a PPT adversary that may corrupt any protocol party $\{P, V\}$ during the execution of the protocol. The adversary can corrupt parties, causing them to behave either as *Byzantine* or as *rational* actors. Byzantine parties can deviate arbitrarily from the honest protocol execution. Contrarily, rational parties deviate from the honest protocol execution only when such action increases their monetary utility.

[7] This is not strictly necessary and is done mainly for convenience. Parties could also take an outputted ledger and remove invalid transactions from it.

The protocol gives different guarantees based on the type of corruption. On a high level, we want to show that (i) honest protocol participants are guaranteed their rightful balance even if the other party is Byzantine, (ii) rational parties follow the honest protocol execution, and (iii) if both parties behave rationally, the protocol follows an optimistic execution (which is efficient). We formally define these properties in Section 2.3.

2.3 Protocol goals

The core objectives of BitVM are termed *balance security* and *rational correctness*. Informally, balance security ensures an honest party will not lose their funds against Byzantine counterparties, whereas rational correctness guarantees that rational parties will follow the protocol. To formally define balance security we argue in terms of utility, i.e., the utility of the on-chain state of an honest party after the settlement of a BitVM instance will be at least equal to its utility of the correct final state, regardless of the actions of its counterparty. Rational correctness implies that if both parties are rational, they will commit on-chain the correct final state of the BitVM instance. These properties are standard in the literature: for instance, an honest user of a Lightning channel [36] can always dispute a malicious commitment and claim the channel funds, while rational players will always commit to the last agreed-upon state [37].

We formalize these objectives on a generic primitive, which we call *on-chain state verification* protocol and is defined as follows.

▶ **Definition 4** (On-chain State Verification Protocol)**.** *An on-chain state verification protocol, parameterized over a distributed ledger protocol that outputs a ledger* L*, is a two-party protocol that exposes the two following functionalities:*

- *setup(*$\mathsf{in}_P, \mathsf{in}_V, \Pi, f$*): takes as input monetary inputs* $\mathsf{in}_P \in [0, \mathsf{bal}_{\mathsf{L}}(P)]$ *and* $\mathsf{in}_V \in [0, \mathsf{bal}_{\mathsf{L}}(V)]$ *of parties* P *and* V*, a computable function (or program)* $\Pi : \mathcal{S} \to \mathcal{O}$ *that maps a set of states* $\mathcal{S}$ *to a set of outcomes* $\mathcal{O}$ *and an outcome mapping function* $f : \mathcal{O} \to \mathbb{R}^2_{\geq 0}$*, that maps the set of outcomes* $\mathcal{O}$ *to pairs of utilities* (v_P, v_V) *where* $v_P + v_V \leq \mathsf{in}_P + \mathsf{in}_V$[8] *and returns an instance* $\mathcal{I}$*.*
- *execute(*$\mathcal{I}, x$*): takes as input an instance* $\mathcal{I}$ *returned by the setup function and a function input* $x \in \mathcal{S}$ *(for function* Π*).*

Consider an execution of this primitive for given inputs $\mathsf{in}_P, \mathsf{in}_V$, Π, f, where $\mathcal{I} \leftarrow \text{setup}(\mathsf{in}_P, \mathsf{in}_V, \Pi, f)$, and then execute$(\mathcal{I}, x)$ are called, and finish in round r. Let $\mathcal{T}$ be the set of transactions that are included in $\mathsf{L}^{\bigcap}_{r+u}$ as a result of this execution. Moreover, we denote the utility of party $A \in \{P, V\}$ in $f(\Pi(x))$ by $f_A(\Pi(x))$.

Balance Security. An execution achieves balance security, if it holds that $\sum_{tx \in \mathcal{T}}(w(A, tx)) \geq v_A$ where $v_A = f_A(\Pi(x))$, for any honest $A \in \{P, V\}$.

Rational Correctness. An execution achieves rational correctness, if P and V are rational and $\sum_{tx \in \mathcal{T}}(w(A, tx)) = v_A$ where $v_A = f_A(\Pi(x))$, for any $A \in \{P, V\}$ and $\Pi(x) \in \mathsf{st}(\mathsf{L}^{\bigcap}_{r+u})$.

An on-chain state verification protocol achieves balance security and rational correctness, respectively, if for any $\mathsf{in}_P, \mathsf{in}_V, \Pi, f$ the probability that the corresponding execution does not achieve balance security and rational correctness, respectively, is negligible in κ.

[8] We use an inequality here to account for potential fees paid during protocol execution.

3 Preliminaries

In this section, we present the necessary background concerning Bitcoin Script and the key primitives our construction builds upon.

Notation. Given a sequence $A := (a_1, \ldots, a_n)$, $A[i]$ represents its i-th element. We use $A[i:j]$ to denote the subsequence $(a_i, \ldots, a_j)$. We use $|A|$ to denote the length of a sequence, e.g., $|(a_1, \ldots, a_n)| = n$. For a string $s \in \{0,1\}^*$, we use $|s|_{bit}$ to denote its bit length. For the full notation used throughout the paper, we refer the reader to the full version of the paper [5, Appendix A].

3.1 Transactions in the UTXO model

A user U on a ledger L is identified by the secret-public key pair $(\mathsf{pk}_\mathsf{U}, \mathsf{sk}_\mathsf{U})$; by $\sigma_\mathsf{U}(m)$ we denote the digital signature of U over the message $m \in \{0,1\}^*$.

In the *unspent transaction output* (UTXO) model, a transaction $\mathtt{Tx}$ maps a (non-empty) list of existing, unspent, transaction outputs to a (non-empty) list of new transaction outputs. A transaction output is defined as an attribute tuple $\mathsf{out} := (a\text{฿}, \mathsf{lockScript})$, where $\mathsf{out}.a \in \mathbb{R}_{\geq 0}$ is the amount of coins (expressed in ฿) held by the output out and $\mathsf{out}.\mathsf{lockScript}$ is the condition that needs to be fulfilled to spend it and transfer the coins to a new output, which we also call UTXO. We distinguish the already existing transaction outputs (input of a transaction $\mathtt{Tx}$) from the newly created outputs, calling them $\mathtt{Tx}.\mathsf{inputs}$ and $\mathtt{Tx}.\mathsf{outputs}$, respectively. A transaction input in is defined as $\mathsf{in} := (\mathtt{PrevTx}, \mathsf{outIndex}, \mathsf{lockScript})$, where the output being spent is uniquely identified by specifying the transaction $\mathtt{PrevTx}$ and an output index $\mathsf{outIndex}$. To improve readability, we also give the locking script $\mathsf{lockScript}$ that is being fulfilled and we specify the different transaction spending conditions.

In this paper, we use the following subset of Bitcoin spending conditions (which we describe in the full version of the paper [5, Appendix B]): signature locks, multisignature locks, relative timelocks, and taproot trees. We use these conditions as building blocks, combining them with standard Bitcoin Script constructions using logical operators $\wedge$ (and) and $\vee$ (or).

We formally define a transaction as a tuple $\mathtt{Tx} := (\mathsf{inputs}, \mathsf{witnesses}, \mathsf{outputs})$ where $\mathtt{Tx}.\mathsf{inputs} := [\mathsf{in}_1, \ldots, \mathsf{in}_n]$ are the transaction inputs, $\mathtt{Tx}.\mathsf{outputs} := [\mathsf{out}_1, \ldots, \mathsf{out}_m]$ are the transaction outputs and $\mathtt{Tx}.\mathsf{witnesses} := [\mathsf{w}_1, \ldots, \mathsf{w}_n]$ represents the witness data, i.e., the list of the tuples that fulfill the spending conditions of the inputs, one witness for each input. The locking script of an output is expressed in the scripting language of the ledger. To transfer the coins held in a UTXO, its locking script is executed with a witness as script input and must return True; if successful, the condition is considered fulfilled. If the script execution returns False, the condition is not fulfilled and the UTXO is not spendable[9].

A transaction is valid only if every UTXO in input is unspent, the witnesses fulfill the conditions of the corresponding locking scripts, and the sum of the coins held in the inputs is equal to or greater than the sum of the coins held in the outputs.

[9] In this work, we separate the locking script from the witness for readability. In practice, the protocol is implemented using SegWit [32] transactions, where the locking script is included in the witness.

3.2 Lamport digital signature scheme

Let $h : X \to Y$ be a one-way function, where $X := \{0,1\}^*$ and $Y := \{0,1\}^\lambda$, for a given security parameter λ. Let $m \in \{0,1\}^\ell$ be a ℓ-bit message, with $\ell \in \mathbb{N}_{>0}$. A *Lamport digital signature scheme* [28] Lamp consists of a triple of algorithms $(\mathsf{KeyGen}, \mathsf{Sig}, \mathsf{Vrfy})$, where:

- $(pk_\mathcal{M}, sk_\mathcal{M}) \leftarrow \mathsf{Lamp.KeyGen}(\ell)$ (cf. Algorithm 1), is a Probabilistic Polynomial Time (PPT) algorithm that takes as input a positive integer ℓ and returns a key pair, consisting of a secret key $sk_\mathcal{M}$ and a public key $pk_\mathcal{M}$ which can be used for one-time signing an ℓ-bit message. We use $\mathcal{M} = \{0,1\}^\ell$ as an alias for the ℓ-bit message space.
- $c_m \leftarrow \mathsf{Lamp.Sig}_{sk_\mathcal{M}}(m)$ (cf. Algorithm 2), is a Deterministic Polynomial Time (DPT) algorithm parameterized by a secret key $sk_\mathcal{M}$, that takes as input a message $m \in \mathcal{M}$ and outputs the signature c_m, which we also call (Lamport) commitment.
- $\{\mathsf{True}, \mathsf{False}\} \leftarrow \mathsf{Lamp.Vrfy}_{pk_\mathcal{M}}(m, c_m)$ (cf. Algorithm 3), is a DPT algorithm parameterized by a public key $pk_\mathcal{M}$ that takes as input a message m, a signature c_m, and outputs True iff c_m is a valid signature for m generated by the secret key $sk_\mathcal{M}$, corresponding to $pk_\mathcal{M}$, i.e., $(pk_\mathcal{M}, sk_\mathcal{M})$ is a key pair generated by $\mathsf{Lamp.KeyGen}$.

Algorithm 1 The key generation algorithm $\mathsf{Lamp.KeyGen}$ for a ℓ-bit message space $\mathcal{M}$. In the following algorithms, we use matrix notation, i.e., for a given two-dimensional matrix a, $a[i,j]$ refers to the element at row i and column j of it.

function $\mathsf{Lamp.KeyGen}(\ell)$
  Let $sk_\mathcal{M} \leftarrow \begin{pmatrix} x[0,0], \ldots, x[0,\ell-1] \\ x[1,0], \ldots, x[1,\ell-1] \end{pmatrix}$, where every element $x[i,j]$ is sampled uniformly at random from the set X;
  for $i = 0, 1$ and $j = 0, \ldots, \ell - 1$ **do**
    $y[i,j] \leftarrow h(x[i,j])$;
  Let $pk_\mathcal{M} \leftarrow \begin{pmatrix} y[0,0], \ldots, y[0,\ell-1] \\ y[1,0], \ldots, y[1,\ell-1] \end{pmatrix}$;
  return $(sk_\mathcal{M}, pk_\mathcal{M})$.

Algorithm 2 The Lamport signature algorithm $\mathsf{Lamp.Sig}$, parameterized over a secret key $sk_\mathcal{M}$ for a ℓ-bit sized message space $\mathcal{M}$.

function $\mathsf{LampSig}_{\mathsf{sk}_\mathcal{M}}(m)$
  for $i = 0, \ldots, \ell - 1$ **do**
    Let $c_m[i] \leftarrow sk_\mathcal{M}[m[i], i]$;
  return c_m.

Algorithm 3 Lamport verification algorithm $\mathsf{Lamp.Vrfy}$, parameterized over a public key $pk_\mathcal{M}$ for a ℓ-bit message space $\mathcal{M}$.

function $\mathsf{Lamp.Vrfy}_{pk_\mathcal{M}}(m,\, c_m)$
  for $i = 0, \ldots, \ell - 1$ **do**
    if $h(c_m[i]) \neq pk_\mathcal{M}[m[i], i]$ **then**
      return False;
  return True.

Lamport signatures are secure one-time signatures. Given a message space $\mathcal{M}$, it is possible to sign any message $m \in \mathcal{M}$ by using the secret key $sk_\mathcal{M}$ of the key pair $(sk_\mathcal{M}, pk_\mathcal{M})$, i.e., the key pair associated to $\mathcal{M}$. When the message m is signed and c_m is created, the key

pair becomes bound to m. No polynomially bounded adversary is able to forge a signature for a different message $m' \neq m$ with non-negligible probability. However, if the signer uses the same secret key $sk_{\mathcal{M}}$ to sign another different ℓ-bit message $m'' \neq m$, they can be held accountable. We call this action *equivocation* and we show how to detect it in Algorithm 4.

Notice that signing the ℓ-bit message m with the secret key $sk_{\mathcal{M}}$ consists in revealing for every bit $i = 0, \ldots, \ell - 1$ of m one of the two preimages that compose the $i-th$ column of secret key $sk_{\mathcal{M}}$, namely, revealing $x[0, i]$ to claim that $m[i] = 0$, or revealing $x[1, i]$ to claim that $m[i] = 1$. When the signer reveals both $x[0, i], x[1, i]$ for any bit i, they are equivocating.

For a formal discussion about one-time security and a proof that Lamport signatures are one-time secure (assuming the existence of one-way functions), see [15]. One-time security is crucial for the correctness of BitVM as it enables the signer of a message to make a non-repudiable commitment to that message. Lamport signatures are implementable using Bitcoin Script, as demonstrated in [2].

Algorithm 4 The CheckEquivocation algorithm for a bit $b \in \mathcal{B} = \{0, 1\}$. The input is the corresponding public key $pk_{\mathcal{B}}$ and two preimages $x', x'' \in X$.

```
1: function CheckEquivocation(pk_B, x', x'')
2:     if (h(x') = pk_B[0,0] and h(x'') = pk_B[1,0]) then
3:         return True;
4:         ▷ The committer is trying to commit to both 0 and 1 for the bit b. ◁
5:     else
6:         return False.
```

In the following, we are interested in Lamport signatures as a mechanism to enable a party to *commit* to (single or multiple bits) messages. Thus, we will refer to Algorithm 2 as Comm instead of Lamp.Sig and to Algorithm 3 as CheckComm instead of Lamp.Vrfy.

3.3 Stateful Bitcoin scripting

Although Bitcoin's scripting language is stateless, clever use of one-time digital signature schemes (e.g., Lamport signatures) enables state preservation across Bitcoin transactions.

Consider the following example: Let a user U hold a Lamport key pair $(sk_{\mathcal{M}}, pk_{\mathcal{M}})$ associated with $\mathcal{M}$, the set of all ℓ-bit messages. We can think of $\mathcal{M}$ as a variable that can hold any ℓ-bit string. U can assign a value m to $\mathcal{M}$ by creating the commitment $c_m \leftarrow \mathsf{Comm}_{\mathsf{sk}_{\mathcal{M}}}(m)$.

By hard-coding $\mathsf{CheckComm}_{pk_{\mathcal{M}}}$ for a public key $pk_{\mathcal{M}}$ in the locking script of multiple outputs, this variable assignment can not only be verified but also transferred from one output to another, effectively establishing a global state in Bitcoin. This is accomplished by reading m and c_m from the unlocking script of one output and passing them to another output through its witness. For example, consider two different transactions $\mathtt{Tx_1} := (*, *, [\mathsf{out}_1, *])$ and $\mathtt{Tx_2} := (*, *, [\mathsf{out}'_1, *])$, where the outputs are defined as $\mathsf{out}_1 := (a\ddot{\mathbb{B}}, \mathsf{CheckComm}_{pk_{\mathcal{M}}})$ and $\mathsf{out}'_1 := (b\ddot{\mathbb{B}}, \mathsf{CheckComm}_{pk_{\mathcal{M}}})$. To unlock both out_1 and out'_1, a Lamport commitment c_m must be provided. Since the same Lamport public key appears in both scripts, every party in the network knows that when U unlocks these scripts, U is assigning a value to the same variable $\mathcal{M}$. Following from *one-time security*, no user other than U can assign a different value to $\mathcal{M}$ without knowing $sk_{\mathcal{M}}$. Moreover, U cannot assign two different values $m_1 \neq m_2$ to $\mathcal{M}$ without equivocating, which is detectable and can be punished on-chain.

4 BitVM Virtual Machine

In the BitVM protocol, both parties employ a *Virtual Machine* (VM) to run off-chain any deterministic program Π. Although the underlying concept closely resembles an *abstract machine*, we choose to retain the term "VM" to stay consistent with the original naming of the construction. In this section, we describe the components of the VM and demonstrate how to initialize them for practical deployment of the protocol.

VM components. At a high level, the virtual machine (VM) executes programs composed of instructions written in a VM-compatible language. While the program is running, the VM continuously performs an instruction cycle, or *state transition function*. In each cycle, the VM fetches the instruction indicated by the program counter, loads the values stored at specific memory addresses referenced by the instruction, executes the operation defined by the instruction on those values, stores the result at the designated memory address, and updates the program counter accordingly (cf. Definition 6).

This process repeats until the program terminates or reaches a predefined execution limit. Throughout its execution, the VM produces an execution trace, recording (i) the current program counter value and (ii) a commitment to the state of memory at each step. The BitVM protocol leverages this execution trace for dispute resolution, as described in the full version of the paper [5, Appendix C.3, C.4].

Formally, let a *VM address* be an integer $addr \in \mathcal{A} := \{0, 1, \ldots, \mathsf{MemLen} - 1\}$ where $\mathsf{MemLen} \in \mathbb{N}_{>0}$ represents the memory length. We define the VM *memory* as the sequence $M \in \mathcal{M} := \{0, 1, \ldots, n\}^{\mathsf{MemLen}}$, where $n \in \mathbb{N}_{>0}$ specifies the range of values stored at any memory address. The *VM program counter*, denoted pc, is an element of the set $\mathcal{PC} := \{0, 1, \ldots, \ell - 1\} \cup \{\bot\}$, where $\ell \in \{1, 2, \ldots, n\}$ is the maximum length of the program, and $\bot$ indicates termination. Let $\mathcal{OP} := \{f_{\mathsf{OP}} : \mathcal{PC} \times \{0, \ldots, n\} \times \{0, \ldots, n\} \to \mathcal{PC} \times \{0, \ldots, n\} \cup \{\bot\}\}$ be a set of CPU instructions that the VM can execute[10]. The function f_{OP} takes as input a triple (pc, val_A, val_B) and outputs a pair (pc, val_C). For any CPU instruction $f_{\mathsf{OP}} \in \mathcal{OP}$, we require that f_{OP} is executable in Bitcoin Script. A *VM program* is an ordered sequence of ℓ elements, denoted $\Pi \in \mathcal{I}^\ell$, where $\mathcal{I} := \{(f_{\mathsf{OP}}, addr_A, addr_B, addr_C) \mid addr_A, addr_B, addr_C \in \mathcal{A}, f_{\mathsf{OP}} \in \mathcal{OP}\}$. We can now define the following.

▶ **Definition 5** (VM State). *A* VM state*, or simply,* state*, is a triple* $S := (M, pc, \Pi)$*, where* M *is the VM memory,* pc *is the VM program counter, and* Π *is a VM program.*

▶ **Definition 6** (State Transition Function). *Let* $\mathcal{S} := \mathcal{M} \times \mathcal{PC} \times \mathcal{I}^\ell$ *be the set of all VM states. We define the* state transition function $f_{ST} : \mathcal{S} \to \mathcal{S}$ *with* f_{ST} *taking as argument the state* (M_i, pc_i, Π) *and giving as output the state* (M_{i+1}, pc_{i+1}, Π) *as specified in Algorithm 5.*

Given a program Π and a memory configuration M, we assume that the entry point of the program, namely the first instruction that a program executes, is always $\Pi[0]$. Thus, we define as *initial* state the tuple $S_0 := (M, 0, \Pi)$. We use the shorthand notation $f^i_{ST}(S)$ when we apply the state transition function f_{ST} to a state S exactly i times, $f^i_{ST}(S) := f_{ST}(f_{ST}(\ldots(f_{ST}(S))))$. We say that a state $S_i := (M_i, pc_i, \Pi)$ at step i is *correct* with respect to an initial state S_0 iff $S_i = f^i_{ST}(S_0)$. We avoid the subscripts (and refer to S_i as (M, pc, Π)) when it is clear from the context which state we are referring to. Finally, after a number of execution steps equal to final (final is decided when a VM instance is created), the program terminates. We denote the final state, or outcome, as $\Pi(S_0) := f^{\mathsf{final}}_{ST}(S_0)$.

[10] Even though $\mathcal{OP}$ can be arbitrary, we are interested in a Turing-complete instruction set. In particular, we later use ADD, BEQ, and JMP, cf. Algorithm 7 – a well-known Turing-complete instruction set [21].

Algorithm 5 State Transition Function f_{ST}.

```
1: function f_ST(M, pc, Π)
2:     if pc = ⊥ then
3:         return (M, pc, Π).
4:     M' ← M;
5:     (f_OP, addr_A, addr_B, addr_C) ← Π[pc];
6:     val_A ← M[addr_A];
7:     val_B ← M[addr_B];
8:     (pc', val_C) ← f_OP(pc, val_A, val_B);
9:     if val_C ≠ ⊥ then
10:        M'[addr_C] ← val_C;
11:    return (M', pc', Π).
```

We define a VM instance as a tuple $\Gamma := \langle \Pi, \mathsf{MemLen}, n, \mathsf{final} \rangle$. We write Γ^A to refer to the VM instance executed by party A. We write S_i^A to denote a VM state S_i that A claims to have produced during the execution of A's VM instance Γ^A. We say that two parties A and B agree on the state S_i if $S_i^A = S_i^B$, and disagree on S_i otherwise.

▶ **Definition 7** (Execution Trace Element)**.** *Let* $(M_i, pc_i, \Pi) := f_{ST}^i(S_0)$*, and let* MR_i *be the root of the Merkle tree with the entries of* M_i *as its leaves. The* i*-th VM* execution trace element*, or simply,* i*-th* trace element *is the pair* $E_i := (MR_i, pc_i)$*, for* $i \in \{0, \ldots, \mathsf{final}\}$*.*

We write E_i^A to denote a VM execution trace element E_i that A claims to have produced during the execution of A's VM instance Γ^A. The VM *execution trace* is defined as a sequence of consecutive trace elements $ExecTrace := (E_0, \ldots, E_{\mathsf{final}})$. We write $ExecTrace^A$ as a short-hand for $(E_0^A, \ldots, E_{\mathsf{final}}^A)$.

We describe how the VM behaves in Algorithm 6: starting from initial state S_0, it applies the state transition function f_{ST} to the state and records the related trace elements until the program Π ends, namely, once pc is set to be $\perp$. The VM algorithm is parameterized by final, a parameter that represents the maximum number of state transitions that the VM is allowed to perform. The VM algorithm returns as output the VM execution trace $ExecTrace$, along with the resulting memory M after the program execution.

Algorithm 6 The VM algorithm. S_0 is the initial VM state.

```
1: function VM_final(S_0)
2:     stepCount ← 0;
3:     while stepCount < final do
4:         E_stepCount ← (MR, pc);
5:         (M, pc, Π) ← f_ST(M, pc, Π);
6:         increment stepCount by 1;
7:     E_stepCount ← (MR, pc);
8:     ExecTrace ← (E_0, ..., E_final);
9:     return (ExecTrace, M).
```

A practical VM instance. For better readability and to provide a protocol instance that can be deployed in practice, in the rest of the paper, we will consider a VM instance $\Gamma := \langle \Pi, \mathsf{MemLen}, n, \mathsf{final} \rangle$ with the following initialization: We set the length of the memory

as $\mathsf{MemLen} = 2^{32}$ and the greatest integer that can be stored in any entry of the memory as $n = 2^{32} - 1$. Furthermore, we assume that the input program Π has $\ell \leq 2^{32}$ number of instructions[11] and we set $\mathsf{final} = 2^{32}$.

As for the set $\mathcal{OP}$ of instructions that the VM can execute, our VM instance employs the following: $\mathcal{OP} := \{\mathsf{ADD}, \mathsf{BEQ}, \mathsf{JMP}\}$. This is a minimal set of computer instructions known to be Turing complete [21]. We underscore that the BitVM protocol can function with any Turing-complete instruction set, provided that each instruction within the set is implementable in Bitcoin script. In Algorithm 7, we give an implementation of ADD, BEQ and JMP that can be easily translated in Bitcoin script.

Algorithm 7 The Algorithms ADD, BEQ, and JMP, each taking as input the tuple (pc, val_A, val_B), and returning a pair (pc, val_C).

```
1: function ADD(pc, val_A, val_B)
2:     if pc = ⊥ then return (⊥, ⊥);
3:     return (pc + 1, val_A + val_B).

4: function BEQ(pc, val_A, val_B)
5:     if pc = ⊥ then return (⊥, ⊥);
6:     if val_A = val_B then return (pc + 1, ⊥);
7:     else return (pc + 2, ⊥);

8: function JMP(pc, val_A, val_B)
9:     if pc = ⊥ then return (⊥, ⊥);
10:    return (val_A, ⊥).
```

5 The BitVM Protocol

The BitVM protocol enhances Bitcoin's expressiveness by enabling spending conditions based on the result of general-purpose computation – performed off-chain, but verifiable on-chain through an interactive protocol. While Bitcoin Script is not (quasi-)Turing complete, BitVM effectively simulates such computation by using cryptographic commitments, economic incentives, and Script-compatible fraud proofs.

BitVM enables two mutually distrusting parties, the prover (P) and the verifier (V), to agree on the output of a program Π run on input S_0. If both parties agree on the outcome, funds are distributed accordingly with minimal on-chain interaction. In case of disagreement, BitVM executes a dispute resolution protocol that isolates the exact point of divergence and verifies correctness using Bitcoin Script. All necessary transaction logic is expressible within Bitcoin, utilizing Lamport signatures and other Bitcoin Script features, thus no consensus changes are needed.

In the following, we provide a high-level description of the protocol's logic, and conclude with a discussion on its security guarantees and on-chain performance. The full protocol specification and concrete transaction constructions are deferred to the full version of the paper [5, Appendix C].

[11] In the BitVM protocol, we build a Taproot tree where every program instruction is a Tapleaf script. We chose such ℓ since $2^{32} << 2^{128}$, the maximum number of leaf scripts in the current specification of Bitcoin [41].

Protocol Phases. The BitVM protocol proceeds in four phases:

1. **Setup:** P and V agree on:
   - A program Π written for the BitVM virtual machine.
   - An outcome mapping function f, defining fund redistribution based on program output.
   - A time parameter Δ, used to timelock transactions to deter inactivity,
   - and a maximum execution trace length $\mathsf{final} = 2^{32}$.

   They presign all necessary transactions, including outcomes and disputes, and publish a `Setup` transaction locking funds on-chain in a multisig with timeout clauses.
2. **Execution:** P communicates the initial input S_0 to V. Then, both parties compute $\Pi(S_0)$ off-chain, producing an execution trace $ExecTrace = (E_0, \ldots, E_{\mathsf{final}})$, where each E_i is an execution trace element as in Definition 7.
3. **Commitment:** P posts `CommitComputation`, committing to both S_0 and the claimed output $\Pi(S_0)$ using Lamport-based commitments. If V agrees, P finalizes with `Close`, spending funds according to f. Otherwise, a dispute begins.
4. **Dispute Resolution:** If V disagrees with the claimed output, V engages P in an on-chain bisection game over the execution trace to identify a disputed VM step. Once found, this VM step is executed on-chain using Bitcoin Script. If the result is different from what P claimed, their funds are forfeited.

Optimistic Case (Happy Path)

If both parties agree on the result of the off-chain computation, only three transactions are posted on-chain:

1. `Setup` – locks funds in a multisig.
2. `CommitComputation` – P commits to the input and the output of the program using Lamport signatures.
3. `Close` – P spends `CommitComputation`'s transaction output with a result consistent with f, redistributing the funds.

Dispute Resolution (Unhappy Path)

If V disagrees with P's committed result, the protocol enters the dispute phase. Let $ExecTrace := (E_0, \ldots, E_{\mathsf{final}})$ be the off-chain VM execution trace, where each element is defined as in Definition 7. Notably, each successive element in this execution trace results from applying a single VM instruction to the preceding element. Since P and V agree on S_0[12], any disagreement on the result implies a disagreement on some S_{i+1}, while agreeing on S_i. The bisection game locates such an index $\mathcal{N}$, enabling on-chain verification of a single VM step (we formalize the bisection game in the full version of the paper [5, Appendix D]). Figure 1 illustrates an overview of the dispute process.

Identify Disagreement. V initiates the dispute by publishing the `KickOff` transaction, starting a 32-round challenge-response game. In each round j, V commits to a bit b_{32-j} via $\texttt{TraceChallenge}_{\texttt{j}}$, directing the search left ($b_{32-j} = 0$) or right ($b_{32-j} = 1$).

P responds with $\texttt{TraceResponse}_{\texttt{j+1}}$, revealing the midpoint E_{n_r}, where $r = 32 - (j+1)$ and $n_r = \sum_{i=r}^{31} b_i \cdot 2^i + 2^r$. After 32 rounds, the final index is $\mathcal{N} = \sum_{i=0}^{31} b_i \cdot 2^i$, with $\mathcal{N}' = \mathcal{N} + 1$. This yields the first divergent pair: agreement on $S_{\mathcal{N}}$, disagreement on $S_{\mathcal{N}'}$.

[12] They agree by default, since P chooses the initial state S_0 and communicates it to V.

Instruction Commitment. P must now prove that $S_{\mathcal{N}'} = f_{ST}(S_{\mathcal{N}})$. To do so, they publish a `CommitInstruction` transaction, committing to:

- $pc_\theta = pc_{\mathcal{N}}$ and $pc_{\theta'} = pc_{\mathcal{N}'}$: the program counters of the states $S_{\mathcal{N}}$ and $S_{\mathcal{N}'}$, respectively;
- $insType_\theta \in \mathcal{OP} := \{\mathsf{ADD}, \mathsf{BEQ}, \mathsf{JMP}\}$: the instruction type at $\Pi[pc_\theta]$;
- $addrA_\theta, addrB_\theta, addrC_\theta$: the memory addresses referenced in $\Pi[pc_\theta]$;
- $valA_\theta, valB_\theta$: the memory values at addresses $addrA_\theta$, $addrB_\theta$ in $S_{\mathcal{N}}$;
- $valC_\theta$: the memory value at address $addrC_\theta$ in $S_{\mathcal{N}'}$, i.e., after executing $f_{ST}(S_{\mathcal{N}})$.

The Taproot script must enforce that: $(pc_{\theta'}, valC_\theta) = insType_\theta(pc_\theta, valA_\theta, valB_\theta)$. Next, V may challenge this step via several failure cases.

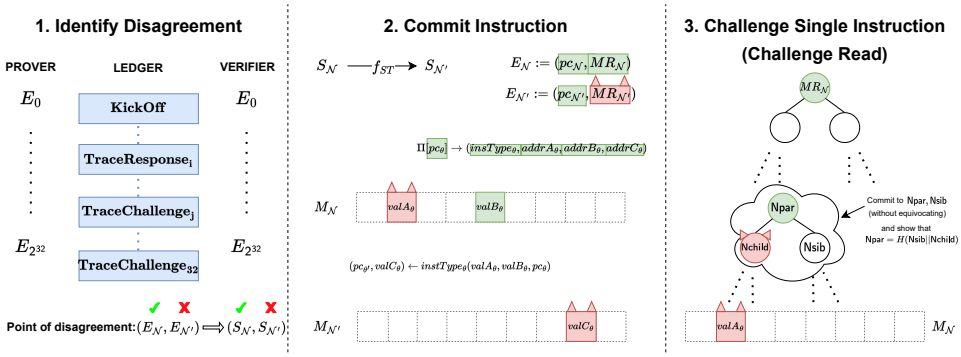


Figure 1 Example of dispute resolution in the BitVM protocol: To resolve a dispute, (1) P and V engage in a bisection game to identify the point of disagreement $(E_{\mathcal{N}}, E_{\mathcal{N}'})$ in their execution traces, indicating a disagreement in the transition from state $S_{\mathcal{N}}$ to $S_{\mathcal{N}'}$. Next, (2) P commits on-chain to all necessary information for executing $S_{\mathcal{N}'} = f_{ST}(S_{\mathcal{N}})$ (i.e., the values highlighted by colored boxes in the figure). In this example, we assume that P is committing to an incorrect value for $valA_\theta$, resulting in incorrect value for $valC_\theta$ and $MR_{\mathcal{N}'}$. (3) V challenges P through a bisection game over the path in the memory Merkle tree $M_{\mathcal{N}}$, from the root $MR_{\mathcal{N}}$ to the leaf containing $valA_\theta$. This bisection game reveals two intermediate nodes, Npar and Nchild, on which P and V disagree. To get away while using an incorrect value, P would need to prove that Nchild is indeed the left child of Npar, which is impossible without equivocating, leading to punishment.

Verifier Challenges. There are five challenge paths to verify the correctness of the transition $S_{\mathcal{N}} \to S_{\mathcal{N}'}$. In essence, either the values are not correctly derived from the agreed upon state $S_{\mathcal{N}}$, or the result is not stored correctly in $S_{\mathcal{N}'}$. By executing this step, V can pinpoint which of these errors occurred, and thus choose the appropriate challenge path.

(i)-(ii) Program Counter (Current/Next). V can dispute that $pc_\theta \neq pc_{\mathcal{N}}$ or $pc_{\theta'} \neq pc_{\mathcal{N}'}$ via `ChallengeCurrPC` or `ChallengeNextPC`. If proven, V claims the funds. If V equivocates on their bitwise commitment to $\mathcal{N}$, P can reveal equivocation via `PunishCurrPC`, claiming all funds instead.

(iii) Instruction Mismatch. Each program line (indexed by pc_θ) is encoded in a tapleaf with the expected $insType_\theta$, $addrA_\theta$, $addrB_\theta$, and $addrC_\theta$. If P commits to inconsistent values, V can reveal this via `DisproveProgram`, spending the corresponding tapleaf. If the program counter matches but any operand or opcode does not, V wins the dispute and claims the funds.

(iv) Read Error. To verify $valA_\theta = M_{\mathcal{N}}[addrA_\theta]$, V initiates a challenge by publishing `ChallengeRead`. A 5-round bisection game over the Merkle path $\mathcal{P}_R := (MR_{\mathcal{N}}, \ldots, M_{\mathcal{N}}[addrA_\theta])$, i.e. the path from the root $MR_{\mathcal{N}}$ to the leaf $M_{\mathcal{N}}[addrA_\theta]$, isolates a disagreement $(\mathsf{Npar}, \mathsf{Nchild})$. If P fails to justify it via a sibling hash Nsib (such that $H(\mathsf{Nsib}||\mathsf{Nchild}) = \mathsf{Npar}$) or equivocates on one of the elements of $\mathcal{P}_R$, V wins via `PunishRead`. Otherwise, P claims the funds after a timeout. The same logic applies to $valB_\theta$ at address $addrB_\theta$.

(v) Write Error. V challenges $valC_\theta \neq M_{\mathcal{N}'}[addrC_\theta]$ by publishing `ChallengeWrite`. The bisection game now runs on two Merkle paths (before and after the write). Again, V identifies a disagreement $(\mathsf{Npar}, \mathsf{Nchild})$ and challenges P to provide a consistent sibling node. If P cannot do so, V wins via `PunishWrite`; otherwise, P claims the funds after a timeout.

6 Security Analysis

We now show that BitVM is an *on-chain state verification protocol* that satisfies two key properties: *Balance Security*, which ensures that honest users never lose funds even if their counterparty behaves arbitrarily; and *Rational Correctness*, which guarantees that rational participants always follow the intended optimistic execution path. Our analysis models BitVM as an *Extensive Form Game (EFG)* (we define it in the full version of the paper [5, Appendix E]), which enables unified reasoning about both Byzantine and rational adversaries. This approach builds on recent work on incentive-compatible Layer-2 protocols [37], and provides a natural way to establish equilibrium guarantees, which are essential in financial settings.[13]

▶ **Theorem 8.** BitVM *is an on-chain state verification protocol that achieves* balance security *and* rational correctness.

Balance Security

We consider two cases: (i) both parties behave honestly, and (ii) one party $A \in \{P, V\}$ deviates at any step. In both scenarios, we prove that the honest party does not lose their funds. We note that if either party deviates during *Setup*, the honest party will refuse to sign the `Setup` transaction, ensuring no coins are locked unless both parties have received all necessary pre-signed transactions ([5, Lemma F.1]). Thus, we assume that the setup phase has concluded successfully.

Honest parties. When both parties are honest, BitVM follows an optimistic path: the prover posts the correct computation result on-chain via `CommitComputation`, and after the timelock expires, publishes `Close` to distribute the funds according to f ([5, Lemma F.3]).

[13] While Universal Composability (UC) is well-suited for modeling arbitrary adversaries, it does not support equilibrium reasoning. Formal tools targeting rational security exist for simple constructions (e.g., Lightning's closing game [16]), but do not scale to BitVM 's combinatorial structure (e.g., bisection-based disputes). Developing general-purpose frameworks for rational security in expressive smart contract systems remains an open challenge.

***V* honest, *P* Byzantine.** If the prover fails to publish in time `CommitComputation`, either due to inactivity or incorrect computation, or subsequently fails to post `Close`, the verifier can reclaim all the funds locked in the multisignature after the respective timelocks expire ([5, Lemma F.2, F.3]). This mechanism prevents hostage scenarios by ensuring that the verifier can recover their coins in case of non-responsiveness.

If the prover commits to an incorrect result in `CommitComputation`, the verifier initiates the Identify Disagreement phase by publishing `KickOff`. If the prover remains inactive, the verifier can claim the coins after the timelock expires ([5, Lemma F.4]). If the phase completes, the verifier obtains a VM step for which the prover has incorrectly committed to the outcome of the state transition function (Algorithm 5) ([5, Lemma F.7]).

The prover may have deviated by using an invalid program counter (current or next), performing an incorrect memory read or write, or executing an invalid instruction. For each case, the verifier can post the corresponding on-chain transaction – such as `ChallengeCurrPC`, `ChallengeNextPC`, `ChallengeRead`, `ChallengeWrite`, or `DisproveProgram` – to initiate the appropriate dispute path. This allows the verifier to disprove the prover's computation and claim the funds ([5, Lemma F.14]).

***P* honest, *V* Byzantine.** A malicious verifier may initiate the Dispute Phase by publishing `KickOff` on-chain, even though the prover has correctly committed to the result in `CommitComputation`. If the verifier becomes inactive during the Identify Disagreement phase, the prover can claim the funds once the timelock expires ([5, Lemma F.5]).

If the phase completes, the verifier must follow up with a challenge by posting one of the transactions `ChallengeCurrPC`, `ChallengeNextPC`, `ChallengeRead`, `ChallengeWrite`, or `DisproveProgram`. Since the prover's commitment is correct, the verifier cannot produce a valid inconsistency and ultimately fails to disprove the computation. In this case, the prover reclaims the funds ([5, Lemma F.15]).

Rational Correctness

We now establish that rational participants are incentivized to follow the optimistic execution path of BitVM. Specifically, in [5, Theorem D.16], we prove that the honest strategy profile constitutes a *Subgame Perfect Nash Equilibrium (SPNE)*. The proof proceeds by backward induction on the game tree: in every subgame, deviation results in strictly lower utility, since the honest counterparty can either recover funds via timeouts or successfully disprove incorrect behavior on-chain. These consequences are established through the same mechanisms formalized in [5, Lemma F.2, F.15].

In particular, if the prover deviates by omitting `CommitComputation`, failing to post `Close`, or committing an invalid transition, the verifier can either reclaim their funds or win the dispute. Conversely, if the verifier initiates an unwarranted dispute, they will be unable to disprove the prover and ultimately forfeit their claim. As any unilateral deviation leads to lower utility, both parties are incentivized to behave honestly. This makes BitVM incentive-compatible: rational players follow the optimistic path without invoking a dispute.

7 Implementation and Evaluation

To show the feasibility of our approach, we implement a prototype of BitVM in JavaScript [2]. In addition to showing how BitVM can be implemented practically, we use it to compute the transaction fees for both an optimistic run and the most expensive dispute branch of BitVM.

We assume constant transaction fees of $3sat/vB$[14], a Bitcoin price of $\$73,000$ (as of May 2026). To make the prototype more efficient, we realize the one-time signatures with Winternitz signatures [34] instead of Lamport signatures. Using Winternitz signatures, both the size of a signature and the size of a public key are around $5vB$ per message bit. As hash function, we use the Bitcoin Script primitive OP_HASH160 [35]. We also assume that $\Delta = 12$ hours, meaning each timelock expires after half a day. Different concrete values can be chosen, but any such selection would require scaling the time evaluation accordingly.

Optimistic case. In the optimistic case, three on-chain transactions are published: `Setup`, `CommitComputation`, and `Close`, totaling $1,944vB$[15]. The protocol's execution cost is $5,832$ sat ($4.26). In terms of execution time, once `Setup` is published, BitVM completes in at most 2Δ time, corresponding to 1 day.

Dispute case. We focus on the most expensive path in terms of fees, the *Write Error* path. Overall, 81 transactions are posted on-chain; the path weighs $244,040vB$. The total protocol execution cost is $732ksat$, or about 534$, and, once the `Setup` transaction is published on-chain, it takes at most $80 \times \Delta = 40$ days to complete its execution. In case of a dispute, all the fees needed to run the protocol on-chain are covered by the misbehaving party[16].

These costs are acceptable for a market deployment, as witnessed by the massive investments of the industry in the BitVM paradigm [23, 38, 17].

8 Bridge Application

In this section, we leverage BitVM to instantiate a bridge application between the Bitcoin ledger and a sidechain system running a distributed ledger protocol, as defined in Definition 1, which satisfies *safety* and *liveness*. This bridge enables users to mint (wrapped) Bitcoin tokens on the sidechain and later redeem them back on the Bitcoin blockchain and is secure, assuming only existential honesty of the participants.

We first outline the bridge protocol, then present some high-level security arguments and conclude this section with an evaluation.

8.1 A BitVM-based Bridge Design

Consider two users, Alice and Bob: Alice mints tokens on the sidechain, transfers them to Bob, who then redeems the equivalent amount back to Bitcoin. The protocol assumes a committee of n members active during the BridgeSetup phase[17], with at least one assumed honest during execution. We illustrate the bridge protocol in Figure 2.

During the *BridgeSetup* phase, the committee members pre-sign the transactions required for the protocol execution. The core procedures are *PegIn* (Alice mints tokens) and *PegOut* (Bob redeems them). To enable `PegOut`, a committee member fronts coins to Bob and later

[14] In Bitcoin, the size of a SegWit [32] transaction is expressed in *virtual Bytes*, or *vBytes* (vB). The number of vBytes of a transaction witness is equal to its number of Bytes divided by four.
[15] Verifying an arbitrary program with BitVM is $5\times$ more expensive than a 2-input standard pay-to-public-key-hash transaction ($374vB$).
[16] Since we do not know beforehand if a party will misbehave, both P and V need to put enough collateral to cover for the fees. To keep low the opportunity cost of running BitVM, we aim to have a protocol with minimal on-chain footprint.
[17] Otherwise, the committee is reshuffled until BridgeSetup completes.

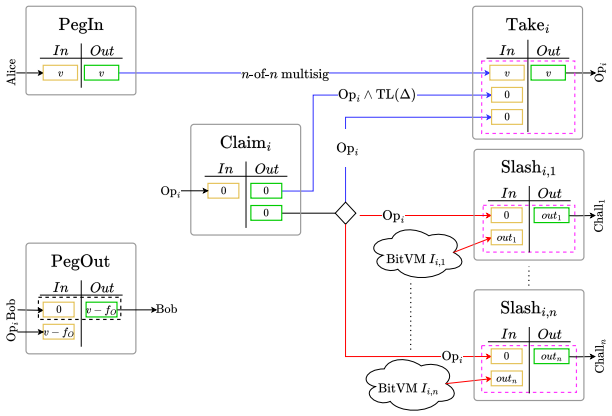


Figure 2 Illustration of BitVM-based bridge. For readability, we represent transactions as grey boxes; the value carried by transaction inputs and outputs is written inside a yellow and green rectangle, respectively. Pink dashed rectangles around the inputs and outputs of `Take` and `Slash` denote that the transaction is pre-signed during BridgeSetup, while the black dashed rectangle indicates which portion of the PegOut transaction Bob signed at PegOut time. Above the arrows, we denote the condition that unlocks the output from which the arrow starts. The arrows are blue if they represent a path taken by the operator, and red if the path is taken by one of the other committee members. The amount out_j, output of the BitVM instance I_i, j, depends on the funds still present after the BitVM dispute phase.

reclaims them in the *reimbursement* phase[18]. We refer to this member as the *operator*. For each operator, the remaining committee members act like *challengers*: they ensure that only honest operators can reclaim funds in the reimbursement phase by disproving any member falsely claiming to have fronted coins to Bob.

BridgeSetup. In that phase, the committee pre-signs, before the PegIn procedure is executed, specific transactions to reimburse an honest operator or punish a misbehaving one.

First, for each distinct pair of committee members denoted c_i and c_j, c_i pre-signs and forwards the transaction $\texttt{Slash}_{(\texttt{i},\texttt{j})}$ to c_j, which c_j can later publish to punish c_i upon misbehavior, as explained below. Second, for each c_i, all the committee members pre-sign the $\texttt{Take}_\texttt{i}$ transaction that refunds c_i, as described below. If a committee member stops collaborating during setup, it is kicked out and the procedure restarts.

PegIn. Alice deposits u coins on Bitcoin via a `PegIn` transaction. This transaction has a single output that deposits the u coins into the multi-signature wallet controlled by the n committee members. The sidechain verifies the inclusion of the `PegIn` transaction in the Bitcoin blockchain using a Bitcoin light client. Once confirmed, the sidechain mints u wrapped tokens to Alice's account.

PegOut. To withdraw the u coins, Bob first publishes a `Burn` transaction on the sidechain. Then, he constructs a `PegOut` transaction with a single output allocating $u - f_0$ coins to his Bitcoin address, where f_0 is a service fee. Bob broadcasts this `PegOut` request to the committee members. At least one committee member effectively fronts $u - f_0$ coins from their own funds to fulfill the withdrawal, in exchange for f_0.

[18] To coordinate among committee members, we can define an operator schedule, for example, in a round-robin fashion.

Reimbursement. To facilitate the reimbursement of the committee member that fronted its coins to Bob, we proceed with the following construction.

Claim. Each committee member c_i can claim that they fronted coins to Bob by posting the transaction $\texttt{Claim}_\texttt{i}$ on-chain. $\texttt{Claim}_\texttt{i}$ takes as input an empty transaction from the committee member c_i and has an empty output that can be spent after a timelock $\mathsf{TL}(\Delta)$ expires, along with a *connector* output, i.e., a transaction output given as input to different transactions to guarantee that only one of them will appear on-chain.

Slashing Mechanism. If c_i falsely claims to have fronted coins to Bob, any c_j can later publish on-chain $\texttt{Slash}_{(\texttt{i},\texttt{j})}$ to punish c_i. To achieve that, we employ a number of $O(n^2)$ BitVM instances as follows. For every pair of distinct committee members c_i and c_j, we consider the instance $I_{i,j}$ where c_i acts as the prover and c_j as the verifier.[19]

For a fixed i, the program Π, hardcoded into each instance $I_{i,j}$, verifies the following conditions: a) Bob's Burn transaction exists on the sidechain, and, b) some `PegOut` transaction, where operator c_i fronts the coins to Bob, exists on the Bitcoin ledger. For example, the program Π could encode a zk-SNARK verifier that verifies a proof of (1) and (2), making use of a light client of both the sidechain and the Bitcoin ledger, as shown in, e.g., [31, 10, 3].

The transaction $\texttt{Slash}_{(\texttt{i},\texttt{j})}$ takes as input: i) the connector output of $\texttt{Claim}_\texttt{i}$ and ii) an input from any transaction where c_j wins the instance $I_{i,j}$ (i.e., any transaction where all the coins of the multi-signature are attributed to c_j) and has a single output where all the remaining coins locked in the instance $I_{i,j}$ are given to c_j.

Take. The committee member c_i that fronted coins to Bob, can finally retrieve their funds by publishing transaction $\texttt{Take}_\texttt{i}$ on-chain. The transaction $\texttt{Take}_\texttt{i}$ has the following inputs: i) the output of PegIn, ii) the first output of $\texttt{Claim}_\texttt{i}$, and iii) the connector output of $\texttt{Claim}_\texttt{i}$, and has a single output where the coins of the first input are transferred to c_i.

For each operator c_i, all committee members presign $\texttt{Take}_\texttt{i}$ only if: (i) $\texttt{Claim}_\texttt{i}$ is constructed according to the protocol, and (ii) they have received all the pre-signed transactions related to the BitVM instance $I_{i,j}$.

8.2 Security Arguments

Since the sidechain ensures `PegIn` succeeds, our goal is to guarantee that a successful `PegIn` implies a successful `PegOut`: if Alice mints and transfers coins to Bob, Bob can eventually reclaim the equivalent amount on Bitcoin. This relies on two properties:

Bridge Safety (no false claims): A committee member cannot falsely claim to have fronted coins to Bob. Otherwise, funds in the multisig are stolen, and future honest operators are unable to reclaim their coins.

Bridge Liveness (honest redemption): Eventually, an honest operator will front coins and reclaim them.

Safety holds because if a malicious member c_i posts $\texttt{Claim}_\texttt{i}$ without first posting `PegOut`, an honest challenger c_j can initiate BitVM instance $I_{i,j}$ and post $\texttt{Slash}_{\texttt{i},\texttt{j}}$, which spends the connector output of $\texttt{Claim}_\texttt{i}$ and invalidates $\texttt{Take}_\texttt{i}$. Liveness holds because an honest member c_i who posts `PegOut` is protected: no malicious c_j can succeed with $\texttt{Slash}_{\texttt{i},\texttt{j}}$, and after the $\texttt{Claim}_\texttt{i}$ timelock, c_i can post $\texttt{Take}_\texttt{i}$ to reclaim their funds.

[19] The only difference from the current protocol is that, in this case, the verifier commits the `BridgeSetup` on-chain to initiate a dispute.

8.3 Evaluation

Table 3 summarizes the transaction costs associated with each phase of the bridge protocol (to see how they are computed, refer to the full version [5, Appendix G]). Specifically, in the optimistic scenario, where all committee members behave honestly, executing an instance of our bridge protocol requires only four on-chain transactions – `PegIn`, `PegOut`, `Claim`, and `Take` – resulting in a minimal total cost of approximately $2.04, which can be covered by the application fee f_0 paid by Bob.

On the contrary, if a challenger opens up a dispute, we have the following cases:

1. if the operator was honest, they only pay fees for `PegIn`, `PegOut`, `Claim`, and `Take` (approximately 2.04$, covered by Bob's fee f_0). The malicious challenger pays transaction fees associated with the execution of a BitVM instance (i.e., 534$);
2. if the operator was illegitimately trying to reclaim coins by publishing `Claim`, they will be slashed by the challenger and will pay the cost of a BitVM instance and the fees of `Slash` and `Claim`. This amounts to 534$.

Table 3 Evaluation of a BitVM-based bridge instance w.r.t. transaction sizes and on-chain costs.

Tx	Size (vB)	On-chain cost ($)
`PegIn`	117	0.26
`PegOut`	180	0.39
`Claim`	160	0.35
`Slash`	212	0.46
`Take`	475	1.04

9 Conclusion and Future Work

This work presents BitVM, the first protocol to enable general-purpose, trustless computation on Bitcoin without requiring consensus changes, or additional assumptions, e.g., trusted hardware or semi-trusted oracles. By combining off-chain execution with a Bitcoin-compatible dispute resolution protocol, BitVM achieves quasi-Turing completeness using existing scripting capabilities. To demonstrate the applicability of BitVM, we construct a trust-minimizing bridge and provide a prototype implementation with a concrete cost analysis.

BitVM extends the design space for Bitcoin-based applications, enabling programmable logic and verifiable off-chain computation in a trustless setting. Its ability to condition Bitcoin transactions on arbitrary program outputs opens new avenues for decentralized infrastructure anchored in Bitcoin's security model.

Several directions remain for future work. First, while BitVM currently supports two-party interactions, generalizing the protocol to support multiparty or permissionless settings – such as decentralized oracle networks or bridges – is an important next step. Second, further reducing on-chain cost through improved dispute resolution mechanisms, such as more efficient encodings or batched verifications, could improve scalability. Third, building higher-level tooling, including compilers or domain-specific languages, would help lower the barrier to adoption and enable broader experimentation with complex BitVM-based applications.

We believe BitVM marks a foundational step in unlocking the next generation of Bitcoin-native applications.

References

1 tbtc. URL: https://tbtc.network.

2 Bitvm poc implementation. https://github.com/BitVM/bitvm-js, 2025.

3 Bob hybrid l2 technical blueprint. https://blog.gobob.xyz/posts/bob-hybrid-l2-technical-blueprint, 2025. Accessed: 2025-04.

4 What are trustless bitcoin vaults?, 2025. Accessed: 2025-11. URL: https://babylonlabs.io/learn/what-are-trustless-bitcoin-vaults.

5 Bitvm: Quasi-turing complete computation on bitcoin (full version). https://eprint.iacr.org/2024/1995.pdf, 2026.

6 Ioannis Alexopoulos, Zeta Avarikioti, Paul Gerhart, Matteo Maffei, and Dominique Schröder. BitPriv: A privacy-preserving protocol for DeFi applications on bitcoin. Cryptology ePrint Archive, Paper 2025/1575, 2025. URL: https://eprint.iacr.org/2025/1575.

7 Lukas Aumayr, Zeta Avarikioti, Matteo Maffei, Giulia Scaffino, and Dionysis Zindros. Blink: An optimal proof of proof-of-work. In *Financial Cryptography and Data Security*, 2025.

8 Lukas Aumayr, Oguzhan Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostáková, Matteo Maffei, Pedro Moreno-Sanchez, and Siavash Riahi. Generalized channels from limited blockchain scripts and adaptor signatures. In *ASIACRYPT*, 2021.

9 Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. Enabling blockchain innovations with pegged sidechains, 2014. URL: https://blockstream.com/sidechains.pdf.

10 Ekrem Bal, Lukas Aumayr, Atacan İyidoğan, Giulia Scaffino, Hakan Karakuş, Cengiz Eray Aslan, and Orfeas Stefanos Thyfronitis Litos. Clementine: A collateral-efficient, trust-minimized, and scalable bitcoin bridge. Cryptology ePrint Archive, Paper 2025/776, 2025. URL: https://eprint.iacr.org/2025/776.

11 Massimo Bartoletti, Tiziana Cimoli, and Roberto Zunino. Fun with bitcoin smart contracts. In *Leveraging Applications of Formal Methods, Verification and Validation. Industrial Practice*, pages 432–449, Cham, 2018. doi:10.1007/978-3-030-03427-6_32.

12 Massimo Bartoletti, Stefano Lande, and Roberto Zunino. Bitcoin covenants unchained. In *Leveraging Applications of Formal Methods, Verification and Validation: Applications*, pages 25–42, 2020. doi:10.1007/978-3-030-61467-6_3.

13 Massimo Bartoletti, Riccardo Marchesin, and Roberto Zunino. Secure compilation of rich smart contracts on poor UTXO blockchains. In *9th IEEE European Symposium on Security and Privacy, EuroS&P*, 2024. doi:10.1109/EUROSP60621.2024.00021.

14 Massimo Bartoletti and Roberto Zunino. Bitml: A calculus for bitcoin smart contracts. In *ACM CCS*, pages 83–100, 2018. doi:10.1145/3243734.3243795.

15 Dan Boneh and Victor Shoup. A graduate course in applied cryptography. *Draft 0.6*, 2023. URL: https://toc.cryptobook.us/.

16 Lea Salome Brugger, Laura Kovács, Anja Petkovic Komel, Sophie Rain, and Michael Rawson. Checkmate: automated game-theoretic security reasoning. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 1407–1421, 2023. doi:10.1145/3576915.3623183.

17 Citrea. Bitcoin settlement trust-minimized btc bridge: Bitvm, 2024. Accessed: 2025-11. URL: https://docs.citrea.xyz/technical-specs/characteristics/bitcoin-settlement-trust-minimized-btc-bridge/bitvm.

18 Poulami Das, Lisa Eckey, Tommaso Frassetto, David Gens, Kristina Hostáková, Patrick Jauernig, Sebastian Faust, and Ahmad-Reza Sadeghi. FastKitten: Practical smart contracts on bitcoin. In *USENIX Security 19*, 2019.

19 Contributor DLC. Discreet log contracts: An overview, 2018. URL: https://adiabat.github.io/dlc.pdf.

20 Stefan Dziembowski, Sebastian Faust, and Kristina Hostáková. General state channel networks. In *ACM CCS*, pages 949–966, 2018. doi:10.1145/3243734.3243856.

21 Esolangs. Addleq, 2025. URL: https://esolangs.org/wiki/Addleq.

22 Ethereum Foundation. State channels, 2024. Accessed: 2025-04. URL: https://ethereum.org/en/developers/docs/scaling/state-channels/.

23 FairGate Labs. Bitvmx & bitvm, 2024. Accessed: 2025-11. URL: https://fairgate.io.

24 Tommaso Frassetto, Patrick Jauernig, David Koisser, David Kretzler, Benjamin Schlosser, Sebastian Faust, and Ahmad-Reza Sadeghi. Pose: Practical off-chain smart contract execution. In *NDSS Symposium*. Internet Society, 2023. doi:10.14722/ndss.2023.23118.

25 Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. *J. ACM*, 71(4), August 2024. doi:10.1145/3653445.

26 Sanjam Garg, Dimitris Kolonelos, Mikhail Sergeevitch, Srivatsan Sridhar, and David Tse. BABE: Verifying proofs on bitcoin made 1000x cheaper. Cryptology ePrint Archive, Paper 2026/065, 2026. URL: https://eprint.iacr.org/2026/065.

27 Harry Kalodner, Steven Goldfeder, Xiaoqi Chen, S Matthew Weinberg, and Edward W Felten. Arbitrum: Scalable, private smart contracts. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1353–1370, 2018. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/kalodner.

28 Leslie Lamport. Constructing digital signatures from a one way function. Technical Report CSL-98, SRI International, October 1979. URL: https://www.microsoft.com/en-us/research/publication/constructing-digital-signatures-one-way-function/.

29 Robin Linus. Bitvm: Compute anything on bitcoin, 2023. Accessed: 2025-11. URL: https://bitvm.org/bitvm.pdf.

30 Robin Linus, Lukas Aumayr, Zeta Avarikioti, Matteo Maffei, Andrea Pelosi, Orfeas Thyfronitis Litos, Christos Stefo, David Tse, and Alexei Zamyatin. Bridging bitcoin to second layers via BitVM2. In *35th USENIX Security Symposium (USENIX Security 26)*, 2026.

31 Robin Linus, Lukas Aumayr, Alexei Zamyatin, Andrea Pelosi, Zeta Avarikioti, and Matteo Maffei. BitVM2: Bridging bitcoin to second layers, 2024. Accessed: 2025-04. URL: https://bitvm.org/bitvm_bridge.pdf.

32 Eric Lombrozo and Pieter Wuille. Bip 141, segregated witness, 2015. Accessed: 2025-04. URL: https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki.

33 Varun Madathil, Sri AravindaKrishnan Thyagarajan, Dimitrios Vasilopoulos, Lloyd Fournier, Giulio Malavolta, and Pedro Moreno-Sanchez. Cryptographic oracle-based conditional payments. In *NDSS Symposium*, 2023. doi:10.14722/ndss.2023.24024.

34 Ralph C Merkle. A certified digital signature. In *Conference on the Theory and Application of Cryptology*, pages 218–238. Springer, 1989. doi:10.1007/0-387-34805-0_21.

35 Flavius Plesu. Turing completeness in evm machines, 2023. Accessed: 2025-04. URL: https://www.flavius.io/media/turing-completeness-in-evm-machines.

36 Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016.

37 Sophie Rain, Georgia Avarikioti, Laura Kovács, and Matteo Maffei. Towards a game-theoretic security analysis of off-chain protocols. In *IEEE CSF*, pages 107–122, 2023. doi:10.1109/CSF57540.2023.00003.

38 Sovryn. Bitcoinos, 2024. Accessed: 2025-11. URL: https://sovryn.com/bitcoinos.

39 Apostolos Tzinas, Srivatsan Sridhar, and Dionysis Zindros. On-chain timestamps are accurate. Cryptology ePrint Archive, Paper 2023/1648, 2023. URL: https://eprint.iacr.org/2023/1648.

40 Robin Linus Woll, Ioannis Alexopoulos, Lukas Aumayr, Zeta Avarikioti, Matteo Maffei, and David Tse. BitVM3: Efficient bitcoin bridges via garbled circuits. In *ACM Conference on Computer and Communications Security (CCS)*, 2026.

41 Pieter Wuille, Jonas Nick, and Anthony Towns. Bip 0341, taproot: Segwit version 1 spending rules, January 2020. URL: https://en.bitcoin.it/wiki/BIP_0341.

Privacy-Preserving Inclusion Lists

Zhengwei Tong
Duke University, Durham, NC, USA

Saba Eskandarian
University of North Carolina at Chapel Hill, NC, USA

Kartik Nayak
Duke University, Durham, NC, USA

Abstract

Blockchains aim to provide open access and censorship resistance, but centralization of block production in blockchains like Ethereum undermines these goals. Inclusion List (IL) protocols mitigate this by requiring block proposers to include transactions selected by an IL committee to enforce the inclusion of transactions that appear to have been censored. However, protecting the confidentiality of individual committee members' contributions is essential to prevent retaliation and ensure robust censorship resistance.

We propose a lightweight, privacy-preserving inclusion list protocol that allows committees to collectively construct transaction lists while hiding individual contributions and ensuring plausible deniability. Our approach builds on multiparty computation (MPC) techniques to achieve strong privacy without relying on heavyweight cryptography or anonymous broadcast channels.

We implement two variants of our protocol design: an optimistic version providing malicious security with abort (latency ~ 4.0s) for speed, and a robust variant (latency ~ 124.7s) for guaranteed output delivery in the presence of a Byzantine threshold of $t < n/3$ malicious parties.

2012 ACM Subject Classification Security and privacy → Privacy-preserving protocols

Keywords and phrases Blockchain, Inclusion List, Privacy, Multiparty Computation, Censorship Resistance

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.2

Related Version *Full Version*: `https://eprint.iacr.org/2026/1582` [32]

Supplementary Material *Software (Source Code)*: `https://github.com/timetzw/Privacy-Preserving-Inclusion-List`, archived at `swh:1:dir:55cf3bee05b224f078f2c18b98fe29e5d40b6f55`

Funding This material is based upon work supported by the National Science Foundation under Award Nos. 2234408 and 2237814. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

1 Introduction

Blockchains provide the ideal functionality of a general-purpose computer that is available and accessible to anyone, has no owner or operator, and supports property rights for user-owned data. However, implementations differ from such an idealized world, and there exists forces of centralization due to which these goals are not achieved. For instance, many blockchain protocols use block proposers or leaders that are responsible for selecting transactions added to the blockchain; thus, at any time, they temporarily control the availability and accessibility of the blockchain. In fact, in blockchains such as Ethereum, block production is outsourced to entities called builders, and the top two builders are responsible for producing more than 90% of blocks [39, 38, 29]. Ironically, this is similar to the centralized credit card networks where 95% of the purchase volume is controlled by the top three networks (Visa, Mastercard, and American Express) in the United States [27].


© Zhengwei Tong, Saba Eskandarian, and Kartik Nayak;
licensed under Creative Commons License CC-BY 4.0



A key concern related to the accessibility of the blockchain is censorship: a valid transaction proposed by any party must be included in the chain without undue delay, and censorship is the act of refusing to include valid transactions on the chain. There are two key reasons for censorship: self-censorship due to regulation and financial incentives. In recent years, we have observed self-censorship in the form of OFAC sanctions [33]. Moreover, applications related to decentralized finance can create financial incentives for censoring [36, 37, 17]. However, for a system intended to be accessible to anyone and everyone, we need to ensure that transactions cannot be censored by potential centralizing entities. To this end, parties that uphold the ideal functionality and provide censorship resistance should not be at risk of exposure or harm for doing so.

In blockchains such as Ethereum, recent proposals for addressing censorship concerns have focused on building a separate sidecar protocol called an *inclusion list protocol* [31, 35]. These protocols produce an agreed-upon list of transactions that the block proposer is then required to include in its proposed block, or else the block is not considered valid [30].

A key approach to prevent censorship in the inclusion list protocol itself is to use a committee of *multiple proposers* (also called IL proposers) and to output transactions that are proposed by a threshold of the proposers. Having multiple parties involved in producing the inclusion list ensures greater decentralization, so it is harder for transactions to be left out if a threshold of the parties are honest. Consequently, the job of the honest parties in the committee is to propose transactions in the mempool that are being censored.

1.1 Privacy and inclusion lists

Unfortunately, the act of combating transaction censorship by serving on an inclusion committee comes with risks. A member of an inclusion list committee may run afoul of censoring entities by forcing inclusion of transactions that would otherwise be censored. Thus, we aim to protect the confidentiality of which committee members proposed which transactions for inclusion in the final inclusion list. This is akin to the principle of protecting confidentiality in voting systems.

Adding privacy to inclusion list protocols safeguards committee members against punishment for supporting censorship resistance, and parties can uphold the ideal properties of the protocol without the risk of exposure or retaliation [1]. The goal of this work is to design such a multi-proposer privacy-preserving inclusion list protocol that incorporates transactions proposed by a threshold of parties while also protecting the privacy of parties proposing these transactions.

1.2 Key challenges and desiderata

There are several key challenges to designing such a privacy-preserving inclusion list. A core challenge involves achieving the desired privacy property that de-links a transaction input by a party from the output of the committee, even if some parties deviate from the protocol and behave arbitrarily. Thus, a proposer obtains the necessary anonymity guarantees and it can plausibly deny having included any transaction. Crucially, these guarantees must also hold at the network level, i.e., we want to achieve these guarantees even against an adversary that is able to observe all network traffic.

The main performance goal for an inclusion list protocol is to ensure potentially censored transactions can be included on a blockchain quickly. The lower the latency of the inclusion list protocol, the sooner the censored transactions can be included in the main chain. Thus, if it takes time equivalent to k blockchain "slots" to produce one inclusion list, then our goal is to reduce this k; ideally $k = 1$ and we are able to include a new inclusion list in every time slot.

Prior private inclusion list proposals [25, 1, 2] have made use of linkable ring signatures [5, 21] used on top of an anonymous communication network [15, 9, 20, 10, 11]. Unfortunately, these proposals have not been fully specified, analyzed, or implemented, and their security relies on a sufficiently performant anonymous communication system, a problem where significant performance and anonymity tradeoffs are inherent [13, 14]. We compare the setting and security properties of our proposed scheme with prior proposals in Section 7.

1.3 Our contribution

We present a protocol that allows an inclusion list committee to produce an inclusion list collectively without revealing any individual member's contributions and while giving each member plausible deniability for introducing each transaction to the final inclusion list. Our protocol is agnostic with regard to committee selection processes, assuming only that committee members can communicate with each other, and it does not require a network-level anonymity mechanism.

Our protocol employs ideas from the literature on multiparty computation (MPC) [19, 3, 12, 23, 22] and private computation of heavy hitters [6] to allow committee members to develop a list of transactions that many members want to place on the inclusion list. Apart from the MPC protocol, which computes simple functions like private sums or products, the protocol only uses lightweight, standard cryptographic tools, e.g., hashing and signatures.

Conceptually, the protocol operates over the prefix tree of all possible transaction hashes, but it explores this tree lazily: at any point, the parties maintain only the active prefixes that have not yet been pruned. At each layer of the tree, they compute a private sum for each node that lies on a path to a transaction they want included in the final list. Nodes with sums that are too low are pruned from the search, allowing the process to run efficiently, and any leaves that have not been pruned by the end of the protocol constitute the transaction hashes that correspond to the chosen inclusion list. Computing all sums privately under an MPC ensures that it is never known which committee members "voted" for which transaction, even if their protocol messages are visible to attackers at both the application and network layers. Our full protocol augments this idea with additional techniques to ensure stronger plausible deniability, protect against malicious adversaries, and improve performance.

1.4 Implementation and evaluation

We implement two versions of our protocol. Both provide all necessary privacy properties, but they differ in output delivery guarantees. The first is an optimistic version that uses the Mascot protocol in MP-SPDZ [23, 22] and completes execution only when all of the committee members cooperate. When evaluated under parameters chosen consistent with EIP-7805 [30] and a 16-member IL committee, the protocol runs within 4.0 ± 0.1s. The second version is robust under a Byzantine threshold of $t < n/3$ committee members, but runs in 124.7s.

The optimistic protocol completes well within Ethereum's 12-second block timing constraints, while the slower robust variant requires several blocks to produce an inclusion list. This suggests the possibility of running both protocols in parallel, allowing for quick inclusion list results in the common case, and eventual results in the slower case. We present a detailed analysis in Section 6.

2 Background and Preliminaries

2.1 Notation

We use $x := f(y)$ to denote assignment to x of the value $f(y)$, and $x \xleftarrow{R} S$ to denote assignment to x of a value chosen uniformly at random from a set S. Depending on context, we use either subscript notation or array notation to access an item on a list, i.e., vals_i or $\mathsf{vals}[i]$. A function $\mathsf{negl}(x)$ is *negligible* if, for all $c > 0$, there exists an x_0 such that, for all $x > x_0$, $\mathsf{negl}(x) < \frac{1}{x^c}$.

2.2 Background: Inclusion Lists (IL)

Inclusion lists were introduced to address the censorship risks that emerged after Ethereum's transition to Proposer–Builder Separation (PBS). With PBS, proposers outsourced block construction to specialized builders through an auction. This market structure created a centralization pressure: when the builder market becomes dominated by a small number of powerful actors [39], those builders effectively have the power to selectively censor transactions.

After a series of efforts to understand and address the problem of transaction censorship [7, 24, 8, 28], Fork-Choice enforced Inclusion Lists (FOCIL)[31] emerged as a solution that addressed concerns about incentives in prior proposals. FOCIL improves on earlier approaches in two ways: first, by replacing a single inclusion-list proposer with a committee to reduce susceptibility to bribery; and second, by enabling inclusion-list enforcement within the current slot, eliminating the one-slot delay required in prior designs.

FOCIL inclusion list committees gossip their views of the transaction mempool to come to a consensus on a list of transactions may have been censored, and the resulting list of transactions must be included in the next block. This paper augments this process with stronger privacy protections for committee members. While addressing the pros and cons of this kind of inclusion list design in general are out of scope for this work, we briefly discuss inherent limitations of inclusion list designs in Section 7.

2.3 Multiparty computation (MPC)

Multiparty computation (MPC) protocols allow parties with private data to jointly compute a function without revealing their individual inputs to each other. Conceptually, this is like a trusted third party collecting secrets and returning the result, but in practice, the parties run a protocol that simulates this role. Inputs are secret-shared – split into pieces $[x]_1, \ldots, [x]_n$ such that $\sum_i [x]_i = x$ – and operations on these shares allow secure addition, multiplication, and thus any arithmetic circuit. At the end, parties combine their shares to reveal the output. We express our protocol by describing the function $\mathcal{F}_{\text{MPC}}$ that the trusted party would compute based on the parties' inputs; in our implementation, this function is realized by replacing it with an actual MPC protocol.

2.4 MPC threat models

MPC protocols exist for a variety of threat models. We consider a static, malicious adversary controlling up to t parties. Our security analysis distinguishes between two notions of security against malicious behavior.

- **Malicious security with abort:** The adversary can corrupt the internal state and messages of corrupted parties. We guarantee that the protocol either outputs the correct result of the functionality $\mathcal{F}_{\text{MPC}}$ or aborts. Malicious security with abort can be achieved with up to $n - 1$ of n parties being malicious.

- **Guaranteed output delivery:** This variant guarantees that, assuming at most $t < n/3$ Byzantine parties, eventual message delivery among honest parties, and the standard assumptions of the underlying MPC protocol, all honest parties eventually output the same valid result. In particular, corrupted parties may deviate arbitrarily from the protocol, but they cannot prevent honest parties from obtaining an output.

3 Problem Description

Our core problem involves a set of n parties $P_1, \ldots, P_n$. We assume that up to t of the n parties are Byzantine, and thus can behave arbitrarily; the remaining parties are honest. Each of the n parties can communicate with each other using an authenticated channel.

There is a global set of values representing transaction hashes, where each value is a κ-bit value chosen uniformly at random. Each party P_i has an input list L_i containing a subset of these values. Our goal is for the parties to privately compute an intersection of all parties' input lists, and output a list L consisting of values that appear in a threshold number of inputs.

3.1 Security Goals

Intuitively, the privacy goal is to ensure that an adversary cannot determine whether a specific party input a specific value based on the protocol transcript and the final outputs. More precisely, we want the following properties to hold for parameters y and t:

- **Correctness.** We use two correctness notions, depending on the output-delivery guarantee of the MPC backend.
  - **Optimistic correctness.** If a value v is included in the inputs of at least y honest parties and the execution completes without abort, then v must appear in the output except with negligible probability in λ.
  - **Robust Correctness.** If a value v is included in the inputs of at least y honest parties, then despite arbitrary behavior by the t Byzantine parties, v must appear in the output except with negligible probability in λ.
- **Soundness.** No value v will be included in the output L unless it is included in L_i for at least one honest party.
- **Anonymity.** Parties to the protocol learn at most the count of how many parties have included a value v in their list L_i, with no information leaked about which parties contributed to the count.
- **Plausible deniability.** For any adversary corrupting up to t parties, any honest parti P_i and any value v, the adversary cannot rule out an execution possibility consistent with its view in which P_i did not contribute v to the protocol.

Anonymity and plausible deniability are related but distinct. Anonymity hides which parties contributed to a revealed aggregate count. However, if the protocol revealed that all committee members contributed to a value v, anonymity alone would not let any party deny contributing v. Deniability addresses this edge case by requiring that, for any given committee member, the transcript and output do not rule out the possibility that the member did not contribute v. We include the plausible deniability definition to augment anonymity, and our proposed protocol satisfies both requirements.

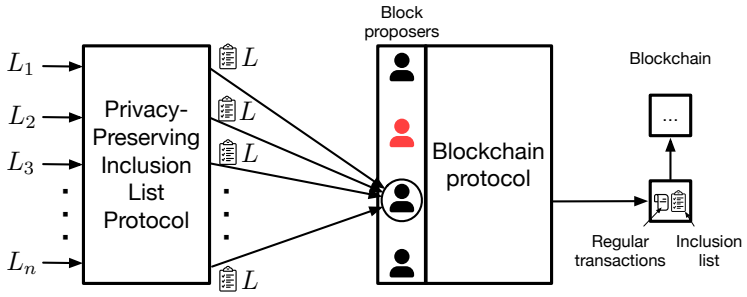


Figure 1 Privacy-Preserving Inclusion List protocol, the protocol should produce a singular version of the inclusion list to the next proposer in the ideal case.

3.2 Design Considerations

We now describe the larger context in which solving the above problem is relevant (cf. Figure 1). As described in the Introduction, in blockchains such as Ethereum, there are designated proposers for each *slot* of the protocol. Each slot can hold a block, and as of today, more than 90% of these blocks are built by two external entities called builders [39, 38, 29]. Such centralization makes it easy for transactions to be censored, either due to economic reasons or due to coercion [33, 36, 37, 17]. The notion of an inclusion list is meant to address this problem by relying on a decentralized set of blockchain validators to agree on a set of censored transactions, and the protocol requires the block proposer to include the agreed-upon set in the next slot.

We rely on the existence of a committee of validators whose members hold responsibility for producing the inclusion list for each slot. The membership of the committee is public, and there must be an agreement around who is on the committee. While the mechanism for constructing the committee is outside the scope of this work, our proposed solution is compatible with any committee that satisfies these requirements. However, for decentralization purposes, it will be beneficial if the committee is chosen at random afresh for each slot.

The committee members form the n parties we referred to in the problem statement. We assume each of these parties must have access to the mempool over time in order to determine which transactions should go on an inclusion list. This is the reason why we assume each of the parties' inputs comes from a global set of values. We note that no individual party may know the exact global set. While there will naturally be a large degree of overlap between committee members' views of the mempool, we do not require them to have identical views.

3.2.1 Protocol inputs and outputs

The committee members provide *transaction hashes* as input, and at the end of the protocol, each member outputs the agreed-upon inclusion list of transaction hashes. Committee members sign the list of hashes and share them with proposers on the network. A list signed by more than t committee members (and thus, at least one honest committee member) is considered a valid inclusion list for the slot. In order to produce a valid block, proposers must show that for each transaction hash h on the inclusion list, they have included a transaction tx such that $H(\mathsf{tx}) = h$ for the hash function H used by our scheme. Note that signing the outputs of the protocol is not connected to each committee member's individual inputs, so signatures on the output of the protocol do not reveal which, if any, of the included transactions were added by any individual committee member.

The hash function H can be any collision-resistant hash function of appropriate output length. Since the performance of our core protocol will depend on the output size of the hash function used, we can optimize performance by setting the output length to be shorter than is typically used for CRHFs, e.g., 64 bits, without affecting the privacy properties of the scheme. This introduces the possibility of occasional collisions where two different transactions hash to an entry on the inclusion list. We consider a block with either transaction to be valid, with the understanding that if the "wrong" transaction is included, the protocol execution on the next slot will still include that hash on the inclusion list, and the "correct" transaction can be included the next time around. In Section 6, we explore parameter tradeoffs like this one and we introduce optimizations in Section 5.4 that replace a single hash with multiple shorter hashes in order to improve parallelism in protocol executions.

3.2.2 Security properties in the larger context

We make a few observations relating to the properties described in the problem statement and the larger context.

- First, our core protocol outputs transaction hashes rather than full transaction payloads. This keeps the MPC lightweight, but it creates an additional availability requirement: a block proposer who receives a hash h must still obtain some transaction tx such that $H(tx) = h$. Soundness ensures only that an included hash was supported by at least one honest committee member, and therefore that a matching transaction exists in at least one honest party's view of the mempool. Looking ahead, our protocol will test that any transaction output has been included by at least some threshold $\tau \geq t + 1$ number of parties. We note that, if payload availability was ensured through other means or it was not needed, we could avoid the constraint on τ. See discussions in the full version [32].
- Second, our correctness property guarantees that a transaction input by at least y honest parties must appear in the output. Given censorship is a motivation, our goal will be to minimize the value of y.
- Finally, our requirements of anonymity and plausible deniability ensures that parties can input censored transactions while remaining unlinkable to the transactions that they input individually (anonymity) and each being able to deny that they input any particular transaction (plausible deniability).

3.3 A Two-Track Approach

Our core technical contribution is a multiparty protocol for voting on and producing a list of transaction hashes that constitute the contents of an inclusion list. The protocol, described in Section 4, consists of a recursive voting/narrowing procedure over the space of all possible hashes that identifies which hashes are included in the individual inclusion lists of τ or more committee members, without revealing which members have which transactions on their lists.

A large part of our core protocol involves running a series of small MPC subprotocols. The underlying MPC protocols used must provide malicious security and robustness due to the possibility of malicious parties on the committee tampering with the protocol or launching a denial of service attack by going offline. Unfortunately, known robust MPC protocols are more expensive than their counterparts that only provide malicious security with abort. The latter provides the guarantee that if an honest party gets an output, then it can check that it has indeed received the correct output, but it leaves room for malicious parties to interrupt the protocol by simply aborting [23, 12].

We observe that since malicious security with abort suffices for our soundness and privacy requirements, we can adopt a two-track approach. We run two versions of the protocol in parallel. The first version is a faster MPC protocol that only provides security with abort, i.e., satisfies optimistic correctness, and outputs a value within a deterministic time bound under a synchronous network. The other version provides correctness even in the face of arbitrary behavior by the t malicious parties. If the first version runs successfully, the output of that version will be considered for the slot; otherwise, we consider the output of the robust version, which will become available for a future slot.

4 Protocol Description

This section describes the core technical contribution of this paper: a privacy-preserving inclusion list protocol that satisfies the design goals and security requirements set forth in Section 3.

Prior approaches to anonymous inclusion lists rely on ring signatures and assume an underlying anonymous broadcast protocol, separating out application and network level privacy concerns [25, 1, 2]. We observe that by collapsing these two levels of the protocol stack, we can construct solutions that simultaneously use more lightweight cryptographic primitives and provide tailor-made privacy guarantees, reducing the demands placed on the underlying communication protocol. Our protocol will only require standard encrypted/authenticated channels between parties.

We adapt ideas from multiparty computation [19, 3, 12, 23, 22] and anonymous communication [9] to produce lightweight privacy-preserving inclusion lists. Our high-level approach is inspired by private heavy-hitters protocols, which recover popular strings from many private client inputs without revealing any individual client's string [6]. In our setting, the "popular strings" are transaction hashes supported by many committee members. Since the protocol itself hides who contributed which transaction, we do not need a separate anonymity layer to hide who sends which message within the protocol.

We also wish to satisfy a deniability property: any committee member should be able to plausibly argue that they did not add a given transaction to the inclusion list. Our scheme leaks only the (approximate) number of members who included each transaction on the final inclusion list. This is because the protocol allows the committee, subject to Byzantine inputs, to enforce arbitrary thresholds on how many members need to include a transaction for it to be on the list.

We discuss our solution below, first assuming a fully semihonest setting where all parties follow the protocol but corrupt parties wish to learn information based on the protocol transcript. Next, we will sketch how to lift the security of our protocol to tolerate malicious behavior.

4.1 A Semihonest Secure Approach

We think of inclusion list construction not as a problem where a number of parties append transactions to a list, but rather as a problem of selecting which transactions from the list of all possible transactions should be included. Intuitively, this is the difference between a sushi restaurant waiter asking for each person's order versus a table filling out a sheet with their menu selections. In the former scenario, each order is directly tied to the customer who asked for it, whereas in the latter, the group as a whole is connected to all parts of the order. Loosely speaking, our protocol will allow a committee to produce the completed order sheet without leaking any one member's contribution.

L_i: list of input transaction hashes for party P_i
AP_l: Set of active prefixes at level l
b: branching factor, a power of 2, representing the number of children of one node to explore
κ: length of input transaction hashes
n: number of parties
τ: threshold
x: maximum permissible number of zero-input parties
$\mathsf{coin}_i \xleftarrow{R} \{0,1\}^\lambda$
$\mathsf{out}_{\mathsf{coin}} := \mathcal{F}_{\text{sum}}(\mathsf{coin}_1, \ldots, \mathsf{coin}_n)$
Let j denote the smallest index in $\mathsf{out}_{\mathsf{coin}}$ s.t. $n - x \leq \mathsf{out}_{\mathsf{coin}}[j] \leq n - (t+1)$
if $\mathsf{coin}_i[j] = 0$ **then** ▷ zero-input party
    $L_i = \{\}$ ▷ Sets input to be empty
$AP_0 := \{0, \ldots, b-1\}$
$d = \log_b(2^\kappa) - 1$
for level $l := 0, \ldots, d-1$ **do**
    $V_{i,l} := \{\}$ ▷ P_i's vote map for level l
    for prefix $p \in AP_l$ **do**
        if $\exists tx \in L_i$ with prefix p **then** $V_{i,l} := V_{i,l} \cup (p, 1)$
        else $V_{i,l} := V_{i,l} \cup (p, 0)$
    $V_{G,l} := \mathcal{F}_{\text{MPC}}(V_{0,l}, \ldots, V_{n-1,l})$
    $AP_{l+1} := \{\}$
    for entry $\in V_{G,l}$ with key p **do**
        if $(V_{G,l}[p] \geq \tau)$ **then** $AP_{l+1} := AP_{l+1} \cup \{(p||0), \ldots, (p||b-1)\}\}$
$IL := \{\}$
for entry $\in V_{G,d}$ with key p **do**
    if $(V_{G,d}[p] \geq \tau)$ **then** $IL := IL \cup \{(p||0), \ldots, (p||b-1)\}\}$
Output IL

Figure 2 Computing the inclusion list. Functionalities $\mathcal{F}_{\text{sum}}$ and $\mathcal{F}_{\text{MPC}}$ are specified in Figure 3.

4.2 Desired functionality

Formally, we logically map all possible transactions, whether or not the transaction really exists, into the exponentially large output space of a collision resistant hash function H. Each party's input to our protocol will be an exponentially large vector with 1s in indices corresponding to hashes of transactions. When an inclusion list committee member holds a list of transaction hashes $L' = (\mathsf{tx}_1, \ldots, \mathsf{tx}_m)$ that it wants in the list, it logically transforms this list into a bit vector L of length 2^κ where an entry i holds 1 if there exists $\mathsf{tx}_j \in L', j \in [m]$, such that $H(\mathsf{tx}_j) = i$. All other entries are set to 0. Importantly, this exponentially long vector is a logical construct and never concretely realized.

We will use a *private sum* protocol for n committee members to go from $L_1, \ldots, L_n$ to a single list $L = \Sigma_{i=1}^n L_i$. Entries of L that are greater than some threshold τ will be added to the final inclusion list. Importantly, the protocol will allow the committee members to efficiently recover only the non-zero entries of the sum vector L, thereby doing work at most linear in the size of the inclusion list and logarithmic in the output space of the hash function to recover the list, rather than the exponential work of a naïve approach.

4.3 Construction details

Our high-level approach is inspired by work on private computation of *heavy hitters* [6] in data held secret shared across a number of parties. Our protocol will allow committee members to recursively explore levels of a binary tree with 2^κ leaves, where each internal

$\mathcal{F}_{\text{MPC}}$: level-l MPC aggregation

1: **for** $i \in \{0, \ldots, n-1\}$ **do**
2: **for** each entry with key p in $V_{i,l}$ **do**
3: **if** $V_{i,l}[p] \cdot (1 - V_{i,l}[p]) \neq 0$ **then**
4: Abort/ignore input
5: **for** each entry with key p in $V_{\_,l}$ **do**
6: $V_{G,l}[p] := \sum_{i=0}^{n-1} V_{i,l}[p]$
7: Return $V_{G,l}$

$\mathcal{F}_{\text{sum}}$: coin-tossing

1: **for** $i \in \{0, \ldots, n-1\}$ **do**
2: **for** $j \in \{0, \ldots, \lambda - 1\}$ **do**
3: **if** $\mathsf{coin}_i[j] \cdot (1 - \mathsf{coin}_i[j]) \neq 0$ **then**
4: Abort/ignore input
5: Return $\sum_{i=0}^{n-1} \mathsf{coin}_i$

Figure 3 $\mathcal{F}_{\text{MPC}}$ is the level-l MPC aggregation algorithm, and $\mathcal{F}_{\text{sum}}$ is the coin-tossing algorithm. All sums are component-wise. A secure-with-abort MPC aborts on failed bit checks, whereas a robust MPC discards invalid inputs.

node consists of an indicator bit revealing whether it is on the path to a non-zero leaf, and the non-zero leaves make up the elements of the list L. Our exploration algorithm will follow only the paths that lead to non-zero entries in L, without revealing any individual member's contribution.

At each level of the tree, committee members compute the sum of their values for each tree node using a simple MPC protocol that only consists of additions. The sums are revealed without anyone learning which members contributed a zero or one for each node. Once the sums are public, the members assemble a list of those nodes whose counts exceed the committee's chosen threshold τ for inclusion in the final list. These are the only nodes that lie on paths which could possibly have a non-zero value in their leaves. The process is repeated for the children of each node whose count is above the threshold. At the κth level of recursion, the remaining nodes are in fact the non-zero entries of the list L, computed without revealing who contributed to each entry.

4.4 Protocol description

The resulting protocol is described in Figure 2; we present a more general protocol with branching factor b. For now, we ignore lines 8-12. The protocol runs for $\log_b 2^\kappa$ iterations (line 15). In the first iteration, the protocol considers b number of prefixes; thus representing a $\log_2 b$ prefix length of transaction hashes. In each iteration, we increase the prefix length by $\log_2 b$. A party P_i maintains a list of active prefixes AP_l to be considered for level l (line 13). Based on the active prefix AP_l, P_i creates a local list $V_{i,l}$ where each element of a list is a key-value pair with a prefix $p \in AP_l$ being the key and a boolean value indicating whether it has an input transaction $tx \in L_i$ with prefix p (lines 17-19). Thus, a value 1 indicates that the party is interested in exploring the path with prefix p based on its input L_i, and it will be explored if sufficiently many other parties also input 1 for the same prefix. To determine this, the parties engage in a multiparty computation protocol $\mathcal{F}_{\text{MPC}}$ (line 20) where party P_i inputs $V_{i,l}$ (described in Figure 3). A secure multiparty computation protocol ensures that the parties only learn the output of the function computed on the parties' inputs, and nothing else; thus, a party's interest in exploring a path based on its input is kept private. In a semi-honest version of this MPC protocol, $\mathcal{F}_{\text{MPC}}$ only adds input $V_{i,l}[p]$ for each prefix p to produce output $V_{G,l}$ (Figure 3 lines 5-6). Party P_i then processes the output $V_{G,l}$ to determine the set of prefixes p that appears $\geq \tau$ times; if a prefix p matches that criteria, we explore prefixes $(p||0), \ldots, (p||b-1)$ in the next iteration (line 22). At the end of the $\log_b 2^\kappa$-th iteration, $AP_{\log_b 2^\kappa - 1}$ stores the desired set of outputs.

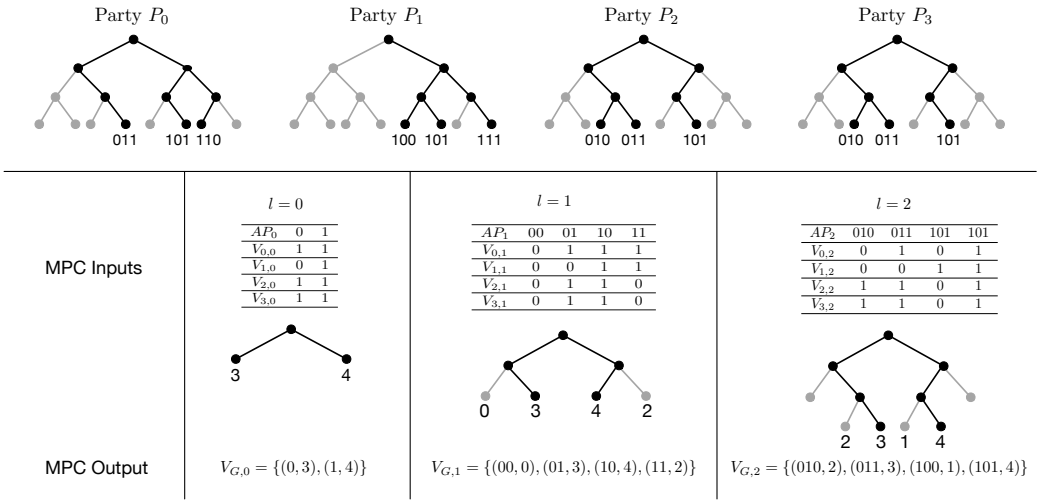


Figure 4 An example execution. For parameters $n = 4$ parties, $\kappa = 3$, $|L_i| = 3$, $b = 2$, and $\tau = 3$. For ease of exposition, we assume all parties have coin values set to 1. At the top, we show the inputs of each party P_i – the black paths denote the paths to input transactions, and the remaining paths are greyed out. In the first iteration, there are only two active prefixes $AP_0 = \{0, 1\}$. Parties input a 1 to a path they are interested in exploring. All parties are interested in exploring both paths except party P_1, which does not want to explore path 0 since none of its inputs start with that prefix. At the end of the computation, parties output $V_{G,0}$ with 3 parties voting to explore prefix 0 and 4 parties voting to explore prefix 1. Since both these values exceed threshold τ, all four paths are explored in the next iteration $l = 1$. Similarly, at the end of this iteration, only two paths 01 and 10 stay active. Finally, at the end of the execution, the parties output 011 and 101 as the output transactions.

4.4.1 Parameter tuning for security and performance

The inclusion threshold τ should be set based on expectations of the number of Byzantine/-malicious parties against which the committee expects to protect itself. For example, if there is an expectation of t Byzantine parties, the threshold can be set to $\tau = t + 1$ so that the Byzantine parties cannot corrupt or disrupt the output of the protocol, e.g., by not providing data corresponding to the transaction.

At any level of recursion, the number of subprotocols in progress is upper-bounded by the total number of items that need to be put in the final inclusion list, within a factor of b. Thus, increasing the value of b increases the number of prefixes processed per level, but the number of levels reduces to $\log_b 2^\kappa$. Since κ affects only the correctness of the protocol and not its security, we can set κ smaller to speed up the protocol, balancing the probability of collisions where it is unclear which of two or more potential transactions are intended to be on the list. These collisions are not fatal to the protocol because if an unnecessary transaction is added, the correct one can be added in the next iteration of the inclusion list protocol. Figure 4 illustrates an execution of the semi-honest example of our protocol.

4.5 Ensuring Plausible Deniability

The scheme described in Section 4.1 satisfies our anonymity requirement (assuming parties do not deviate from the protocol), but it does not provide plausible deniability. If every member of the inclusion list committee includes a given transaction, revealing the sum eliminates all parties' plausible deniability. If there are n parties and we see that n of them included a transaction, it is immediately clear that all n parties included that transaction.

More precisely, at each level of our protocol, $\mathcal{F}_{\text{MPC}}$ reveals the frequency with which a given prefix appears (line 20), allowing us to perform a cleartext comparison with threshold τ (line 23). This allows an observer to discern whether everyone on the committee included a transaction with that prefix. Plausible deniability against an adversary who controls t committee members can be compromised by a sum as low as $n - t$ since the adversary knows the votes of the t parties it controls.

From a privacy standpoint, this problem can be easily addressed by performing the required comparison with τ within $\mathcal{F}_{\text{MPC}}$; however, computing a comparison inside an MPC incurs additional multiplication gates, increasing the communication, round, and time complexity of each level of the protocol. Instead, we add plausible deniability by adding in a lightweight, randomized mechanism whereby certain parties, chosen privately and at random, always contribute zeros to the counts. Since any party can claim to be among the ones who only contributed zeros, plausible deniability is retained.

As mentioned above, the plausible deniability issue only occurs if the revealed count for a prefix is $\geq n - t$ since $\leq t$ of the inputs are adversarial and the adversary may input 0; thus, if we can guarantee that the output is always $\leq n - (t + 1)$, then an honest party has plausible deniability. We achieve this by ensuring that at least $t + 1$ parties are secretly designated to input all 0s for all levels of the protocol; we will call them *zero-input parties*. Since t is the upper bound on the number of adversarial parties, there must be at least one honest zero-input party. Since the identity of the zero-input parties remains secret, any party can claim to have been one of them, maintaining plausible deniability.

4.5.1 Selecting zero-input parties

Our intuitive idea is for each party to privately toss a 0/1 coin at the beginning of the protocol. Any party whose coin comes up 0 acts as a zero-input party. Unfortunately, this intuition does not work exactly because there is a possibility of there being too few or too many zero-input parties selected, which would result in either plausible deniability being lost or the protocol failing to produce any outputs, respectively. To remedy this, the parties reuse our private sum functionality to ascertain the number of zero-input parties, and they retoss their coins if there are too many or too few parties selected. To avoid extra communication rounds, we have the parties toss and sum many coins in parallel and use the first successful result.

The precise logic for selecting zero-input parties appears in lines 8-12 of Figure 2. The minimum acceptable sum of coin tosses is set to $n - x$, where x is the maximum permissible number of zero-input parties, where any larger number of zero-input parties could prevent honest parties from being able to include transactions on the inclusion list. The parameter x provides a trade-off between plausible deniability and the number of honest parties that must input a transaction for it to be guaranteed in the output. We discuss how to determine the range of acceptable values of x in terms of other parameters in the full version [32].

4.6 Protecting Against Malicious Behavior

In this section, we discuss our protocol in the malicious setting where the adversarial parties may deviate arbitrarily from the prescribed protocol. Recall that our goal is to obtain correctness, soundness, and privacy for the honest parties in the presence of such an adversary.

4.6.1 Using MPC secure under malicious adversaries

First, we ensure that the underlying $\mathcal{F}_{\text{MPC}}$ functionality described in Figure 3 is implemented under an MPC protocol tolerating t *malicious* adversaries. As described in Section 3.3, our implementation instantiates the protocol with two types of malicious security: malicious security with abort and robust MPC. Running the robust and non-robust MPC protocols in parallel means that a block can either immediately have its inclusion list (the optimistic case) or that an inclusion list is guaranteed to become available for a future block (the pessimistic case).

4.6.2 Addressing adversarial inputs

Our protocol is made up of a number of invocations of a smaller MPC protocol. While malicious security for the MPC ensures that the MPC functionalities are computed correctly, a malicious party can still attempt to break security by manipulating its inputs to the MPC (sub-)protocols. Observe that a malicious adversary in an MPC protocol has input independence, i.e., it can input anything to the protocol. By doing so, the adversary can potentially attempt to disrupt the correctness or privacy of the computation. There are two key concerns:

1. **Incorrectness through malformed inputs.** A malicious party may input arbitrary values that interfere with the correct aggregation of votes, potentially causing transactions that should have been considered to be excluded. For example, an attacker could artificially increase the count for a particular node in the tree in order to include an extra transaction in the final list, or even worse, an attacker could send values that "cancel out" the counts sent by other committee members, thereby preventing transactions from making it onto the final inclusion list.
2. **Inconsistent inputs.** The adversary can send inconsistent values across different levels of the protocol. As an example, consider the case where a party votes 1 at level l for a given active prefix $AP_l[p]$, thereby claiming to have transactions matching that prefix. The same party may then later vote 0 at all children vertices of this prefix, which is inconsistent with its earlier input and violates the intended protocol semantics. This can potentially cause additional paths to be explored due to "fake" inputs while honest execution prunes branches that do not meet the voting threshold or vice versa. A malicious party that repeatedly votes "1" for all active prefixes can force the exploration of many unnecessary paths.

For the first case, we enforce input validity through a bit-checking mechanism. Each party is restricted to submitting only binary values $\{0, 1\}$ to indicate its interest in extending a prefix. If a party attempts to inject a non-binary value, the check detects the deviation and causes the computation to abort. Concretely, for each prefix p in party i's input $V_{i,l}$ at level l, we ensure $V_{i,l}[p] \cdot (1 - V_{i,l}[p]) = 0$ (Figure 3 (left), lines 1-4). We emphasize that this check is performed over secret-shared values, ensuring that no information about honest inputs is leaked. A similar check is also performed to check for coin values when determining zero-input parties (Figure 3 (right) lines 1-4).

For the second case, note that this behavior does not compromise security guarantees. Since at most t parties are malicious, each can contribute at most one vote per transaction. Thus, a spurious transaction proposed only by malicious parties cannot reach the inclusion threshold, and will never appear in the final list. The effect is limited to performance: In the worst case, the number of paths explored is bounded by $O(|L|) \times b$.

4.6.3 Security analysis

Our complete construction is described in Figures 2 and 3. We analyze the security of the construction and give the proof intuition for the following theorems in the full version [32]. We omit formal cryptographic proofs of the theorems, which would primarily involve invoking the security of the underlying MPC protocols.

▶ **Theorem 1.** *Assuming it is instantiated with an MPC that satisfies malicious security with abort and parameters are set such that* $n - x \geq \tau + t$, $x \geq t + 1$, *and* $y \geq \tau + x$, *our scheme satisfies optimistic correctness with all but negligible probability in the choice of parameter* λ.

▶ **Theorem 2.** *Assuming it is instantiated with an MPC that provides robustness and parameters are set such that* $n - x \geq \tau + t$, $x \geq t + 1$, *and* $y \geq \tau + x$, *our scheme satisfies correctness with all but negligible probability in the choice of parameter* λ.

▶ **Theorem 3.** *Assuming the malicious security with abort of the underlying MPC protocol, and that* $\tau > t$, *our inclusion list protocol satisfies soundness.*

▶ **Theorem 4.** *Assuming the malicious security with abort of the underlying MPC protocol, our inclusion list protocol satisfies anonymity.*

▶ **Theorem 5.** *Assuming the malicious security with abort of the underlying MPC protocol and that* $n - (t + 1) \geq 0$, *our inclusion list protocol satisfies plausible deniability.*

5 Implementation and Optimizations

We implemented our malicious-secure privacy-preserving inclusion list protocol using the MASCOT protocol [23] as implemented in the `MP-SPDZ` [22] MPC framework, and we use `HoneyBadgerMPC` [26] (referred to as `HB-MPC` in our evaluation) for the robust variant of the protocol. Our code for both `MP-SPDZ` and `HB-MPC` is publicly available [1].

The rest of this section describes performance-related design decisions and optimizations that improve performance over a straightforward implementation of the scheme as described in the preceding sections. Our implementation includes the first of these optimizations but omits the others, as most of them trade off some security in exchange for improved performance. We evaluated our protocol without these optimizations in order to show performance with the highest security level, but we present them here as they may provide favorable tradeoffs for deployments of this protocol.

5.1 Interacting with `mp-spdz`

The `mp-spdz` library incurs some overhead to setup a new protocol each time an MPC starts. This means that running a separate MPC protocol per level of our protocol would result in repeatedly incurring the setup cost. Instead, we run a single MPC protocol that takes additional inputs and returns outputs at each level. However, `mp-spdz` only allows inputs to MPC protocols from static input files or command-line input. Thus we use the python `pexpect` module to send and receive inputs to the MPC protocol on a simulated command line rather than using file I/O for inputs.

[1] `https://github.com/timetzw/Privacy-Preserving-Inclusion-List`

This implementation decision comes with a tradeoff, as the library needs to know how many inputs to take at compile time (python-like MPC code is compiled into a final protocol). This means the protocol will need to take in a number of inputs corresponding to the maximum expected inclusion list size at each level, discarding inputs that end up not being needed. We find that this cost is preferable to that of paying the protocol initialization cost at each level.

5.2 Adding concurrency on top of `HoneyBadgerMPC`

We find that HoneyBadgerMPC runs in a single-threaded mode by default, relying on batching operations rather than concurrency in order to improve performance. While we ran our evaluation in a setting with only one core available per party, we briefly describe how to modify our implementation in order to take advantage of parallelism without altering the protocol. To do this, we could implement our protocol to run in separate MPC processes, each running on a different core. Consider the case of exploring a tree with branching factor four. We can think of each of the four nodes in the first level of the protocol as themselves being the root of a smaller instance of the same protocol, and run each one as a separate process. Since the outcomes of separate paths down a tree are independent of each other, this has no impact on the computation or results of the protocol, except to allow different parts of the protocol to run in parallel on different cores. The same idea applies for any branching factor, so we can split the protocol into as many subproblems as we need to optimally use the computing resources available. Moreover, the branching factor at this first level need not match the branching factor used within each instance of the protocol, allowing for even greater flexibility.

5.3 Speculative execution of subprotocols

We can reduce the overall number of protocol rounds considerably via a speculative subprotocol execution optimization. Since checking that all the sharings consist of 0/1 values requires a round of communication to compute the necessary multiplication, each subprotocol now requires an additional communication round to complete this step. However, the output sums of each subprotocol can be computed before these checks are completed. As such, the next subprotocol can proceed assuming that the checks will work out. In the case that they do, the overall rounds of the protocol are reduced, and in the case that they do not, the additional speculative round will not reveal anything about honest users' inputs because of the security properties of the MPC. Thus our protocol will at all times have a pipeline of subprotocols in progress.

This optimization comes with a mild security tradeoff. Since subprotocols execute speculatively without waiting for all the bit checks of a previous level to complete, it is possible for a malicious party to temporarily inflate/deflate counts for a node in the tree by sending inputs that are not zero or one. While this will eventually be caught, in the meantime, the protocol may explore parts of the tree that do not have sufficient votes to be included, thereby revealing additional inclusion counts that would not have been visible in the honest protocol (while still not revealing which parties are responsible for those counts). This may or may not be a desirable tradeoff for a given deployment scenario.

5.4 Reducing rounds with short trees

We can reduce the round complexity of our protocol by breaking the output of the hash function H into two halves of length $\kappa/2$ and running the protocol separately on each half in parallel. Then a transaction is only included in the inclusion list if both halves of its hash

are present in the protocol output. This increases collision probability but cuts the number of rounds needed in half because the number of protocol rounds depends directly on the length of the hash, which determines the number of protocol levels.

Concretely, the probability of a collision in a list of q transactions is upper bounded by $q^2/2^\kappa$ for a κ bit hash. Since collisions in the left and right halves of a hash function modeled as a random oracle [16, 4] occur independently, collisions will occur with probability at most $\frac{q^2}{2^{\kappa/2}} \cdot \frac{q^2}{2^{\kappa/2}} = \frac{q^4}{2^\kappa}$.

To make the reduction in collision probability less severe, we instead split the output of H into three overlapping bit strings of length $\kappa/2$, requiring a transaction be included in all three halves in order to be included in the list. That is, we run the protocol with the first and second halves of the output of H, but also with the bits in positions $\kappa/4$ to $3\kappa/4$. This increases the computation per round but allows using a smaller value of κ to get the same concrete security level.

6 Evaluation

This section reports on our evaluation of our protocol implementation and empirical comparison to various baselines. See Section 7 for further discussion and qualitative comparison to other approaches which do not have a baseline implementation available.

6.1 Experimental setup

We conducted our experiments on four machines connected on a LAN. Each machine is equipped with an Intel Xeon Gold 6248R CPU @ 3.00GHz, running on VMware virtualization. Each of the machines have 16 virtual CPUs, 10Gbps bandwidth, and 256 GB of disk storage. One of the machines has 64 GB RAM whereas the rest have 48 GB RAM.

We note that based on the functionality provided by the underlying framework, the experiments on `MP-SPDZ` utilized all four machines equally, simulating multiple parties per machine when needed, whereas the ones on `HB-MPC` were run on a single machine. Moreover, for `MP-SPDZ`, each party's instance could utilize multiple CPU cores, when available, to improve performance.

6.2 Baselines

A natural point of comparison for our protocol would be other proposals for privacy-preserving inclusion lists. Unfortunately, the existing proposals are sketches and not yet fully formalized, and therefore do not have reference implementations we could use as a baseline. We qualitatively compare our approach to them in Section 7. In the absence of available implementations of other baselines, we consider two different MPC-based baseline algorithms to which we compare our solution. We implement both baselines using MP-SPDZ. The first baseline, `base-naïve`, is a naïve baseline where, for every input transaction from every party, we compare it with every other transaction to check if it appears at least τ number of times. Performing this computation requires $O(n^2|L_i|^2)$ comparisons, where each comparison requires at least $O(\kappa)$ multiplication gates under MPC, where κ is a security parameter corresponding to the length of transaction hashes, even if we ignore the cost of checking for the threshold to reach τ. Observe that implementing the output of `base-naïve` under MPC requires either using an oblivious list or naively adding every element to the secret-shared output and securely shuffling the output before revealing it. Our implementation does not perform the shuffling step, and thus the reported performance is a lower bound on the actual performance cost incurred by this algorithm.

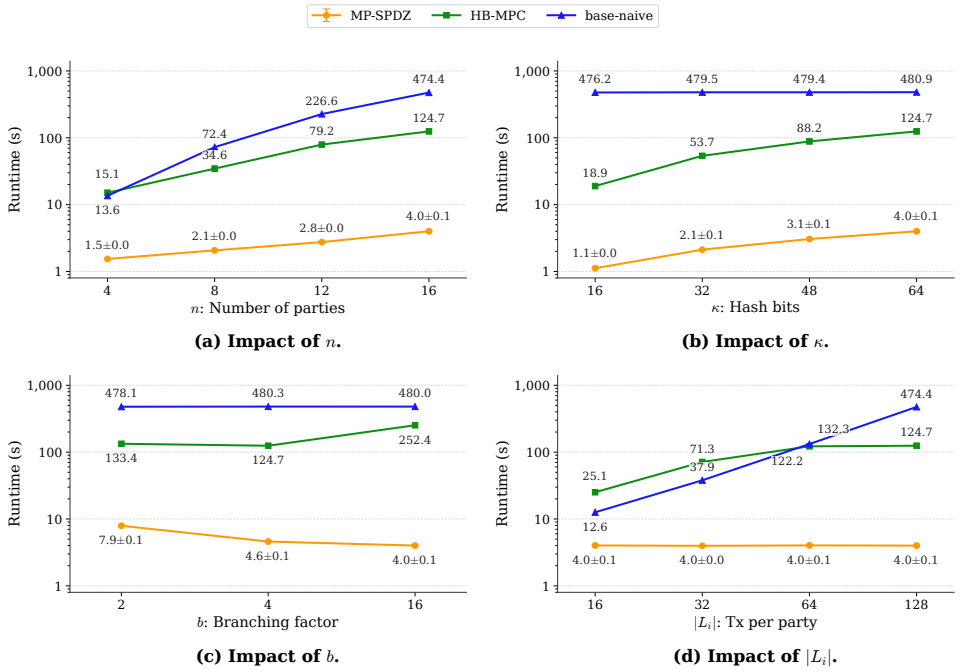


Figure 5 Performance as a function of parameters.

6.3 Experiment parameters and metrics

We choose our parameters in line with EIP-7805 [30], which considers the proposed implementation for inclusion list design (without privacy) for Ethereum. Thus, we consider $n = 16$. We set the mempool size to 128 and let the parties choose transactions from this mempool up to the maximum of $|L_i| = 128$ for all parties P_i. We set $\kappa = 64$; thus, we consider the first 64 bits of a transaction hash in our tree exploration. Moreover, we choose $b = 16$ empirically as the default parameter for `MP-SPDZ` and $b = 4$ for `HB-MPC` since it yields better performance (as will be shown later).

The primary metric of interest is the latency for running the inclusion list protocol; thus we report the maximum time across all honest parties for the online portion of the MPC protocol. For each parameter setting, for `MP-SPDZ`, we repeated the experiments five times and report the mean latency (across all parties and all runs) together with 95% confidence intervals. For `base-naïve` and `HB-MPC`, since they have a longer running time, we executed these experiments only once; we present the mean latency across all parties.

6.4 Evaluation Results

6.4.1 Efficiency in the default setting

In the default setting, the `MP-SPDZ` implementation runs in 4.0 ± 0.1s whereas the `HB-MPC` implementation runs in 124.7s. Thus, the execution time for achieving optimistic correctness is within the 12s slot time for Ethereum. The offline time needed for MPC setup under the default setting is 14.7s for `MP-SPDZ` and 33.2s for `HB-MPC`. We note that this offline setup can be performed in advance and does not affect the online MPC latency reported above.

The running time of the more efficient `MP-SPDZ` implementation is dominated by 16,384 MPC multiplications distributed over 32 sequential levels of execution. These results suggest that the core computation performed by our protocol is lightweight enough for slot-level

optimistic execution in our default experimental setting. In this sense, the evaluation supports the computational efficiency of our design, but the number of sequential communication rounds induced by the tree traversal is now the bottleneck.

We also conducted preliminary experiments in a WAN setting. As expected, WAN latency increases the running time of the current implementation. We therefore view the WAN results as a stress test of the current implementation rather than a fully optimized deployment benchmark. Several optimizations described in Section 5, including speculative execution and shorter trees, are designed to reduce the effective round complexity of the protocol. In addition, the tree structure exposes further opportunities for parallelism across independent subtrees. We leave a full WAN-optimized implementation and evaluation of these optimizations to future work.

While our robust protocol variant guarantees output delivery, its latency significantly exceeds the 12-second Ethereum slot time. This reflects a tradeoff between performance and robustness; the HoneyBadgerMPC framework uses heavier tools to provide the stronger robustness property. Thus, while our protocol demonstrates the feasibility of a robust private inclusion list protocol, which could perhaps be run in parallel with the faster, non-robust protocol as a fail-safe, further performance improvements will be needed to meet real-world requirements. Fortunately, improvements in the underlying robust MPC will directly lead to improvements in our protocol's performance.

6.4.2 Protocol performance as a function of parameters

Figures 5 present the performance of `MP-SPDZ` and `HB-MPC` as a function of n, κ, τ, and $|L_i|$.

First, we observe that as n increases, the execution time for both protocols increases. For `MP-SPDZ`, the multiplication gates scale linearly in n, while communication grows quadratically. For `HB-MPC`, the communication grows super-linearly and the computation increases linearly in n.

Second, as κ increases, we observe that the execution time increases roughly linearly. Every level of the tree, except for the first few levels, performs a roughly similar amount of computation corresponding to the max number of prefixes/transactions processed per level. In the first few levels, the number of prefixes processed per level increases until it reaches that threshold. Thus, we observe a roughly linear change in latency with linear change in κ.

Third, $b = 16$ works best for `MP-SPDZ`, and $b = 4$ performs better for `HB-MPC`. Observe that a larger b implies checking for a larger number of prefixes at once; thus, this increases the amount of communication. At the same time, for a fixed κ, the height of the tree is $\log_b \kappa$, and thus, the number of levels decreases. It is not always guaranteed a monotonic relationship between branching factor and performance.

Finally, the number of transactions $|L_i|$ in each party's input does not affect performance very much for `MP-SPDZ`. This is due to a framework constraint where the input size processed at each level must be hard-coded in advance, we therefore set this size to 128, which matches the mempool size used in Section 5. `HB-MPC` performs faster with smaller size of inputs due to fewer computation counts.

6.4.3 Comparison with baselines

Figure 5 compares the performance of `MP-SPDZ` with baselines `base-naïve`. We execute the `base-naïve` protocol only for a subset of data points since its performance is much slower than `MP-SPDZ`. Under the default parameterization, it is $\sim 120\times$ slower than `MP-SPDZ`.

Our performance improvement over the baselines is explained by the smaller number of MPC multiplications needed in our protocol. Since multiplications of secret shared data are the most expensive operation in an MPC, we seek to minimize the number of multiplications, even at the cost of more additions, which are effectively free. Thus, since our solution generally requires fewer multiplications than `base-naïve`, we have significantly better performance.

7 Discussion

This section discusses general issues related to inclusion lists that are not specific to our protocol. It also compares our approach qualitatively to other proposed or potential approaches to private inclusion lists. Given that there are not concrete proposals or implementations to which we can compare, an empirical comparison is unfortunately impossible. Nonetheless, we hope this discussion helps to frame the tradeoffs results in the context of other related approaches.

7.1 Committee Membership Risks

Our protocol has been designed to be compatible with a number of different deployment scenarios for inclusion lists. For example, committee members can be chosen in any way as long as the selection process ends by publicly identifying the committee members. However, there are certain inherent limitations to this kind of inclusion list design. While correcting the general (non-privacy-related) flaws of inclusion lists is outside the scope of this work, the widespread attention inclusion lists have received makes them an important topic for study.

While private inclusion lists help hide which members of a committee chose to include a transaction in an inclusion list, the fact that a transaction was included means that each member may be subject to heightened scrutiny or suspicion for including that transaction. This places a limit on how much plausible deniability such a protocol can provide, as there is always a τ/n chance that each member included each transaction, by definition of the inclusion criteria. It is possible that this inherent limit on plausible deniability means that the entire committee can be held responsible for including a transaction. The only alternative we foresee to address this limitation is if the committee consists of all validators as in zkFOCIL [2] (discussed in the next subsection).

7.2 Ethereum Proposals

There are two high-level proposals in the Ethereum community. The first is an "anonymous inclusion list" design that combines a committee-based design (akin to this work) with linkable ring signatures [21, 5] to achieve anonymity [1]. All committee members sign their local inclusion lists without revealing their specific identity, instead proving they belong to the valid group. It relies on an implementation of an anonymous broadcast protocol based on Dandelion++ [15] to protect anonymity at the network layer. As described in the blog post [1], a thorough security analysis is necessary to assess the trade-offs between network anonymity and security. Our proposal eliminates the need for linkable ring signatures and replaces the anonymous broadcast with limited use of MPC, whose provable security properties are more easily analyzed and understood.

The second design sketch, called zkFOCIL [2], uses a private lottery scheme to select the inclusion list committee obliviously. Their instantiation also relies on linkable ring signatures, and the proposal includes a proposed scheme based on general-purpose SNARKs. This proposal increases the size of the anonymity set for committee members by making members

anonymous among the somewhat larger set of validators from which the committee is chosen, attempting to mitigate the plausible deniability issues inherent to the committee-based IL design. However, anonymity in this approach still relies on the underlying anonymous broadcast mechanism. A recent implementation of zkFOCIL [25] focuses primarily on the timing of private committee selection and proof generation. It does not yet provide an end-to-end implementation and benchmark of the anonymous broadcast channel, so the full online latency of constructing and publishing an inclusion list remains unclear. Moreover, private committee selection provides an additional layer of pseudonymity, but pseudonymity alone does not imply transaction-level plausible deniability: if messages, timing, or dissemination patterns can be linked to the same anonymous participant, then access-pattern-style analysis may still leak additional information. In contrast, our MPC-based design focuses on the transaction-selection layer: given a committee and its members' local transaction-hash lists, we provide and evaluate a concrete protocol for producing an inclusion list while hiding which members contributed to each transaction.

7.3 Other Potential Approaches

Since our protocol provides an efficient way for committee members to anonymously send each other the contents of their inclusion lists, one alternative would be to rely on a classic approach like DC-nets [9], whereby committee members could anonymously broadcast the contents of their inclusion lists and then count up the number of times each transaction has appeared. In a semihonest setting, this approach would require more communication than our scheme, but would otherwise be feasible. However, when considering malicious clients, it becomes more challenging. Malicious clients may try to send random messages to clobber others' contributions or send the same transaction(s) multiple times to inflate their count. While techniques exist to ensure that participants in a DC-net cannot corrupt the outputs of the protocol [20, 10, 11], additional ideas would be needed to ensure that parties cannot enter duplicate inputs.

Another approach to inclusion list construction would be to rely on protocols for private set intersection [18]. While two-party PSI protocols have become increasingly efficient, multiparty protocols lag behind in performance [34]. One of our naïve baselines was to build a PSI-type protocol on top of MPC, and perhaps this approach could be refined to build more efficient protocols. Specifically, inclusion list formation as described here can be thought of as a multi-party PSI cardinality problem. In fact, there is such a protocol implicit in our construction, albeit with additional leakage that would not be suitable for the standard PSI problem formulation. Moreover, just as our core semihonest protocol needs additional protections for malicious behavior and to preserve deniability, any PSI protocol would need to be modified to provide similar features as well.

8 Conclusion

We have presented a new protocol to achieve privacy-preserving inclusion lists. Our protocol allows members of an inclusion list committee to collectively construct an inclusion list without revealing which transactions they have chosen to include and while maintaining plausible deniability that they did not contribute any particular transaction. In the presence of centralization of block proposing power, our hope is that our protocol can allow committee members to prevent retaliation and ensure robust censorship resistance.

References

1 0xF1D. Anonymous inclusion lists (anon ils). https://ethresear.ch/t/anonymous-inclusion-lists-anon-ils/19627, 2024. Accessed: 2025-08-21.

2 0xF1E. zkfocil: Inclusion list privacy using linkable ring signatures. https://ethresear.ch/t/zkfocil-inclusion-list-privacy-using-linkable-ring-signatures/21688, 2025. Accessed: 2025-08-21.

3 Donald Beaver. Efficient multiparty protocols using circuit randomization. In *CRYPTO*, 1991.

4 Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS*, 1993.

5 Ward Beullens, Shuichi Katsumata, and Federico Pintore. Calamari and falafl: logarithmic (linkable) ring signatures from isogenies and lattices. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 464–492. Springer, 2020. doi:10.1007/978-3-030-64834-3_16.

6 Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Lightweight techniques for private heavy hitters. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 762–776. IEEE, 2021. doi:10.1109/SP40001.2021.00048.

7 Vitalik Buterin. State of research: increasing censorship resistance of transactions under proposer/builder separation (PBS). HackMD, January 2022. URL: https://notes.ethereum.org/@vbuterin/pbs_censorship_resistance.

8 Vitalik Buterin and Mike Neuder. No free lunch – a new inclusion list design. Ethereum Research, August 2023. URL: https://ethresear.ch/t/no-free-lunch-a-new-inclusion-list-design/16389.

9 David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *J. Cryptology*, 1(1):65–75, 1988. doi:10.1007/BF00206326.

10 Henry Corrigan-Gibbs and Bryan Ford. Dissent: accountable anonymous group messaging. In *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 340–350. ACM, 2010. doi:10.1145/1866307.1866346.

11 Henry Corrigan-Gibbs, David Isaac Wolinsky, and Bryan Ford. Proactively accountable anonymous messaging in verdict. In *Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA, August 14-16, 2013*, pages 147–162. USENIX Association, 2013. URL: https://www.usenix.org/conference/usenixsecurity13/technical-sessions/presentation/corrigan-gibbs.

12 Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, 2012. doi:10.1007/978-3-642-32009-5_38.

13 Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency - choose two. In *IEEE Symposium on Security and Privacy*, 2018.

14 Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. Comprehensive anonymity trilemma: User coordination is not enough. *PoPETs*, 2020(3):356–383, 2020. doi:10.2478/POPETS-2020-0056.

15 Giulia Fanti, Shaileshh Bojja Venkatakrishnan, Surya Bakshi, Bradley Denby, Shruti Bhargava, Andrew Miller, and Pramod Viswanath. Dandelion++: Lightweight cryptocurrency networking with formal anonymity guarantees. *arXiv preprint arXiv:1805.11060*, 2018. doi:10.48550/arXiv.1805.11060.

16 Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, 1986.

17 Elijah Fox, Mallesh M. Pai, and Max Resnick. Censorship Resistance in On-Chain Auctions. In Joseph Bonneau and S. Matthew Weinberg, editors, *5th Conference on Advances in Financial Technologies (AFT 2023)*, volume 282 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 19:1–19:20, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.AFT.2023.19.

18 Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2004. doi:10.1007/978-3-540-24676-3_1.

19 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, 1987.

20 Philippe Golle and Ari Juels. Dining cryptographers revisited. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 456–473. Springer, 2004. doi:10.1007/978-3-540-24676-3_27.

21 Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 253–280. Springer, 2015. doi:10.1007/978-3-662-46803-6_9.

22 Marcel Keller. MP-SPDZ: A versatile framework for multi-party computation. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pages 1575–1590. ACM, 2020. doi:10.1145/3372297.3417872.

23 Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 830–842. ACM, 2016. doi:10.1145/2976749.2978357.

24 Quintus Kilbourn. Censorship Resistance: crlists in mev-boost issue #215. GitHub, July 2022. URL: https://github.com/flashbots/mev-boost/issues/215.

25 Shreyas Londhe and Suyash Bagad. zkFOCIL: Implementation & Benchmarking Report - Cryptography, January 2026. Section: Cryptography. URL: https://ethresear.ch/t/zkfocil-implementation-benchmarking-report/23966.

26 Donghang Lu, Thomas Yurek, Samarth Kulshreshtha, Rahul Govind, Aniket Kate, and Andrew Miller. Honeybadgermpc and asynchromix: Practical asynchronous MPC and its application to anonymous communication. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 887–903. ACM, 2019. doi:10.1145/3319535.3354238.

27 Credit and debit card market share by network and issuer. https://www.fool.com/money/research/credit-debit-card-market-share-network-issuer/, 2025. Accessed: 2025-08-21.

28 Mike Neuder and Toni Wahrstätter. Unconditional inclusion lists. Ethereum Research, January 2024. URL: https://ethresear.ch/t/unconditional-inclusion-lists/18500.

29 Burak Öz, Danning Sui, Thomas Thiery, and Florian Matthes. Who wins ethereum block building auctions and why?, 2024. doi:10.48550/arXiv.2407.13931.

30 Thomas Thiery, Francesco D'Amato, Julian Ma, Barnabé Monnot, Terence Tsao, Jacob Kaufmann, and Jihoon Song. EIP-7805: Fork-choice enforced Inclusion Lists (FOCIL). Ethereum Improvement Proposals, July 2024. URL: https://eips.ethereum.org/EIPS/eip-7805.

31 Thomas Thiery, Barnabé Monnot, Francesco D'Amato, and Julian Ma. Fork-Choice enforced Inclusion Lists (FOCIL): A simple committee-based inclusion list proposal. Ethereum Research, June 2024. URL: https://ethresear.ch/t/fork-choice-enforced-inclusion-lists-focil-a-simple-committee-based-inclusion-list-proposal/19870.

32 Zhengwei Tong, Saba Eskandarian, and Kartik Nayak. Privacy-Preserving Inclusion Lists, 2026. Publication info: Published elsewhere. Minor revision. To appear in Advances in Financial Technologies (AFT 2026). URL: `https://eprint.iacr.org/2026/1582`.

33 U.S. Department of the Treasury. OFAC Specially Designated Nationals Data, 2022. URL: `https://www.treasury.gov/ofac/`.

34 Jelle Vos, Mauro Conti, and Zekeriya Erkin. Sok: Collusion-resistant multi-party private set intersections in the semi-honest model. In *IEEE Symposium on Security and Privacy, SP 2024*, pages 465–483. IEEE, 2024. `doi:10.1109/SP54263.2024.00079`.

35 Sarisht Wadhwa, Julian Ma, Thomas Thiery, Barnabe Monnot, Luca Zanolini, Fan Zhang, and Kartik Nayak. AUCIL: An inclusion list design for rational parties. Cryptology ePrint Archive, Paper 2025/194, 2025. URL: `https://eprint.iacr.org/2025/194`.

36 Anton Wahrstätter, Jens Ernstberger, Aviv Yaish, Liyi Zhou, Kaihua Qin, Taro Tsuchiya, Sebastian Steinhorst, Davor Svetinovic, Nicolas Christin, Mikolaj Barczentewicz, and Arthur Gervais. Blockchain Censorship, June 2023. arXiv:2305.18545 [cs]. `doi:10.48550/arXiv.2305.18545`.

37 Fredrik Winzer, Benjamin Herd, and Sebastian Faust. Temporary censorship attacks in the presence of rational miners. In *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 357–366. IEEE, 2019. `doi:10.1109/EUROSPW.2019.00046`.

38 Fei Wu, Thomas Thiery, Stefanos Leonardos, and Carmine Ventre. From competition to centralization: The oligopoly in ethereum block building auctions, 2024. `doi:10.48550/arXiv.2412.18074`.

39 Sen Yang, Kartik Nayak, and Fan Zhang. Decentralization of ethereum's builder market, 2025. `doi:10.48550/arXiv.2405.01329`.

SoK: Cryptographic Key Recovery for Cryptoasset Custody and Financial Technologies

Francisco Javier Becerra Sanchez
University of Luxembourg, Luxembourg

Antonio Ken Iannillo
University of Luxembourg, Luxembourg

Radu State
University of Luxembourg, Luxembourg

Abstract

Cryptoasset systems often bind cryptographic key control to financial control: losing a wallet seed, custody share, hardware device, or smart-account credential can remove spend authority, while compromised recovery can enable theft. Existing work treats recovery through separate vocabularies–key backup, secret sharing, account recovery, credential re-issuance, social recovery, and asset migration–making mechanisms and tradeoffs difficult to compare.

This paper presents a Systematization of Knowledge (SoK) on cryptographic key recovery for cryptoasset custody and financial technologies. Starting from a 118-paper systematic-review discovery corpus, we derive a 77-paper synthesis corpus and code each retained system in a master matrix covering recovered objects, recovery semantics, mechanisms, enrollment and storage, authorization, trust placement, failure events, post-recovery state, validation evidence, deployment status, privacy, usability, and limitations. The matrix supports an axis-first taxonomy that separates secret-restoring, hybrid, control-restoring, forensic/extractive, and framework-oriented recovery.

Our central observation is that recovery is not a single operation: systems may reconstruct an original secret, regenerate a seed, restore a share, reissue a credential, migrate signing authority, restore account control, move assets, or extract forensic artifacts. We derive a generalized construction model, check it against production-facing designs, and identify six findings: recovery semantics are heterogeneous; recovery shifts trust; liveness improvements create abuse paths; post-recovery lifecycle management is uneven; protocol evidence outpaces user evidence; and recovery metadata remains underprotected. These gaps motivate a research agenda for recovery-aware financial technologies.

2012 ACM Subject Classification Security and privacy → Key management

Keywords and phrases Cryptoasset Custody, Key Recovery, Wallet Security, Smart Accounts, Threshold Cryptography, Secret Sharing, Decentralized Identity, Secure Hardware

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.3

Related Version *Full Version*: `https://arxiv.org/abs/2608.07104`

Supplementary Material
Collection (Reproducibility Package): `https://github.com/franciscobecerra97/Cryptographic-Key-Recovery-for-Financial-Technologies/tree/aft-2026-artifacts` [86]

Funding This research was funded in part by the Luxembourg National Research Fund (FNR), grant NCER22/IS/16570468/NCER-FT, and by FinnovationHub, funded by the Ministry of Finance of the Government of Luxembourg.

Acknowledgements The authors thank colleagues and reviewers for helpful feedback.

© Francisco Javier Becerra Sanchez, Antonio Ken Iannillo, and Radu State;
licensed under Creative Commons License CC-BY 4.0

1 Introduction

In cryptoasset systems and related financial technologies, control of a cryptographic key can become control of financial value: loss may remove access to funds or services, while compromised recovery may transfer that control to an attacker. Key recovery is therefore part of the security boundary for systems that rely on cryptographic authorization [110, 92, 8, 95].

Recovery must balance two goals that pull against each other. It should prevent legitimate users and institutions from being permanently locked out after loss, unavailability, or operational failure. At the same time, it must not create a weaker path for theft, coercion, surveillance, insider abuse, or unauthorized transfer. This tension is the core reason recovery remains difficult: the mechanisms that improve availability also introduce new actors, records, and decisions that must be secured.

Existing work proposes many mechanisms, but uses incompatible vocabulary. Recovery may mean restoring cryptographic material, changing account control, moving value, reissuing identity credentials, repairing custody roles, or extracting forensic artifacts. Comparisons can therefore obscure what is actually restored and what risks remain after recovery.

This paper addresses that fragmentation through a Systematization of Knowledge (SoK) focused on cryptographic key recovery for cryptoasset custody and financial technologies. We center the paper on financial control, while including adjacent identity, data-protection, governance, and secure-hardware work when it contributes a recovery pattern that can transfer to financial systems.

Recovery is not one operation. A useful comparison must ask what control is restored, who authorizes it, what failure it handles, and what state remains afterward.

1.1 Contributions

This Systematization of Knowledge makes the following contributions:

- **Financial-technology corpus and scope.** We construct a 118-paper discovery corpus on key backup, recovery, and restoration, with auditable queries, screening flow, exclusion reasons, snowballing rounds, QA threshold, and manually verified LLM-assisted screening. We then derive a 77-paper synthesis corpus scoped around financial technologies while retaining transferable recovery patterns from identity, hardware, and data-protection systems.
- **Recovery semantics for financial control.** We distinguish secret-restoring recovery from control-restoring recovery and show why this distinction matters for financial systems. This clarifies when a system restores the same secret and when it instead restores financial control through a changed control relationship.
- **A 77-system master matrix and codebook.** We code every retained system using a controlled vocabulary for financial context, recovery semantics, mechanism family, enrollment and storage, failure events, authorization, trust placement, adversary model, post-recovery state, validation evidence, deployment status, privacy, usability, post-quantum relevance, and limitations. The full version provides the complete per-system matrix, while the reproducibility package provides the machine-readable matrix and its codebook.
- **Evaluation vocabulary, taxonomy, and construction model.** We organize mechanisms by whether they restore secrets, restore control, combine both, perform forensic extraction, or define requirements. We define adversary classes and recovery-specific safety, liveness, and privacy properties, then derive a seven-stage construction model.

- **Production practice checks and research agenda.** We check the taxonomy against six production or standards-facing recovery designs and tie six findings to matrix counts and examples, including sparse user-study evidence, under-specified post-recovery hygiene, metadata leakage, custody boundaries, recovery economics, inheritance, and post-quantum migration.

1.2 SoK Framing and Novelty

Motivation for systematization

The goal is not only to summarize prior mechanisms. We define comparable recovery semantics, code every retained system into the master matrix provided in the full version, and separate secret reconstruction from restored financial control. A threshold wallet that reconstructs an old seed, a smart account that rotates ownership, and a recovery service that moves assets are all recovery systems, but they leave different post-recovery states and abuse paths.

Novelty relative to closest related work

Prior work usually studies one slice of the problem: cryptocurrency protocols and DeFi, wallet protocols and attacks, threshold recovery, deterministic or hardware-wallet designs, smart-account recovery, identity recovery, or forensic extraction. This SoK contributes a cross-domain comparison centered on recovery semantics rather than mechanism names. Its novelty claims are: *(i)* secret-restoring and control-restoring recovery should be evaluated separately; *(ii)* post-recovery state is a first-class security property for financial systems; and *(iii)* recovery evidence must connect cryptographic mechanisms with operational controls, metadata exposure, production practice, and human behavior. Table 1 compares the closest prior-work clusters.

2 Background and Scope

This paper uses the term *cryptographic key recovery* broadly, but not without distinction. In the corpus, recovery may mean reconstructing an original private key, regenerating a deterministic seed, restoring a key share, reissuing a credential, transferring signing authority, recovering account control, or moving assets to a new control path. These operations differ in what they restore and in the risks they introduce.

The primary scope is financial technology: systems in which cryptographic material authorizes assets, transactions, account access, custody operations, or credentials used by financial infrastructure. This includes blockchain wallets, smart-contract accounts, custody systems, decentralized applications, decentralized identity, and secure hardware when they support financial control or transferable recovery patterns.

The paper also considers non-financial work when it contributes a mechanism relevant to financial systems. Examples include identity-management recovery, data-protection recovery, governance-related recovery, and hardware-enforced recovery. These systems are included because financial technologies often reuse the same primitives and operational patterns: secret sharing, threshold reconstruction, deterministic derivation, secure elements, trusted authorities, biometric or physical anchoring, and delegated recovery.

Table 1 Closest related-work clusters and this paper's added recovery-semantics focus.

Prior work	Scope	Recovery semantics	Financial-control treatment	Corpus or evidence	This paper adds
Cryptocurrency and DeFi SoKs [6, 97, 107]	Protocols, DeFi primitives, risks, and incidents	Mostly indirect: key management, incidents, rescue windows, or mitigation	Protocol and market-level financial control	SoK syntheses and incident datasets	Recovery-specific matrix for wallets, custody, smart accounts, identity, and forensics
Wallet taxonomies and security surveys [56, 46, 45, 24]	Wallet definitions, protocols, factors, attacks, and defenses	Recovery is a feature or fallback, not the organizing axis	Wallets as transaction interfaces and attack targets	Taxonomies, SLR evidence, and incident analysis	Recovered objects, secret/-control axis, post-recovery state, and lifecycle evidence
Custody and key-management work [103, 95]	Dynamic access structures, threshold custody, HSMs, and operations	Access update, backup restoration, share reconstruction, or signer continuity	Central: assets must stay live without weakening controls	Formal constructions, prototypes, benchmarks, and cases	Cross-family comparison with regulation, economics, inheritance, production cases, and users
Web3 account recovery [110, 17, 5, 79, 22, 12, 58]	Guardians, fallback signatures, passkeys, presigned transfers, and composition	Concrete control-restoring mechanisms	Direct wallet and smart-account asset control	Mechanism papers, simulations, gas costs, and composability	Shows how control restoration differs from secret restoration and exposes hygiene/-metadata gaps
Identity and WebAuthn recovery [92, 93, 59, 94, 34, 60]	SSI wallets, DID backup, backup authenticators, credential reissue, and share restoration	Credential or identity-control recovery	Transferable to regulated payments and account access	Prototypes, benchmarks, and formal analyses	Separates identity restoration from asset-control recovery and maps transfer to finance
Usability and key-management studies [98, 63, 88, 90]	Human coordination, wallet mental models, memory cues, seed phrases, and factor loss	User-facing ceremonies and factor handling	Often implicit through wallet access	Three retained user-study rows plus adjacent usability evidence	Conservative user-evidence table and agenda for stress, death, phishing, and accessibility

3 Review Methodology and Corpus Construction

The contribution of this paper is the systematization, not the review protocol itself. We nevertheless used a systematic literature review (SLR) workflow as an auditable discovery process and then narrowed the final synthesis around recovery in cryptoasset custody and financial technologies.

Table 2 Database search audit. The table separates library exports from normalized screening input so reviewers can trace the corpus before snowballing and SoK-stage filtering.

Database	Exported	Normalized	Syntax and filter note
Scopus	218	129	`TITLE-ABS-KEY` logical query; Computer Science, English, article/conference filters
IEEE Xplore	462	434	All-metadata logical query; English conference/journal records exported in BibTeX
ACM Digital Library	1,542	1,506	Full-record or metadata logical query; ACM BibTeX batch exports
ScienceDirect	2,100	2,075	Title/abstract/keywords logical query; ScienceDirect BibTeX batch exports

3.1 Guiding Questions

The review was guided by four questions: **RQ1:** What object or control relationship is recovered? **RQ2:** What cryptographic primitives, system architectures, and operational procedures enable recovery? **RQ3:** What safety, liveness, privacy, usability, and trust tradeoffs are reported? **RQ4:** What open challenges remain for wallets, smart accounts, custody systems, decentralized identity, secure hardware, and related financial infrastructure?

3.2 Discovery Corpus

We searched Scopus, IEEE Xplore, the ACM Digital Library, and ScienceDirect for peer-reviewed English-language computer-science papers on cryptographic key backup, recovery, and restoration. The search was run and exported on 14 October 2025. The logical query was:

> ("key" OR "private key" OR "cryptographic key") AND ("recovery" OR "backup") AND ("public key infrastructure" OR blockchain OR cryptocurrency OR wallet OR "self-sovereign identity" OR "decentralized identity" OR "identity management" OR "digital identity").

The same logical query was adapted only for database field syntax and export format. Table 2 reports the database-specific search surface and counts. The SLR log records 13,637 initial database-interface hits before database-side filters; the local export audit contains 4,322 BibTeX records and 4,144 normalized records. The round-0 screening log contains 4,314 records after basic filters and 4,044 after duplicate removal.

The initial title-and-abstract screening used a Large Language Model through an API as a decision-support tool: `gpt-5` for title/abstract screening and `gpt-5-mini` for basic filtering of snowballing references. Each prompt included the candidate title and abstract, inclusion and exclusion criteria, and a request for an accept/reject verdict plus justification. Outputs were recorded as status and comments fields. Both LLM-assisted inclusions and exclusions were manually verified, and accepted records were reconciled into the round-level BibTeX files before quality assessment. The model scaled screening; it did not make final inclusion decisions.

Backward and forward snowballing were applied until convergence: a final iteration produced no new snowballing references and left the retained primary-study count unchanged at 312. Table 3 reports the round-level counts.

Table 4 summarizes grouped exclusion reasons. Title/abstract reasons were rule-coded from the manually verified decision comments. The final SoK-stage exclusions were coded from the full-text extraction rows.

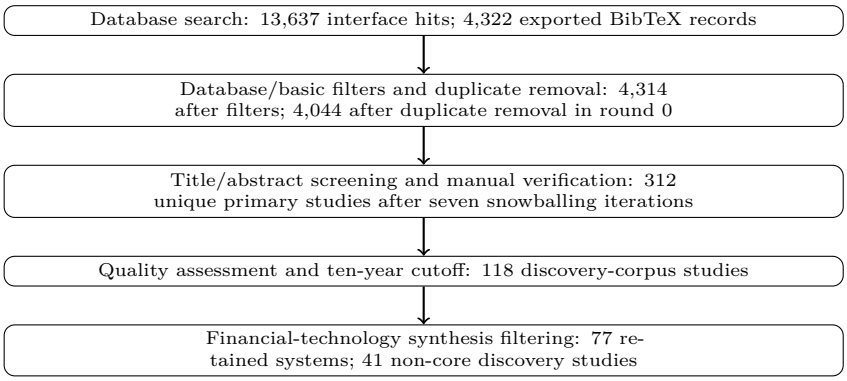


Figure 1 Auditable corpus-construction flow. The key transition is from broad SLR discovery to a 77-system synthesis corpus used for taxonomy, matrix counts, and findings.

Table 3 Screening and snowballing counts. Round 7 provides the stopping rule. RA = research articles or candidate references; BF = after basic filters; RD = after duplicate removal; IE = cumulative retained records after inclusion/exclusion screening; BS = new backward/forward snowballing references.

Round	RA	BF	RD	IE	BS
0	13,637	4,314	4,044	129	1,820
1	1,949	1,147	817	230	1,390
2	1,620	849	623	281	668
3	950	594	478	303	208
4	511	401	360	308	67
5	375	331	321	311	21
6	332	321	315	312	5
7	317	314	313	312	0

Table 4 Grouped exclusion reasons. The table distinguishes title/abstract screening exclusions from full-text SoK-stage exclusions, showing why records left the evidence base.

Title/abstract exclusion reason	Count	SoK-stage exclusion reason	Count
Cryptographic security or authentication, but no recovery	2,119	Transferable non-financial background only	28
Adjacent domain, but no recovery mechanism	1,866	Financial-adjacent but not core comparable recovery	9
Review or non-primary literature	434	Requirements or background only	4
Unrelated application or out of scope	359		
Key generation/protection, not recovery of an existing key	279		
No cryptographic private-key or ownership context	108		
Insufficient or missing abstract	15		

3.3 Quality Assessment and Discovery Corpus

After snowballing convergence, 312 primary studies were retained for quality assessment. Each study was scored using five criteria: evidence and validation; cryptographic foundations and techniques; security analysis and limitations; proposed solution or contribution; and open problems or future directions. Each criterion was scored as yes (1.0), partial (0.5), or no (0.0).

Table 5 Two-stage corpus construction. Reviewers should read the 118 papers as the discovery corpus and the 77 retained papers as the evidence base for the SoK's comparative claims.

Stage	Role in the paper	Result
SLR discovery	Identify peer-reviewed work on cryptographic key backup, recovery, and restoration through database search, manual verification, and snowballing.	118 papers
Financial-technology extraction	Retain papers with concrete recovery semantics and direct or transferable relevance to wallets, custody, smart accounts, decentralized identity, secure hardware, or financial infrastructure.	77 papers

Applying the QA threshold retained 194 studies with score at least 4.0. Of these, 76 were pre-2016 foundational papers and are used as background where appropriate, while 118 had score at least 4.0 and publication year at least 2016. Those 118 studies form the discovery corpus used for full-text extraction and SoK narrowing.

3.4 Financial-Technology Synthesis Corpus

The 118-paper discovery corpus was intentionally broader than the final SoK needed to be. It included direct wallet and custody papers, but also adjacent work on generic key management, authentication, identity, hardware, cloud backup, and data protection. For the final synthesis, we therefore performed a second filtering step after full-text extraction.

Each extracted paper was coded for recovered object, secret/control recovery axis, mechanism family, enrollment and storage, recovery trigger, failure-event class, authorization, trust placement, adversary model, post-recovery state, financial-technology context, validation evidence, deployment status, privacy, usability, post-quantum relevance, and limitations. We retained papers that described a concrete recovery, backup, restoration, transfer, re-issuance, or extraction mechanism; were directly financial or clearly transferable to financial control; and contained enough detail to compare recovery semantics and tradeoffs. We excluded papers where recovery was peripheral, generic without a financial-technology transfer, or not comparable along the SoK dimensions.

This filtering reduced the corpus from 118 discovery papers to 77 synthesis papers. The excluded 41 papers were not treated as irrelevant; rather, they informed boundary-setting and background but were not used as core evidence for the taxonomy, counts, and cross-cutting findings. Figure 1 summarizes the full flow, while Table 5 summarizes the final two-stage construction.

4 Corpus Overview

The 77-paper synthesis corpus is the evidence base for the taxonomy, mechanism comparison, and cross-cutting findings. It is narrower than the 118-paper discovery corpus because the goal is to systematize mechanisms that restore cryptographic or financial control, not to catalog every paper that mentions key recovery. The retained studies are concentrated in self-custody and blockchain wallet settings, with additional evidence from institutional custody, permissioned blockchains, smart-contract accounts, decentralized identity, secure hardware, and forensic recovery. Categories are non-exclusive.

Table 6 Non-exclusive corpus facets. Counts can exceed 77 within a facet when a study belongs to multiple categories.

Facet	Largest extracted groups
Financial context	Self-custody wallets (57), payment or permissioned blockchains (14), transferable non-financial patterns (15), institutional custody (11), DeFi or smart-contract applications (10), smart-contract accounts (9), decentralized identity for finance (8), secure hardware for financial infrastructure (8).
Recovered object	Original private keys (34), signing authority (20), account control (19), seed or root secrets (15), asset ownership or transfer paths (13), key shares (10), recovery keys (5), credentials (3), forensic artifacts (3).
Validation evidence	Prototypes (56), benchmarks (50), formal proofs (28), simulations (17), security games (11), case studies (9), gas-cost measurements (7), user studies (3), no evaluation in the extracted evidence (6), operational deployment evidence (1).

The venue and validation mix affects how the counts should be read. The corpus is recent-heavy: 7 of the 77 synthesis rows are from 2016–2018, while the remaining rows are from 2019 or later, with 43 conference or symposium papers, 32 journal articles, and 2 workshop or CEUR papers. This makes the evidence strongest for mechanism comparison, protocol feasibility, and engineering validation. It is weaker for deployment prevalence, recovery under stress, and operational or user outcomes, which is why we report user-study and deployment evidence separately instead of treating prototype, proof, benchmark, and deployment rows as equivalent. Table 7 reports the confidence tiers used to interpret the venue and validation mix.

The master matrix gives each retained system one auditable row covering the comparison dimensions, evidence fields, limitations, and SoK relevance. Table 8 compresses that matrix by the paper's primary organizing axis. The reproducibility package provides the complete machine-readable matrix, its codebook, summary tables, and audit artifacts.

Table 7 Venue and evidence confidence tiers used to interpret corpus counts. The tiers affect confidence in claims drawn from the counts, not whether a retained row is counted in the matrix.

Confidence tier	Rows	Count interpretation
Higher for protocol soundness	28	Strongest for cryptographic claims, safety arguments, and mechanism feasibility; weaker for deployed prevalence or user behavior.
Higher for external validity	5	Stronger support for user, forensic, or operational observations, but still narrow relative to the breadth of wallet and custody practice.
Medium engineering evidence	40	Useful for comparing implemented prototypes, benchmarks, and design tradeoffs; not enough to infer production reliability.
Lower or contextual evidence	4	Included for requirements, boundary-setting, or conceptual relevance; used cautiously and not as strong evidence for mechanisms.

Table 8 Compressed view of the 77-system master matrix by recovery axis.

Recovery axis	Rows	Dominant mechanisms	Typical post-recovery state	Studies
Secret-restoring	38	Threshold reconstruction, password/factor-protected recovery, biometric or fuzzy regeneration, TEE/HSM-assisted recovery	Same or equivalent secret available; occasional share refresh or credential reset	[52, 98, 108, 53, 100, 43, 3, 47, 82, 92, 73, 91, 67, 80, 93, 35, 89, 66, 65, 106, 61, 88, 85, 81, 74, 55, 51, 104, 41, 105, 101, 48, 72, 10, 33, 9, 11, 18]
Hybrid secret/control	16	Deterministic regeneration, threshold continuity, institutional backup, social or guardian recovery	Secret material is restored while signing authority, custody participation, or account control may also change	[42, 70, 23, 7, 54, 68, 57, 79, 102, 59, 87, 95, 34, 78, 60, 20]
Control-restoring	16	Smart-contract control transfer, fallback or ownership proof, passkey/backup authenticator recovery, pre-signed asset evacuation	New owner, credential, signing path, or asset-control relation replaces the failed path	[110, 103, 109, 19, 17, 13, 8, 5, 4, 71, 21, 22, 96, 94, 90, 58]
Forensic/ extractive	5	Forensic artifact recovery and cryptanalytic key extraction	Artifact, credential, or key is extracted from device, backup, memory, nonce failure, or investigation evidence	[16, 62, 44, 49, 50]
Framework/ requirements	2	Requirements analysis and composability modeling	No single post-recovery state; evaluates recovery obligations or composed mechanisms	[77, 12]

5 Recovery Semantics and Evaluation Vocabulary

Recovery mechanisms in the corpus do not all recover the same thing: some reconstruct a private key or seed, some repair a threshold role, some rotate account control, and some extract artifacts after an incident. We therefore compare each design using four recovery-semantics dimensions: the *recovered object*, the *failure event*, the *authorization path*, and the *post-recovery state*. These dimensions describe what is restored, why recovery is invoked, what evidence or actors authorize it, and what control relation remains afterward. The adversary classes and recovery properties at the end of this section are supporting evaluation lenses, not additional parts of the four-part framework.

5.1 Recovered object

Recovery may reconstruct an original private key or seed, restore a share or signing role, restore account control or asset ownership, or extract forensic artifacts. Mechanisms include password-protected shares, threshold reconstruction, deterministic regeneration, biometric or physical

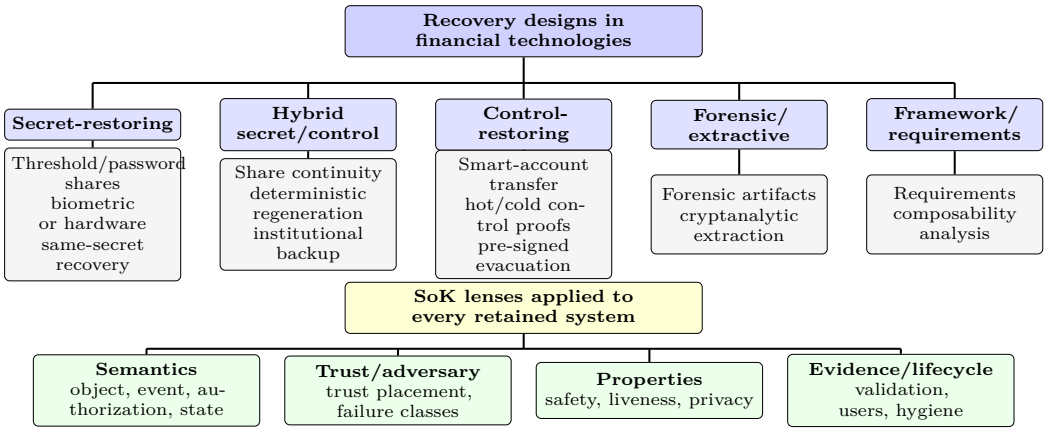


Figure 2 Roadmap of the SoK taxonomy and lenses. The top tree groups recovery designs by the promise they make; the bottom lenses are applied to every retained system in the matrix and used in the tradeoff analysis.

regeneration, hardware-assisted retrieval, contracts, fallback keys, backup authenticators, pre-signed transactions, and ownership proofs. Table 8 lists retained studies by recovery axis, the full version provides the complete row-level coding, and Table 9 gives representative mechanism examples.

5.2 Failure event

The corpus covers several failure modes: loss of a local key, seed, device, mnemonic, or share; unavailability of threshold participants or custody roles; loss or compromise of smart-account owners; failed identity-wallet migration or backup restoration; and post-incident forensic recovery. Table 10 summarizes how these failure modes map to adversary classes. A recurring limitation is that many designs focus on loss but give less detail for compromise, coercion, malware, or malicious recovery helpers.

5.3 Authorization path

Recovery is authorized by different evidence of legitimacy. Depending on the family, authorization may require enough shares or servers, a memorized or factor-derived credential, retained root material, biometric helper data, guardian approval, contract timeouts and challenges, pre-authorized transactions, organizational procedures, hardware controls, or custody roles. The authorization basis matters because it determines who can approve recovery, who can block it, and what evidence remains after the event.

5.4 Post-recovery state

The post-recovery state is often under-specified. Some protocols return the same secret and continue with unchanged keys [52, 108]; some repair or refresh shares while preserving threshold signing authority [70, 59]; some restore access by rotating credentials or moving control to a new key [110, 22]; and some transfer assets away from a failing account [71, 58]. For financial technologies, this distinction is material: a user, exchange, or smart account that rotates ownership after recovery has a different audit and dispute problem from one that reconstructs the old private key.

5.5 Supporting adversary vocabulary

We also normalize the threat vocabulary used across papers, because terms such as "lost key," "recovery," and "compromise" otherwise mix different financial risks. We distinguish eight recurring classes. *Loss* means the legitimate party no longer has usable access to a key, seed, credential, device, share, passkey, or signer. *Compromise or theft* means that the material may be copied, stolen, malware-exposed, or attacker-controlled. *Coercion* means that a user, guardian, provider operator, or institutional signer is pressured to approve recovery or disclose recovery material. A *malicious helper* is a guardian, shareholder, trustee, server, or custody participant who deviates from the recovery protocol alone or in collusion. *Server or provider abuse* covers wallet providers, cloud backup services, identity verifiers, recovery providers, HSM operators, and infrastructure services that misuse their role or metadata. An *institutional insider* is an organizational actor such as an administrator, treasurer, support agent, compliance role, or recovery officer who misuses delegated recovery authority. *Unavailability* means that a user, helper, server, blockchain, signer, network, device, or institution cannot participate when needed. A *forensic adversary* attempts to extract wallet artifacts, keys, backups, traces, or evidentiary control material from devices or services.

5.6 Recovery-specific properties

Finally, we evaluate each design through three recovery-specific properties. *Safety* means unauthorized parties cannot trigger recovery, reconstruct a secret, rotate control, evacuate assets, or exploit recovery metadata to steal financial control. *Liveness* means a legitimate claimant can complete recovery after loss, participant churn, device replacement, institutional outage, or disaster without relying on a single fragile actor. *Privacy* means the recovery process does not unnecessarily reveal helper relationships, provider choices, account linkage, recovery attempts, identity-proofing material, device replacement, or forensic traces.

6 Taxonomy of Key Recovery Mechanisms

Table 9 is axis-first: mechanism families are nested under the recovery semantics they primarily realize. This avoids grouping systems once by cryptographic primitive and again by design family. The primary recovery axis is mutually exclusive and assigned by the final recovery promise and post-recovery state recorded in the master matrix. Mechanism families remain non-exclusive because some systems combine mechanisms, such as threshold reconstruction inside a hardware wallet or backup-authenticator recovery inside a smart account. Citation columns are representative; the full version provides exhaustive row-level coverage.

The primary split is between *secret-restoring* and *control-restoring* recovery. Secret-restoring mechanisms preserve the old control relation by reconstructing or regenerating the same key, seed, or share. Control-restoring mechanisms intentionally create a new control relation, for example by moving an account owner, registering a backup authenticator, executing a pre-signed transaction, or proving ownership to migrate assets. Hybrid systems combine both promises, such as restoring a share while also preserving institutional signing authority or social-account control. Control-restoring designs are not weaker merely because they do not reconstruct the original secret; they make a different promise and need different post-recovery checks. Figure 2 summarizes the taxonomy spine and the lenses applied to each retained system.

Table 9 Axis-first taxonomy and design comparison. Mechanisms that reconstruct a secret, rotate control, move assets, or extract evidence make different recovery promises.

Axis	Mechanism family	What recovery restores	Enrollment or storage pattern	Examples
Secret-restoring	Threshold, password-protected, and factor-assisted reconstruction	Same key, seed, mnemonic fragment, or share	Distributed shares, password/factor material, protected server shares, backup fragments	[52, 98, 108, 53, 100, 66, 55, 105, 101, 10]
Secret-restoring	Memory-based, biometric, fuzzy, and hardware-assisted regeneration	Same or equivalent wallet-control secret	Memory artifact, helper data, biometric source, secure element, HSM, or TEE	[82, 73, 65, 88, 51, 48, 72, 33]
Hybrid secret/control	Share continuity, social/delegated recovery, biometric helper-key regeneration, and institutional disaster recovery	Secret material plus signing role, custody role, account access, or wallet-governance authority	Committees, guardians, biometric helper keys, organizational media, custody procedures, or DID backup stores	[42, 70, 7, 79, 59, 95, 34, 78, 60]
Control-restoring	Deterministic hot/cold signing-authority recovery	Future signing authority after hot-key compromise, without reconstructing the compromised hot key	Cold-wallet seed or master state, derivation path, and synchronized wallet state	[19]
Control-restoring	Smart-contract transfer, fallback signing, passkey recovery, credential re-issuance, and ownership proof	Account owner, signer, backup credential, ownership proof, or control path	On-chain state, fallback keys, backup authenticators, challenge windows, re-issuance, or proofs	[110, 103, 17, 13, 5, 21, 22, 96, 94]
Control-restoring	Asset evacuation and pre-authorized transfer	Value or asset-control relation moved away from a failed path	Pre-signed transactions, emergency controls, provider state, or encrypted transfer material	[71, 58]
Forensic/extractive	Forensic artifact recovery and cryptanalytic extraction	Wallet artifacts, credentials, keys, or evidentiary traces	Device access, backups, memory, nonce failures, investigation workflow, or seized evidence	[16, 62, 44, 49, 50]
Framework/requirements	Requirements and composability analysis	No single recovered object; compares obligations or composed mechanisms	Requirements model, key-person analysis, or formal composition model	[77, 12]

The taxonomy also separates *cryptographic trust distribution* from *operational trust distribution*. Threshold systems distribute cryptographic material but still need participant governance, refresh, authentication, and availability. Hardware systems reduce software exposure but concentrate trust in devices, attestation, or recovery artifacts. Social recovery and smart contracts reduce single-custodian dependence but add guardian, metadata, liveness, and irreversibility risks.

7 Mechanism-Family Tradeoff Analysis

Using the master matrix provided in the full version and the axis-first taxonomy, we compare safety and liveness surfaces, post-recovery hygiene and metadata privacy, and validation evidence. Table 9 gives the axis-level mechanism comparison, Table 10 adds the adversary lens, and Table 11 summarizes the obligations that follow after recovery. The family tables are cross-cutting lenses rather than primary-axis reclassifications; each cited system keeps the canonical axis assigned in the master matrix.

Table 10 Family-by-adversary comparison. Actors who improve liveness can also become collusion, coercion, provider, insider, or metadata surfaces.

Family	Loss / unavailability	Compromise / theft	Coercion / malicious helpers	Provider, insider, metadata, or forensic exposure	Examples
Threshold and password-protected reconstruction	Lost keys, shares, or passwords when enough shares/servers remain live	Exposed shares, weak passwords, or stale fragments can reconstruct	Collusion or coercion crossing the threshold	Server contact, helper membership, and recovery attempts may leak; hosted shares add provider trust	[52, 98, 108, 53, 100, 61, 55, 105, 101, 59]
Deterministic, biometric, and hardware recovery	Device, seed, mnemonic, signer, or factor loss when source material remains available	Restored seeds, helper keys, or cold-wallet authority may preserve attacker access unless old paths are quarantined	Biometrics, memories, devices, or support flows can be coerced	Vendors, attestation, HSMs, helper data, and identity checks become trust/privacy points	[19, 82, 73, 65, 88, 51, 48, 72, 33, 78, 11]
Social, guardian, and smart-account transfer	Lost owners, passkeys, or signers when guardians, fees, and windows are available	Compromised guardians, fallback signers, modules, or passkeys can rotate control	Vote buying, social engineering, recoverer abuse, and guardian collusion	On-chain proposals, guardian sets, and signer rotations can reveal distress and relationships	[110, 103, 17, 13, 7, 5, 71, 79, 21, 22, 96, 94, 90, 12, 58]
Institutional custody and disaster recovery	Signer churn, lost institutional devices, outages, and disasters	Compromised operator devices, HSM credentials, backups, or policy engines can move funds	Collusion or coercion of officers can satisfy dual-control policies	Administrators, recovery officers, compliance roles, logs, and drills expose insider/metadata surfaces	[109, 77, 102, 95, 60]
Forensic and extractive recovery	Inaccessible devices, backups, memory, nonce failures, or investigation evidence	The same extraction path can be abused by attackers or overbroad investigators	Coercion may compel device access, backup disclosure, or support escalation	Forensic traces are often the recovered object; chain-of-custody and privacy risks are intrinsic	[16, 62, 44, 49, 50]

The first comparison axis is *safety*: who can cause unauthorized recovery? Threshold systems bound exposure by shares or servers but depend on enrollment, participant independence, and refresh; smart-account and guardian systems avoid custodial key storage but add vote-buying, social-engineering, coercion, and challenge-window risks; hardware designs isolate secrets but depend on device, attestation, backup, or HSM assumptions. The second axis is *liveness*: threshold, contract, biometric, memory-based, and institutional systems can all fail when helpers, fees, inputs, terminals, or policies are unavailable. The third axis is *evidence*: the corpus validates feasibility and performance more often than recovery under stressful, high-value conditions.

Table 11 Post-recovery hygiene and metadata/privacy obligations by mechanism family.

Family	Post-recovery hygiene	Metadata/privacy exposure	Common gap	Examples
Threshold and password-protected reconstruction	Rotate/rekey if compromise is possible; revoke stale fragments; refresh shares; audit helpers/servers	Helper linkage, server contact, timing, and repeated use can reveal distress or linkage	Old-secret reconstruction often lacks rotation, revocation, or dispute handling	[52, 98, 108, 53, 100, 61, 55, 105, 101, 59]
Deterministic, biometric, and hardware recovery	Revoke backups/helper data; reset authenticators; quarantine lost devices; migrate funds or rotate control if old material may still authorize value	Device replacement, helper data, vendor contact, identity checks, and account linkage expose context	Many designs lack a decision rule for continued use of restored material or retirement of old authority	[19, 82, 73, 65, 88, 51, 48, 72, 33, 78, 11]
Social, guardian, and smart-account transfer	Revoke old owners/passkeys; refresh guardians; review delays/cancellation; audit final owner state	Guardian relationships, proposals, signer rotations, challenge windows, and module state reveal attempts	Guardian reset, cancellation, rollback, and dispute handling are uneven	[110, 103, 17, 13, 7, 5, 71, 79, 21, 22, 96, 94, 90, 12, 58]
Institutional custody and disaster recovery	Rotate signer/API/HSM credentials and policy roles; preserve logs; quarantine affected devices	Approver graphs, provider contacts, audits, drills, KYC, insurance, and legal holds reveal operations	Governance, notification, and disputed-instruction workflows are less uniform	[109, 77, 102, 95, 60]
Forensic and extractive recovery	Move funds or rotate keys; revoke recovered credentials; maintain chain-of-custody; quarantine devices/secrets	Device images, backups, artifacts, clusters, investigator actions, and asset movement are metadata-rich	Legitimate recovery and adversarial extraction share tooling, so authority boundaries matter	[16, 62, 44, 49, 50]

8 Generalized Construction Model

Across the taxonomy, recovery designs can be described by a common construction model. It is not a protocol template; it is a checklist of design obligations that recur across wallets, custody systems, identity wallets, and smart accounts.

1. **Control material.** The system identifies what is at risk: a private key, seed, key share, recovery key, credential, signing role, account owner, or asset-control relation.
2. **Enrollment.** The system creates recovery material before failure, such as secret shares [98, 108, 66], masked password-protected shares [52, 53, 80], deterministic seeds [19, 23, 48], biometric helper data [51, 72, 78], hardware-wrapped secrets [82, 65, 33], guardians [42, 17, 79], backup authenticators [21, 94, 11], or pre-signed transactions [71, 58].
3. **Storage and governance.** Recovery material is placed with the user, helpers, servers, institutions, smart contracts, secure hardware, cloud providers, or forensic evidence stores. This step determines whether the design is non-custodial, threshold-custodial, institutionally governed, or provider-assisted.
4. **Failure trigger.** Recovery begins after loss, unavailability, compromise, migration, death, device replacement, identity-wallet restoration, or investigation. The trigger may be user-initiated, contract-triggered, institutional, or forensic.
5. **Authorization and checks.** The system verifies that the recovery claimant is legitimate. Authorization can be cryptographic, social, institutional, on-chain, biometric, hardware-backed, or evidentiary. Strong designs separate authorization from mere possession of backup data.

6. **Recovery action.** The system reconstructs a secret, regenerates a derived key, repairs a share, reissues a credential, rotates ownership, transfers assets, or extracts an artifact.
7. **Post-recovery hygiene.** The system should decide whether to rotate keys, invalidate stale shares, revoke old authenticators, refresh guardians, record an audit event, or migrate funds. Many retained studies do not fully specify this stage, which is a central gap for financial deployments.

This construction model explains why apparently similar mechanisms differ in financial risk. A threshold wallet that reconstructs the old seed [108, 105, 101] exposes the recovered secret after completion. A smart-account recovery protocol that rotates ownership [110, 103, 22] avoids exposing the old key but must secure the transfer process. A pre-signed evacuation mechanism [71, 58] may preserve asset value while abandoning the old account. The full version walks four representative systems through all seven stages to show where their post-recovery hygiene obligations diverge. These semantics should not be evaluated under a single "key recovery" label.

9 Production Recovery Designs

The 77-paper synthesis corpus remains the peer-reviewed evidence base for this SoK. Deployed wallet and custody systems are nevertheless useful practice checks: they are not counted as corpus rows, but they show whether the taxonomy and construction model cover recovery semantics used by retail users, institutional customers, and smart-account developers. Table 12 classifies representative systems using official documentation.

Representative production recovery workflows

The seven-stage model separates production systems that all call themselves "recovery." Ledger Recover restores seed-equivalent material through device-approved fragmentation, provider-held encrypted fragments, and identity verification. Safe{RecoveryHub} stores recoverer and delay state on-chain so a recoverer can propose owner or threshold changes unless current signers cancel. MPC custody restores a share or signer path across customer devices, provider infrastructure, passphrases, and policy. Passkey smart wallets add or replace a signer, after which stale passkeys and recovery phrases remain asset-control material.

10 Cross-Cutting Findings for Financial Technologies

10.1 Finding 1: Recovery semantics are heterogeneous

Recovery may mean reconstructing a key, regenerating a seed, restoring a share, recovering a recovery key, rotating credentials, restoring account control, transferring assets, or extracting forensic artifacts. The largest recovered-object groups are original private keys (34 studies), signing authority (20), account control (19), seeds or root secrets (15), asset ownership or transfer paths (13), and key shares (10). Counts are non-exclusive, but the design implication is clear: systems that restore account control should not be evaluated as if they reconstruct the original key.

10.2 Finding 2: Non-custodial recovery still moves trust somewhere

In this corpus, recovery mechanisms reduce or redistribute trust rather than eliminating it. Password-protected systems trust server independence and rate limiting; threshold systems trust participant diversity and share availability; hardware systems trust devices, attestation,

Table 12 Production and standards-based practice checks outside the 77-paper corpus.

Design	Evidence source	Recovery axis and mechanism	Recovery action	Main SoK implication
Ledger Recover	Vendor product docs	Secret-restoring; secure-element export, encrypted fragments, HSM-backed providers	Identity checks release fragments to reconstitute seed entropy inside a Ledger secure element [64]	Restores seed-equivalent material; moves trust to identity proofing, firmware consent, providers, and HSMs
MPC custody	Vendor developer and help docs	Hybrid secret/-control; MPC share backup, signer reprovisioning, disaster recovery	Fireblocks and Coinbase Prime describe share/signer recovery through provider infrastructure, devices, passphrases, and policy [40, 39, 14]	Threshold cryptography still needs governance for passphrases, devices, signers, and disaster drills
Safe {RecoveryHub} and modules	Safe documentation	Control-restoring; delayed recoverer module	A Recoverer proposes owner or threshold changes; signers can cancel during the delay [83, 84]	Smart-account recovery migrates control and exposes on-chain recovery metadata
ERC-4337 / EIP-7702	Standards and protocol docs	Control-restoring substrate; custom validation and EOA delegation	ERC-4337 supports custom account validation; EIP-7702 lets EOAs delegate to code using authorization tuples [26, 25, 27]	Account abstraction expands recovery design space; EIP-7702 leaves the EOA key as a bypass risk
WebAuthn/ passkeys	W3C, FIDO, and product docs	Control-restoring; backup credential or passkey signer replacement	WebAuthn exposes backup state; Coinbase Smart Wallet uses a recovery phrase to create a new signer and passkey [99, 36, 15]	Passkey recovery depends on credential-provider recovery, fallback policy, and signer cleanup
Argent guardian recovery	Wallet support docs	Control-restoring; guardian social recovery with delay	Guardians approve recovery; Argent Vault recovery has majority approval and a 48-hour delay [2, 1]	Social recovery improves liveness but creates guardian availability, coercion, and relationship-metadata questions

or HSM boundaries; social recovery trusts guardians; and smart-contract recovery trusts public rules and challenge windows. In the trust and adversary fields, collusion appears in 22 rows, malicious actors in 20, server terms in 20, provider terms in 8, and insider terms in 8. These are conservative text-coded signals rather than a threat prevalence measure, but Table 10 makes the trust shift explicit. A useful recovery design should state where trust has moved.

10.3 Finding 3: Liveness improvements often create abuse paths

Adding helpers, backup authenticators, pre-signed transactions, cloud backups, or institutional procedures can prevent permanent loss while creating coercion, collusion, phishing, insider, or unilateral-transfer paths. Unavailability appears in 16 failure-event rows, while compromise or theft appears in 16 rows; the adversary text mentions collusion in 22 rows, malicious actors in 20, and coercion in 8. This tension appears across social recovery [42, 17, 79], smart-account transfer [110, 103, 5, 22], institutional custody [77, 95], and hosted or provider-assisted identity recovery [102, 94, 34]. Because coercion is rarely formalized, the observed coercion count is best read as a lower bound.

10.4 Finding 4: Post-recovery lifecycle is a common blind spot

Many studies explain how to recover but not what happens next: whether to rotate keys, refresh shares, revoke helpers, clear backup authenticators, audit the event, quarantine a device, migrate funds, or handle disputes. The post-recovery-state coding is dominated by same-or-equivalent secret restoration (46 rows), followed by new control relations or credentials (17), forensic extraction (4), operational access restored or rekeyed (3), analysis-only rows (3), asset movement (2), and share or committee refresh (2). Table 11 shows that same-secret systems need rotation or migration decisions, guardian systems need owner and helper reset, institutional systems need audit and policy cleanup, and forensic systems need chain-of-custody and authority boundaries.

10.5 Finding 5: User evidence lags behind protocol evidence

Formal proofs, prototypes, and benchmarks are common, but user-study evidence is rare. The extraction records 56 prototypes, 50 benchmarks, 28 formal proofs, 17 simulations, 11 security-game analyses, 9 case studies, 7 gas-cost measurements, and 6 rows with no evaluation in the extracted evidence. Only three retained studies include user-study evidence [98, 88, 90], and one forensic tool reports operational deployment evidence [50]. Because recovery is invoked under stress, device loss, death, institutional disruption, or suspected compromise, we treat usability claims without user data as claims, not as user evidence.

10.6 Finding 6: Recovery metadata is underprotected

Several designs protect key material while saying less about metadata: which guardians exist, which servers were contacted, whether a recovery attempt occurred, which device was replaced, which account is being migrated, or which institution is under stress. Privacy or private-state terminology appears in 51 privacy-discussion rows, but 52 rows also include limited, no-formal-analysis, not-stated, or out-of-scope caveats; metadata, contact, timing, log, linkage, trace, or surveillance terms appear in 33 rows. These counts reflect conservative coding of what papers report, not proof that every omitted discussion leaks data. The issue cuts across threshold servers [52, 61, 105], guardians [42, 79], smart accounts [103, 5, 22], identity wallets [59, 94, 34], and forensic tools [44, 50].

11 Financial, Regulatory, and Human-Factors Implications

Custody, compliance, and operational controls

Recovery becomes a regulated financial-technology concern when a provider holds, administers, restores, or can influence access to crypto-assets. MiCA-style custody rules, FATF virtual-asset guidance, and DORA operational-resilience obligations bring custody policy, client authentication, records, incident recovery, and provider duties into the design surface [32, 29, 37, 30]. This paper is not a legal analysis; the design point is that each mechanism should state whether recovery is self-custodial, provider-assisted, sub-custodial, institutional, or only a user-held backup. Obligations change depending on who verifies identity, sees recovery attempts, controls logs, and can block or reverse a disputed recovery [109, 77, 102, 95, 60].

Travel Rule metadata

Recovery designs that transfer assets, restore hosted access, or involve a crypto-asset service provider can also trigger Travel Rule information flows. FATF treats the Travel Rule as the virtual-asset application of Recommendation 16, requiring originator and beneficiary information to accompany covered transfers, while the EU implements parallel requirements for transfers of funds and certain crypto-assets in Regulation 2023/1113 and EBA guidelines [37, 38, 31, 28]. For recovery, the design implication is that identity proofing, beneficiary changes, emergency transfers, provider logs, and self-hosted-address interactions may become compliance metadata. Systems should therefore separate recovery authorization from Travel Rule reporting, minimize unnecessary recovery metadata, and state who can disclose records during disputes or investigations.

Economics and incentives

Recovery is also an incentive system. Guardians, shareholders, providers, support agents, and institutions may be unpaid, compensated, bonded, insured, or penalized; attackers can exploit those incentives through bribery, coercion, fee griefing, or collusion. Rational secret-sharing rows model some behavior [105, 101], composability work makes rational adversaries explicit [12], and pre-signed evacuation exposes fee drift and stale-UTXO issues [58]. Designs should specify who pays recovery fees, who absorbs disputed losses, and whether insurance or reimbursement changes diligence.

Inheritance and death or incapacity

Death and incapacity are not ordinary key-loss events: authority may shift to an heir, executor, trustee, provider, court, or pre-authorized beneficiary. The corpus contains only a small inheritance thread–key-person loss [77], biometric/heir wallet governance [78], re-registration recovery [18], and pre-signed recovery for owner disappearance or death [58]. Estate recovery therefore needs separate treatment for enrollment while alive, beneficiary changes, disputed claims, privacy, and post-transfer auditability.

Post-quantum migration

NIST finalized the first three post-quantum cryptography FIPS in 2024 and selected HQC for future standardization as a backup key-establishment algorithm in 2025 [75, 76]. Recovery material is migration-sensitive because wallet seeds, encrypted mnemonics, guardian secrets, recovery tokens, custody shares, and pre-signed emergency transactions may outlive active signing keys. Same-secret restoration can reintroduce classical ECDSA or Schnorr keys after

migration, so recovery should bind restored material to algorithm/version metadata and require re-derivation, fund migration, or key rotation when old material controls quantum-vulnerable addresses. Control-restoring recovery must update every path, not only the active signer: stale EOAs, guardians, fallback keys, passkeys, authorizations, and pre-signed transactions can remain bypasses. Institutional and forensic recovery also need crypto-agile policies for shares, approvals, HSM rules, audit signatures, and evidentiary records. The corpus has early signals [13, 48, 96, 94, 11], but no complete migration playbook for deployed wallets or custody systems.

Table 13 User-study evidence in the retained corpus. Existing studies test tangible shares, memory cues, and multi-factor wallet recovery, but not high-stress, adversarial, estate, or broad-population recovery.

Study	Participants and task	Main finding	Main limitation for recovery claims
Offline secret sharing [98]	24-person form-factor survey; 18 completed field-study participants distributed five physical shares to 90 confidants and later retrieved enough shares for a 3-of-5 threshold	Physical share distribution and retrieval was practical, but recovery coordination could take days or weeks; paper versus plastic key tags did not dominate success	Student sample, one threshold configuration, simulated cryptographic reconstruction, and limited compliance checking
Reminisce [88]	51 participants/510 pictures for parameters, 105-person survey, and 20 wallet-experienced users who repeated picture-location recovery after one month	All 20 recovered successfully; 18 succeeded on the first attempt, with 1.10–8.29 minute recovery times and 2.56 minutes average	Repository security and authentication out of scope; fixed parameters; 20–40-year-old experienced wallet users; photo/location privacy leakage
Multi-factor key-derivation wallet [90]	27 United States participants compared custodial, conventional non-custodial, and MFKDF-based Ethereum wallet prototypes across creation, login, and recovery tasks	The MFKDF wallet reported 37% higher System Usability Scale (SUS) and 71% faster task completion than the conventional non-custodial wallet, with less seed-phrase handling	Small US-only and not gender-balanced sample; prototype fidelity limits; accessibility and marginalized populations not evaluated

Together, these studies show that tangible shares, picture-location memory, and familiar multi-factor workflows can work in controlled settings, but not how users behave under theft, grief, coercion, provider lockout, legal dispute, or market stress. Open questions include ceremony comprehension, stress errors, guardian coordination, inheritance, phishing, support escalation, accessibility, and recovery-metadata privacy.

12 Open Problems and Research Agenda

- **Standard recovery semantics.** Wallets, custody systems, and identity wallets need explicit terminology for original-key reconstruction, seed regeneration, share repair, credential re-issuance, control transfer, and asset evacuation. Without this, designs with different promises are difficult to compare.

- **Custody, regulation, and incentives.** Recovery mechanisms should declare whether they create self-custody, provider-assisted custody, sub-custody, or institutional obligations, and should model guardian availability, provider fees, Travel Rule metadata, reimbursement, insurance, fee drift, and collusion [105, 101, 12, 58, 32, 29, 37, 31, 30].
- **Inheritance and estate recovery.** Death and incapacity require different authorization semantics from ordinary loss, including beneficiary enrollment, revocation while alive, dispute handling, privacy, and auditability for heirs, executors, trustees, and providers [77, 78, 18, 58].
- **Composable recovery after loss or compromise.** Smart-account recovery increasingly combines guardians, backup authenticators, fallback signatures, timeouts, ownership proofs, and pre-signed transactions [103, 13, 21, 96, 58]. Research is needed on how these paths compose, and on recovery that rotates, revokes, audits, and disputes after compromise rather than only restoring after loss [52, 72, 9].
- **Privacy-preserving recovery coordination.** Recovery should hide helper relationships, server choices, attempts, and account linkage where possible, especially for threshold servers, guardians, identity wallets, and smart-contract metadata.
- **Human-centered recovery under stress.** Recovery ceremonies should be evaluated with users and operators under time pressure, grief, device loss, or fear of theft. Existing user evidence is useful but too sparse and controlled for the importance of the problem [98, 88, 90].
- **Operational threshold custody.** Threshold protocols need lifecycle support for lost shares, changing personnel, institutional disaster recovery, participant churn, and auditability [70, 59, 95].
- **Long-lived recovery material.** Wallet seeds, backup credentials, custody shares, and pre-signed recovery paths may survive for years. Post-quantum recovery is emerging [48, 94, 75], while forensic recovery raises privacy, chain-of-custody, and abuse questions [62, 44, 49, 50].

13 Ethics, Limitations, and Reproducibility

This SoK studies mechanisms that can restore or transfer financial control. The ethical risk is therefore not limited to confidentiality of keys. Recovery systems can enable theft, coercion, unauthorized transfer, insider abuse, account lockout, or exclusion from identity and financial services. For this reason, the taxonomy distinguishes what is recovered and who authorizes recovery; ambiguity in these two points can hide serious harms.

The review has methodological limitations. The search considered peer-reviewed journal and conference articles written in English and classified as computer science. The final synthesis retained 77 of the 118 discovery papers after full-text extraction and financial-technology-focused filtering. LLM assistance was used as a decision-support tool during screening, and all LLM-assisted decisions were manually verified. The resulting corpus is suitable for a focused SoK, but it does not exhaust gray literature, wallet documentation, standards drafts, or production incidents.

The extraction is also limited by the evidence reported in each paper. If a study did not state a deployment, user evaluation, privacy analysis, or post-recovery rotation procedure, we classify that evidence as absent rather than infer it. This conservative choice may understate properties implemented outside the paper, but it avoids giving recovery mechanisms credit for unstated guarantees.

For reproducibility, the paper reports the source databases, exact logical query, search/export date, database counts, de-duplication and screening counts, grouped exclusion reasons, snowballing rounds, QA threshold, retained-study count, and extracted comparison dimensions. The reproducibility package contains search-count summaries, screening-round summaries, the LLM screening protocol and prompts, exclusion-reason summaries, QA counts, extracted rows, the matrix codebook, count audits, venue/year confidence analyses, and the machine-readable 77-system master matrix. Copyrighted PDFs, raw library exports, private credentials, repository history, and private working materials are excluded from the package.

Generative-AI tools were used to support title and abstract screening and to assist with language and structural editing of the manuscript. All screening decisions and all AI-assisted manuscript content were reviewed and verified by the authors. The authors retained full responsibility for the extracted evidence, numerical counts, citations, interpretations, and final manuscript. Generative-AI tools were not treated as authoritative sources.

14 Related Work

Table 1 compares the closest external SoKs and surveys. Broad cryptocurrency and DeFi SoKs organize protocols, applications, attacks, and economic risk [98, 72, 69], while wallet-focused surveys organize protocols, authentication factors, attack surfaces, incidents, and defenses [54, 89, 35, 100]. These works establish the broader security context but do not use recovery semantics – what object or control relation is restored, who authorizes it, and what state remains afterward – as the primary unit of comparison.

The retained corpus connects bodies of work usually treated separately, including threshold reconstruction, factor- and hardware-assisted recovery, social and smart-account control transfer, decentralized-identity backup, institutional disaster recovery, and forensic extraction. This SoK makes these semantics auditable in a system-level matrix and distinguishes secret reconstruction from control restoration, signing-authority migration, credential re-issuance, asset evacuation, and forensic extraction. That distinction drives the taxonomy, construction model, and agenda.

15 Conclusion

Cryptographic recovery in financial technologies is broader than private-key backup. The retained corpus shows mechanisms for reconstructing secrets, regenerating wallet seeds, repairing shares, restoring credentials, delegating recovery, rotating account ownership, evacuating assets, and extracting wallet artifacts. A wallet that reconstructs a seed, a custody protocol that repairs a signing share, a smart account that rotates ownership, and a forensic tool that extracts artifacts are not making the same promise. Recovery-aware financial systems need explicit recovery semantics to compare safety, liveness, privacy, usability, and trust without creating new paths for theft, coercion, surveillance, or exclusion.

References

1 Argent. How to add a guardian to your argent ethereum wallet, 2025. Accessed: 2026-05-26. URL: https://support.argent.xyz/hc/en-us/articles/360008013258-How-to-add-a-guardian-to-your-Argent-Ethereum-wallet.

2 Argent. How to recover my wallet with guardians onchain, 2025. Accessed: 2026-05-26. URL: https://support.argent.xyz/hc/en-us/articles/360007338877-How-to-recover-my-wallet-with-guardians-onchain-complete-guide.

3 Soumya Basu, Alin Tomescu, Ittai Abraham, Dahlia Malkhi, Michael K. Reiter, and Emin Gün Sirer. Efficient verifiable secret sharing with share recovery in bft protocols. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, page 2387–2402. ACM, 2019. doi:10.1145/3319535.3354207.

4 Michele Battagliola, Alessio Galli, Riccardo Longo, and Alessio Meneghetti. A provably-unforgeable threshold schnorr signature with an offline recovery party. *CEUR Workshop Proceedings*, 3166:60–76, 2022. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85134323353&partnerID=40&md5=66337d80bb6e1705175aa7285cda93d4.

5 Sam Blackshear, Konstantinos Chalkias, Panagiotis Chatzigiannis, Riyaz Faizullabhoy, Irakliy Khaburzaniya, Eleftherios Kokoris Kogias, Joshua Lind, David Wong, and Tim Zakian. Reactive key-loss protection in blockchains. In *Financial Cryptography and Data Security. FC 2021 International Workshops*. Springer, 2021. doi:10.1007/978-3-662-63958-0_34.

6 Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. SoK: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy*, pages 104–121. IEEE, 2015. doi:10.1109/SP.2015.14.

7 Micha Boro and Anna Kobusiska. Alternative authentication with smart contracts for online games. In *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, pages 415–418, 2021. doi:10.1109/LCN52139.2021.9524886.

8 Florian Breuer, Vipul Goyal, and Giulio Malavolta. Cryptocurrencies with security policies and two-factor authentication. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 140–158, 2021. doi:10.1109/EuroSP51992.2021.00020.

9 Syeda Tayyaba Bukhari, Muhammad Umar Janjua, and Junaid Qadir. Secure storage of crypto wallet seed phrase using ecc and splitting technique. *IEEE Open Journal of the Computer Society*, 2024. doi:10.1109/OJCS.2024.3398794.

10 Haewon Byeon, Aadam Quraishi, Praveen Thuniki, Ismail Mohamed Keshta, Mukesh Soni, and Mohammad Shabaz. Optimized security mechanism for publicly secret key sharing over cloud using blockchain. *CEUR Workshop Proceedings*, 3774:137–150, 2024. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85206453527&partnerID=40&md5=80d82f75bfa4de5397d86131d6d70b38.

11 Bertrand Francis Cambou and Mahafujul Alam. Challengeresponse pair mechanisms and multi-factor authentication schemes to protect private keys. *Applied Sciences (Switzerland)*, 15, 2025. doi:10.3390/app15063089.

12 Panagiotis Chatzigiannis, Ke Coby Wang, Sunpreet Singh Arora, and Mohsen Minaei. A composability analysis framework for web3 wallet recovery mechanisms. *Proceedings - IEEE Symposium on Security and Privacy*, pages 1531–1546, 2025. doi:10.1109/SP61157.2025.00158.

13 David Chaum, Mario Larangeira, Mario Yaksetig, and William Carter. W-ots$^+$ up my sleeve! a hidden secure fallback for cryptocurrency wallets. In *Applied Cryptography and Network Security*. Springer, 2021. doi:10.1007/978-3-030-78372-3_8.

14 Coinbase. How to secure your onchain wallet, 2026. Accessed: 2026-05-26. URL: https://help.coinbase.com/en/prime/onchain-wallet/how-to-secure-your-onchain-wallet.

15 Coinbase. Recover your smart wallet, 2026. Accessed: 2026-05-26. URL: https://help.coinbase.com/en/wallet/getting-started/smart-wallet-recovery.

16 Nicolas T. Courtois and Rebekah Mercer. Stealth address and key management techniques in blockchain systems. In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - ICISSP*. SciTePress, 2017. doi:10.5220/0006270005590566.

17 Weiqi Dai, Yan Lv, Kim-Kwang Raymond Choo, Zhongze Liu, Deqing Zou, and Hai Jin. Crsa: A cryptocurrency recovery scheme based on hidden assistance relationships. *IEEE Transactions on Information Forensics and Security*, 16:4291–4305, 2021. doi:10.1109/TIFS.2021.3104142.

18 Weiqi Dai, Yang Zhou, Xiaohai Dai, Kim-Kwang Raymond Choo, Xia Xie, Deqing Zou, and Hai Jin. Cr-dap: A comprehensive and regulatory decentralized anonymous payment system. *IEEE Transactions on Information Forensics and Security*, 20:3274–3286, 2025. doi:10.1109/TIFS.2025.3550821.

19 Poulami Das, Sebastian Faust, and Julian Loss. A formal treatment of deterministic wallets. In *CCS '19*, 2019. doi:10.1145/3319535.3354236.

20 Cristfol Daudn-Esmel, Jordi Castell-Roca, Alexandre Viejo, and Ignacio Miguel-Rodrguez. Multi-platform wallet for privacy protection and key recovery in decentralized applications. *Blockchain: Research and Applications*, 6:100243, 2025. doi:10.1016/j.bcra.2024.100243.

21 Varun Deshpande, Atharva Vijay Khade, Harish J, Gandhi Kishor Addanki, and Srivatsan Deenadayalan. A passkey based recovery mechanism for blockchain hardware wallets. In *2024 IEEE/ACM 17th International Conference on Utility and Cloud Computing (UCC)*, pages 276–281, 2024. doi:10.1109/UCC63386.2024.00046.

22 Varun M. Deshpande, J. Harish, and Atharva Vijay Khade. A practical recovery mechanism for blockchain hardware wallets. *IEEE Access*, 12:169928–169944, 2024. doi:10.1109/ACCESS.2024.3497302.

23 Adriano Di Luzio, Danilo Francati, and Giuseppe Ateniese. Arcula: A secure hierarchical deterministic wallet for multi-asset blockchains. In *Cryptology and Network Security*, 2020. doi:10.1007/978-3-030-65411-5_16.

24 Yimika Erinle, Yathin Kethepalli, Yebo Feng, and Jiahua Xu. SoK: Design, vulnerabilities, and security measures of cryptocurrency wallets. *Computer Networks*, 273:111691, 2025. doi:10.1016/j.comnet.2025.111691.

25 Ethereum Improvement Proposals. EIP-7702: Set code for EOAs, 2026. Accessed: 2026-05-26. URL: https://eips.ethereum.org/EIPS/eip-7702.

26 Ethereum Improvement Proposals. ERC-4337: Account abstraction using alt mempool, 2026. Accessed: 2026-05-26. URL: https://eips.ethereum.org/EIPS/eip-4337.

27 ethereum.org. Pectra 7702 guidelines, 2026. Accessed: 2026-05-26. URL: https://ethereum.org/roadmap/pectra/7702/.

28 European Banking Authority. Guidelines on information requirements in relation to transfers of funds and certain crypto-assets transfers under regulation (EU) 2023/1113, 2024. URL: https://www.eba.europa.eu/activities/single-rulebook/regulatory-activities/anti-money-laundering-and-countering-financing-terrorism/guidelines-information-requirements-relation-transfers-funds-and-certain-crypto-assets-transfers. Accessed: 2026-05-27.

29 European Banking Authority. MiCA article 75: Providing custody and administration of crypto-assets on behalf of clients, 2026. Accessed: 2026-05-26. URL: https://www.eba.europa.eu/regulation-and-policy/single-rulebook/interactive-single-rulebook/17894.

30 European Parliament and Council of the European Union. Regulation (EU) 2022/2554 on digital operational resilience for the financial sector, 2022. Accessed: 2026-05-26. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32022R2554.

31 European Parliament and Council of the European Union. Regulation (EU) 2023/1113 on information accompanying transfers of funds and certain crypto-assets, 2023. Accessed: 2026-05-27. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32023R1113.

32 European Parliament and Council of the European Union. Regulation (EU) 2023/1114 on markets in crypto-assets, 2023. Accessed: 2026-05-26. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32023R1114.

33 Sebastian Faller, Tobias Handirk, Julia Hesse, Máté Horváth, and Anja Lehmann. Password-protected key retrieval with(out) hsm protection. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 2445–2459. Association for Computing Machinery, 2024. doi:10.1145/3658644.3690358.

34 Mohammad Farhad, Gourab Saha, Masum Alam Nahid, Fairuz Rahaman Chowdhury, Partha Protim Paul, Mohammed Raihan Ullah, and Md Sadek Ferdous. Secure backup and recovery of ssi wallets using solid pod technology. In *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1101–1111, 2024. doi:10.1109/COMPSAC61105.2024.00149.

35 Shahin Fatima and Shish Ahmad. Secure and effective key management using secret sharing schemes in cloud computing. *International Journal of e-Collaboration*, 2020. doi:10.4018/IJeC.2020010101.

36 FIDO Alliance. Synced passkey deployment: Emerging practices for consumer use cases, 2024. Accessed: 2026-05-26. URL: https://fidoalliance.org/wp-content/uploads/2024/05/Synced-Passkey-Deployment_-Emerging-Practices-for-Consumer-Use-Cases_2024-May-31.pdf.

37 Financial Action Task Force. Updated guidance for a risk-based approach to virtual assets and virtual asset service providers, 2021. Accessed: 2026-05-26. URL: https://www.fatf-gafi.org/en/publications/Fatfrecommendations/Guidance-rba-virtual-assets-2021.html.

38 Financial Action Task Force. FATF updates standards on recommendation 16 on payment transparency, 2025. Accessed: 2026-05-27. URL: https://www.fatf-gafi.org/en/publications/Fatfrecommendations/update-Recommendation-16-payment-transparency-june-2025.html.

39 Fireblocks. Backup and recovery, 2026. Accessed: 2026-05-26. URL: https://ncw-developers.fireblocks.com/docs/backup-recovery-1.

40 Fireblocks. Backup and recovery overview, 2026. Accessed: 2026-05-26. URL: https://developers.fireblocks.com/docs/perform-drs-process.

41 Qianwen Gao and Yichi Tu. A blockchain wallet scheme with multi-factor authentication based on distributed system. In *2024 4th International Conference on Blockchain Technology and Information Security (ICBCTIS)*, pages 55–61, 2024. doi:10.1109/ICBCTIS64495.2024.00017.

42 Shuangyu He, Qianhong Wu, Xizhao Luo, Zhi Liang, Dawei Li, Hanwen Feng, Haibin Zheng, and Yanan Li. A social-network-based cryptocurrency wallet-management scheme. *IEEE Access*, 6:7654–7663, 2018. doi:10.1109/ACCESS.2018.2799385.

43 Xiaojian He, Jinfu Lin, Kangzi Li, and Ximeng Chen. A novel cryptocurrency wallet management scheme based on decentralized multi-constrained derangement. *IEEE Access*, 7:185250–185263, 2019. doi:10.1109/ACCESS.2019.2961183.

44 Arran Holmes and William J. Buchanan. A framework for live host-based bitcoin wallet forensics and triage. *Forensic Science International: Digital Investigation*, 44, 2023. doi:10.1016/j.fsidi.2022.301486.

45 Ivan Homoliak and Martin Perešíni. SoK: Cryptocurrency wallets: A security review and classification based on authentication factors. In *2024 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–8. IEEE, 2024. doi:10.1109/ICBC59979.2024.10634439.

46 Sabine Houy, Philipp Schmid, and Alexandre Bartel. Security aspects of cryptocurrency wallets: A systematic literature review. *ACM Computing Surveys*, 56(1):1–31, 2023. doi:10.1145/3596906.

47 Felix Hrandner and Christof Rabensteiner. Horcruxes for everyone – A framework for key-loss recovery by splitting trust. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 50–57, 2019. doi:10.1109/TrustCom/BigDataSE.2019.00017.

48 Mingxing Hu, Zhen Liu, and Yunhong Zhou. Efficient post-quantum secure deterministic wallet scheme. *Cybersecurity*, 7, 2024. doi:10.1186/s42400-024-00216-w.

49 Ah hyun Park, Hyejin Ryu, Woobeen Park, and Doowon Jeong. Forensic investigation framework for cryptocurrency wallet in the end device. *Computers & Security*, 2023. doi:10.1016/j.cose.2023.103392.

50 Lilita Infante, Roger A. Hallman, John Hays, Evelyn Cronnon, and Uri Stav. Recovery cat: A digital forensics tool for cryptocurrency investigations. In *2024 12th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–5, 2024. doi:10.1109/ISDFS60797.2024.10527279.

51 Olson Italis, Samuel Pierre, and Alejandro Quintero. Privacy-preserving model for biometric-based authentication and key derivation function. *Journal of Information Security and Applications*, 78:103624, 2023. doi:10.1016/j.jisa.2023.103624.

52 Stanislaw Jarecki, Aggelos Kiayias, Hugo Krawczyk, and Jiayu Xu. Highly-efficient and composable password-protected secret sharing (or: How to protect your bitcoin wallet online). In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016. doi:10.1109/EuroSP.2016.30.

53 Stanisław Jarecki, Aggelos Kiayias, Hugo Krawczyk, and Jiayu Xu. Toppss: Cost-minimal password-protected secret sharing based on threshold oprf. In *Applied Cryptography and Network Security*, 2017. doi:10.1007/978-3-319-61204-1_3.

54 Farah Maath Jasim, Ali Makki Makki Sagheer, and Abdullah Mohammed Awad. Enhancement of digital signature algorithm in bitcoin wallet. *Bulletin of Electrical Engineering and Informatics*, 10:449–457, 2021. doi:10.11591/eei.v10i1.2339.

55 Woochang Jeong, Haesung Park, and Chanik Park. Keyscrub: A reliable key backup and recovery method for blockchain. *IEEE Access*, 11:91747–91755, 2023. doi:10.1109/ACCESS.2023.3308208.

56 Kostis Karantias. SoK: A taxonomy of cryptocurrency wallets. Cryptology ePrint Archive, Paper 2020/868, 2020. URL: https://eprint.iacr.org/2020/868.

57 Atharva Vijay Khade, Harsh Rajesh Patel, and Chirag Modi. Mnemonic phrase management and sim based two-factor authentication (2fa) for mobile wallets in blockchain. In *2023 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS)*. IEEE, 2023. doi:10.1109/ICBDS58040.2023.10346266.

58 Hyeonhak Kim and Sanjin Park. Cryptocurrency recovery framework using pre-signed transaction. *International Journal of Information Security*, 24, 2025. doi:10.1007/s10207-025-00994-5.

59 Jihwan Kim, Pyung Kim, Younho Lee, and Daeseon Choi. Key backup and recovery for resilient did environment. *Mathematics*, 12, 2024. doi:10.3390/math12060830.

60 Jihwan Kim, Younho Lee, and Daeseon Choi. Key-audit chain (kac): A resilient and reliable key management framework for did-based decentralized environments. *IEEE Access*, 13:141871–141885, 2025. doi:10.1109/ACCESS.2025.3597593.

61 Myungsun Kim, Sangrae Cho, Seongbong Choi, Young-Seob Cho, Soohyung Kim, and Hyung Tae Lee. A key recovery protocol for multiparty threshold ecdsa schemes. *IEEE Access*, 10:133206–133218, 2022. doi:10.1109/ACCESS.2022.3230683.

62 Ju-seong Ko and Jin Kwak. Private key recovery on bitcoin with duplicated signatures. *KSII Transactions on Internet and Information Systems*, 14:1280–1300, 2020. doi:10.3837/tiis.2020.03.020.

63 Katharina Krombholz, Aljosha Judmayer, Matthias Gusenbauer, and Edgar Weippl. The other side of the coin: User experiences with bitcoin security and privacy. In *Financial Cryptography and Data Security*. Springer, 2017. doi:10.1007/978-3-662-54970-4_33.

64 Ledger. What is ledger recover?, 2026. Accessed: 2026-05-26. URL: https://www.ledger.com/academy/what-is-ledger-recover.

65 Niko Lehto, Kimmo Halunen, Outi-Marja Latvala, Anni Karinsalo, and Jarno Salonen. Cryptovault – A secure hardware wallet for decentralized key management. In *2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS)*, pages 1–4, 2021. doi:10.1109/COINS51742.2021.9524133.

66 Guojia Li and Lin You. A consortium blockchain wallet scheme based on dual-threshold key sharing. *Symmetry*, 13:1444, 2021. doi:10.3390/sym13081444.

67 Guojia Li, Lin You, Gengran Hu, and Liqin Hu. Recoverable private key scheme for consortium blockchain based on verifiable secret sharing. *KSII Transactions on Internet and Information Systems*, 15:2865–2878, 2021. doi:10.3837/tiis.2021.08.009.

68 Zhen Liu, Guomin Yang, Duncan S. Wong, Khoa Nguyen, Huaxiong Wang, Xiaorong Ke, and Yining Liu. Secure deterministic wallet and stealth address: Key-insulated and privacy-preserving signature scheme with publicly derived public key. *IEEE Transactions on Dependable and Secure Computing*, 2022. doi:10.1109/TDSC.2021.3078463.

69 Amin Mahnamfar, Kemal Bicakci, and Yusuf Uzunay. Rostam: A passwordless web single sign-on solution mitigating server breaches and integrating credential manager and federated identity systems. *Computers & Security*, 139:103739, 2024. doi:10.1016/j.cose.2024.103739.

70 Sai Krishna Deepak Maram, Fan Zhang, Lun Wang, Andrew Low, Yupeng Zhang, Ari Juels, and Dawn Song. Churp: Dynamic-committee proactive secret sharing. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2369–2386. Association for Computing Machinery, 2019. doi:10.1145/3319535.3363203.

71 Natsume Matsuzaki, Masayuki Fukumitsu, and Yoshihiro Kita. Emergency button: Evacuation of crypto asset when key loss. In *2023 Eleventh International Symposium on Computing and Networking Workshops (CANDARW)*, 2023. doi:10.1109/CANDARW60564.2023.00048.

72 Ken Naganuma, Takayuki Suzuki, and Takatoshi Ohara. Leakable mnemonic phrase: A new orientation of wallets backup based on biometric-key. In *2024 International Symposium on Information Theory and Its Applications (ISITA)*, pages 325–328, 2024. doi:10.23919/ISITA60732.2024.10858205.

73 Ken Naganuma, Takayuki Suzuki, Masayuki Yoshino, Kenta Takahashi, Yosuke Kaga, and Noboru Kunihiro. New secret key management technology for blockchains from biometrics fuzzy signature. In *2020 15th Asia Joint Conference on Information Security (AsiaJCIS)*, 2020. doi:10.1109/asiajcis50894.2020.00020.

74 Vivek Nair and Dawn Song. Decentralizing custodial wallets with mfkdf. In *2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9, 2023. doi:10.1109/ICBC56567.2023.10174998.

75 National Institute of Standards and Technology. Announcing approval of three federal information processing standards for post-quantum cryptography, 2024. Accessed: 2026-05-26. URL: https://csrc.nist.gov/News/2024/postquantum-cryptography-fips-approved.

76 National Institute of Standards and Technology. NIST selects HQC as fifth algorithm for post-quantum encryption, 2025. Accessed: 2026-05-27. URL: https://www.nist.gov/news-events/news/2025/03/nist-selects-hqc-fifth-algorithm-post-quantum-encryption.

77 Chan Nam Ngo, Daniele Friolo, Fabio Massacci, Daniele Venturi, and Ettore Battaiola. Vision: What if they all die? crypto requirements for key people. In *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 178–183, 2020. doi:10.1109/EuroSPW51379.2020.00032.

78 Saki Otsuki, Hiroto Tamiya, Kengo Mori, Toshiyuki Isshiki, ShinIchiro Matsuo, and Satoshi Obana. Biometrics enhances blockchain wallet governance. In *2025 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9, 2025. doi:10.1109/ICBC64466.2025.11114682.

79 Allan B. Pedin, Nazli Siasi, and Mohammad Sameni. Smart contract-based social recovery wallet management scheme for digital assets. In *Proceedings of the 2023 ACM Southeast Conference*, pages 177–181, 2023. doi:10.1145/3564746.3587016.

80 Gyeongjin Ra, Chang-hyun Roh, and Imyeong Yeoung Lee. A key recovery system based on password-protected secret sharing in a permissioned blockchain. *Computers, Materials and Continua*, 65:153–170, 2020. doi:10.32604/cmc.2020.011293.

81 Avishka Rathnavibhushana, Tharinda Thamaranga, Sandaru Kaveesha, and Chandana Gamage. A social wallet scheme with robust private key recovery. In *2023 IEEE 17th International Conference on Industrial and Information Systems (ICIIS)*, pages 1–6, 2023. doi:10.1109/ICIIS58898.2023.10253598.

82 Hossein Rezaeighaleh and Cliff C. Zou. New secure approach to backup cryptocurrency wallets. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2019. doi:10.1109/GLOBECOM38437.2019.9014007.

83 Safe. Account recovery with safe recoveryhub, 2026. Accessed: 2026-05-26. URL: https://help.safe.global/articles/9622260218-account-recovery-with-saferecoveryhub.

84 Safe. Safe modules, 2026. Accessed: 2026-05-26. URL: https://docs.safe.global/advanced/smart-account-modules.

85 Asia A. Salman, Nidaa Flaih Hassan, and Raghad Abdulaali Azeez. A secure private key recovery based on dna bio-cryptography for blockchain. *Iraqi Journal of Science*, 64:958–972, 2023. doi:10.24996/ijs.2023.64.2.38.

86 Francisco Javier Becerra Sanchez, Antonio Ken Iannillo, and Radu State. Reproducibility Package: SoK Cryptographic Key Recovery for Cryptoasset Custody and Financial Technologies. Collection, version aft-2026-artifacts. (visited on 2026-09-16). URL: https://github.com/franciscobecerra97/Cryptographic-Key-Recovery-for-Financial-Technologies/tree/aft-2026-artifacts, doi:10.4230/artifacts.27945.

87 Aaliya Sarfaraz, Ripon K. Chakrabortty, Saad Aslam, and Ahmad Sahban Rafsanjani. Optimized key recovery for blockchain wallets in sustainable supply chains. In *2024 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 343–347, 2024. doi:10.1109/IEEM62345.2024.10857165.

88 Jungwon Seo, Deokyoon Ko, Suntae Kim, Vijayan Sugumaran, and Sooyong Park. Reminisce: Blockchain private key generation and recovery using distinctive pictures-based personal memory. *Mathematics*, 10, 2022. doi:10.3390/math10122047.

89 Cheman Shaik. Securing cryptocurrency wallet seed phrase digitally with blind key encryption. *International Journal on Cryptography and Information Security*, 2020. doi:10.5121/ijcis.2020.10401.

90 Tanusree Sharma, Vivek C. Nair, Henry Wang, Yang Wang, and Dawn Xiaodong Song. I can't believe it's not custodial! usable trustless decentralized key management. *Conference on Human Factors in Computing Systems - Proceedings*, 2024. doi:10.1145/3613904.3642464.

91 Har Preet Singh, Kyriakos Stefanidis, and Fabian Kirstein. A private key recovery scheme using partial knowledge. In *2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5, 2021. doi:10.1109/NTMS49979.2021.9432642.

92 Reza Soltani, Uyen Trang Nguyen, and Aijun An. Practical key recovery model for self-sovereign identity based digital wallets. In *2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pages 320–325, 2019. doi:10.1109/DASC/PiCom/CBDCom/CyberSciTech.2019.00066.

93 Reza Soltani, Uyen Trang Nguyen, and Aijun An. Decentralized and privacy-preserving key management model. In *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–7, 2020. doi:10.1109/ISNCC49221.2020.9297294.

94 Douglas Stebila and Spencer Wilson. Quantum-safe account recovery for webauthn. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*, 2024. doi:10.1145/3634737.3661138.

95 Yuto Takei and Kazuyuki Shudo. Pragmatic analysis of key management for cryptocurrency custodians. In *2024 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2024. doi:10.1109/ICBC59979.2024.10634356.

96 Chen Wang, Zi Yuan Liu, and Masahiro Mambo. On the proof of ownership of digital wallets. *Cryptography*, 8, 2024. doi:10.3390/cryptography8040057.

97 Sam M. Werner, Daniel Perez, Lewis Gudgeon, Ariah Klages-Mundt, Dominik Harz, and William J. Knottenbelt. SoK: Decentralized finance (DeFi). In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, pages 30–46. ACM, 2022. doi:10.1145/3558535.3559780.

98 Oliver Wiese, Christoph Weinhold, Jan-Ole Malchow, and Volker Roth. I need this back, later! an exploration of practical secret sharing. In *Proceedings of the 34th Annual Computer Security Applications Conference*, pages 478–489. Association for Computing Machinery, 2018. doi:10.1145/3274694.3274729.

99 World Wide Web Consortium. Web authentication: An API for accessing public key credentials level 3, 2026. Accessed: 2026-05-26. URL: https://www.w3.org/TR/webauthn-3/.

100 Feng Xiong, Ruiyang Xiao, Wei Ren, Rongyue Zheng, and Jianlin Jiang. A key protection scheme based on secret sharing for blockchain-based construction supply chain system. *IEEE Access*, 7:126773–126786, 2019. doi:10.1109/ACCESS.2019.2937917.

101 Zhimei Yang, Changgen Peng, Chongyi Zhong, and Yangyang Long. Consortium blockchain private key protection scheme based on rational secret sharing and blockchain. *Computer Networks*, 242:110260, 2024. doi:10.1016/j.comnet.2024.110260.

102 Mert Yldz and erif Bahtiyar. A novel key management framework for secure and scalable decentralized identity systems. In *2024 17th International Conference on Security of Information and Networks (SIN)*, pages 1–8, 2024. doi:10.1109/SIN63213.2024.10871880.

103 Fan Zhang, Philip Daian, Iddo Bentov, Ian Miers, and Ari Juels. Paralysis proofs: Secure dynamic access structures for cryptocurrency custody and more. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, page 115. Association for Computing Machinery, 2019. doi:10.1145/3318041.3355459.

104 Yifang Zhang, Mingyue Wang, Yu Guo, and Fangda Guo. Towards dynamic and reliable private key management for hierarchical access structure in decentralized storage. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 3371–3380. Association for Computing Machinery, 2023. doi:10.1145/3583780.3615090.

105 Xingfan Zhao, Changgen Peng, Weijie Tan, and Kun Niu. Blockchain-based key management scheme using rational secret sharing. *Computers, Materials and Continua*, 79:307–328, 2024. doi:10.32604/cmc.2024.047975.

106 Wenbo Zheng, Kunfeng Wang, and Feiyue Wang. Gan-based key secret-sharing scheme in blockchain. *IEEE Transactions on Cybernetics*, 51:393–404, 2021. doi:10.1109/TCYB.2019.2963138.

107 Liyi Zhou, Xihan Xiong, Jens Ernstberger, Stefanos Chaliasos, Zhipeng Wang, Ye Wang, Kaihua Qin, Roger Wattenhofer, Dawn Song, and Arthur Gervais. SoK: Decentralized finance (DeFi) attacks. In *2023 IEEE Symposium on Security and Privacy*, pages 2444–2461. IEEE, 2023. doi:10.1109/SP46215.2023.10179435.

108 Xiuwen Zhou, Qianhong Wu, Bo Qin, Xinyi Huang, and Jianwei Liu. Distributed bitcoin account management. In *2016 IEEE Trustcom/BigDataSE/ISPA*, 2016. doi:10.1109/TrustCom.2016.0052.

109 Fangdong Zhu, Wen Chen, Yunpeng Wang, Ping Lin, Tao Li, Xiaochun Cao, and Long Yuan. Trust your wallet: A new online wallet architecture for bitcoin. In *2017 International Conference on Progress in Informatics and Computing (PIC)*, pages 307–311, 2017. doi:10.1109/PIC.2017.8359562.

110 Yanlin Zhu, Lirong Xia, and Oshani Seneviratne. A proposal for account recovery in decentralized applications. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 148–155, 2019. doi:10.1109/Blockchain.2019.00028.

Gatling: Rapid-Fire Consensus from Parallel Composition

Giulia Scaffino
TU Wien, Austria
Common Prefix, Vienna, Austria

Max Resnick
Anza, New York, NY, USA

Joachim Neu
a16z Crypto Research, New York, NY, USA

Abstract

Consensus protocols form the core of blockchains and other replicated state machines, ensuring that all correct nodes process the same totally ordered log of input transactions. In fault-free executions, performance is driven by the good-case transaction latency – the time between a transaction becoming known to all nodes and its confirmation by the consensus protocol – which depends on both how frequently proposals are made and, once made, how quickly they are confirmed. While prior work has established tight lower bounds on confirmation latency that modern protocols already achieve, it remains open whether the inter-proposal time can be further reduced below the state-of-the-art of one network delay.

We introduce Gatling, an atomic broadcast protocol that achieves arbitrarily small inter-proposal times under rotating leader schedules; in particular, smaller than the network delay. Gatling runs multiple parallel instances of a black-box atomic broadcast protocol and staggers their proposal schedules to generate proposals in faster succession than state-of-the-art protocols. A deterministic interleaving rule merges the outputs of these instances into a single global log. We analyze the effects of head-of-line blocking caused by crashed leaders, and derive Gatling's optimal number of parallel instances. We further study the impact of Gatling on predictable validity and present two variants that retain this property. Finally, our experiments confirm that Gatling can be used with off-the-shelf component protocols to achieve low latency without fine-tuning the component protocol for minimum latency.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Distributed Systems, Byzantine Fault Tolerance Consensus, Parallel Composition, Inter-Proposal Time

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.4

Related Version *Extended Version*: `https://arxiv.org/abs/2606.18220` [40]

Supplementary Material *Software (Source Code)*: `https://github.com/scaffino/Gatling` archived at `swh:1:dir:316ac6dbf49dd46e0a39e88b31573b63a9068dcb`

1 Introduction

After the launch of the Bitcoin network in 2009, block times fell exponentially for fifteen years. Bitcoin 2009: 10 minutes. Ethereum 2015: 15 seconds. Solana 2020: 400 ms [35, 11, 51]. However, more than five years later, blockchains still launch with almost the same block times Solana had in 2020. Why did fifteen years of exponential progress, which promised to drive block times to zero, suddenly stall in 2020?

The reason is structural. All of the blockchains listed above proceed one block at a time. Each block references its parent, and so before consensus can begin on the next block, the previous block must be known. Roughly speaking, this bounds block times to the time it takes


© Giulia Scaffino, Max Resnick, and Joachim Neu;
licensed under Creative Commons License CC-BY 4.0



for a single round of consensus – and that time has tight, well-known lower bounds. Prior work [2] proved a 3Δ confirmation bound under partial synchrony in the $3f+1 \leq n < 5f-1$ regime, and a smaller bound holds at $n \geq 5f+1$, achieved by Minimmit [18] at the cost of resilience. The remaining knob is the inter-proposal time. Pipelining (e.g., in Moonshot [22] and Hydrangea++ [44]) pushed it to Δ. Hydrangea++ asserts, without proof, that Δ is the floor. Intuitively, they argue that the next leader cannot propose a block without seeing the previous block, and that takes at least Δ time. This argument is of course correct; however, in this paper we challenge the assumption that the next leader must see the previous block before proposing their own.

We present Gatling, an atomic broadcast protocol whose inter-proposal time can be made arbitrarily small, in particular smaller than the network delay Δ. Gatling runs K parallel instances of a closed-box atomic broadcast protocol with inter-proposal time T_{ipt}, staggered by T_{ipt}/K, and interleaves their outputs into a single log via a deterministic merge rule. Each component instance still respects the 3Δ confirmation bound; the speedup comes from composition. Setting K large drives the inter-proposal time toward zero. As a proof of concept, we instantiate Gatling on top of a Simplex [15] implementation and obtain 244 ms end-to-end transaction latency and 50 ms slot times on a global cluster with nodes in Asia, Europe, and North America.

Setting. Blockchains are distributed systems whose operation relies on atomic broadcast protocols to ensure consistent replication of state across participating nodes. In such protocols, nodes receive *transactions* as inputs from the environment – an abstraction representing all entities external to the protocol – and, after some time, they output to the environment a totally ordered sequence of *confirmed* transactions, commonly referred to as the *log*. At their core, atomic broadcast protocols typically consist of a proposal phase, in which a designated node proposes a payload of transactions to be appended to the log, followed by a number of voting phases (often two), during which nodes collectively decide whether to confirm the proposal. Phases repeat over time, allowing the protocol to extend the log incrementally as new transactions arrive.

In atomic broadcast executions in which all nodes follow the protocol (*good-case*), a key performance metric is the *average transaction latency*, defined as the average time required for a transaction to be confirmed after it has been received by all nodes. The good-case average transaction latency, hereafter for brevity referred to as *transaction latency*, naturally decomposes into two components: the *inter-proposal time*, which captures the time until a transaction is included in a proposal, and the *confirmation time*, which measures how long a proposal takes to become confirmed. Minimizing transaction latency therefore requires reducing both components simultaneously. Tab. 1 summarizes the inter-proposal time, confirmation time, and transaction latency of several state-of-the-art protocols. The remainder of this section unpacks where each component's lower bound comes from and how prior protocols have approached it.

Confirmation Time. Prior work [2] fully characterizes the tight good-case bounds on confirmation time in the authenticated setting across synchronous, partially synchronous, and asynchronous network models. Under partial synchrony with $3f+1 \leq n < 5f-1$, the tight lower bound is three communication rounds. Traditional protocols such as PBFT [13], Tendermint [10], and Simplex [15] operate in conservative rounds of duration Δ and saturate this bound at 3Δ. The bound therefore offers no further room for improvement at the standard resilience threshold: any reduction in transaction latency must come from the inter-proposal time.

Table 1 Inter-proposal time (T_{ipt}), confirmation time (T_{conf}), and average good-case transaction latency of different protocols. Gatling is parameterized by K such that $\epsilon = 2\Delta/K$ can be made arbitrarily small. Unlike the other protocols, Gatling is a closed-box construction: it uses the underlying protocol only through its interface, without regard to internals.

Protocol	Inter-Proposal Time	Confirmation Time	Transaction Latency	Closed Box
PBFT [13], Tendermint [10]	3Δ	3Δ	4.5Δ	No ✗
HotStuff [52]	2Δ	7Δ [a)]	8Δ	No ✗
HotStuff-1 [27], HotStuff-2 [32]	2Δ	5Δ [b)]	6Δ	No ✗
Simplex [15]	2Δ	3Δ	4Δ	No ✗
Moonshot [22], Hydrangea++ [44]	Δ	3Δ [c)]	3.5Δ	No ✗
Shoal++ [6][d)]	Δ	4.5Δ	5Δ	No ✗
Gatling (this work)	ϵ	3Δ	$(3+\epsilon/2)\Delta$	Yes ✓

[a)] HotStuff achieves a minimum commit latency of 7Δ when the next leader aggregates the votes for the current leader's proposal. [b)] HotStuff-1 has a 5Δ confirmation time on the regular path, and an optimistic (speculative) path that reduces latency to 3Δ by allowing nodes to send early execution (confirmation) responses to clients after a single QC. Clients need to handle responses with care, to avoid accepting provisional confirmations that may later be revoked or reordered. [c)] We ignore the 20%-resilience fast path of Hydrangea++ as this is an orthogonal technique. [d)] After completion of this work, it was brought to our attention that Shoal++ [6] employs a construction similar to Gatling in the context of DAG-based protocols to reduce the inter-proposal time.

Inter-Proposal Time: Pipelining. Traditional atomic broadcast protocols operate in discrete slots (or views), each associated with a leader pseudo-randomly elected from the participating nodes (rotating leaders). The leader of the slot is entitled to cast a proposal for its slot. Typically, proposals are strictly sequential: each proposal must complete all voting phases before the next can be issued. In PBFT and Tendermint, for instance, a leader proposes a block and waits for two full rounds of voting, yielding a "propose-vote-vote" structure and an inter-proposal time of 3Δ. The HotStuff family of protocols [52, 27, 32] and Jolteon [25] relax this strict sequentiality by cryptographically chaining proposals with quorum certificates (QCs), such that each proposal includes a QC that carries votes for the previous proposal: as a result, a leader can issue a new proposal while nodes are still voting on the previous one. This technique, called pipelining, produces an effective "propose-vote-propose-vote" pattern and reduces the protocols' inter-proposal time to 2Δ.[1] Simplex [15] also achieves a 2Δ inter-proposal time: unlike the HotStuff family, which attains this bound via chained QCs in a single-vote pipeline, Simplex advances to the next slot immediately upon notarization of the current slot, while a distinct finalization vote proceeds in parallel with the subsequent proposal.

Optimistic designs push pipelining even further: under the assumption of consecutive honest leaders, Moonshot [22] and Hydrangea++ [44] allow leaders to issue new proposals even before the previous one has been quorum-certified. This aggressive pipelining results in a "propose-propose-propose" structure, enabling a new proposal at every round and achieving the state-of-the-art minimum inter-proposal time of Δ. Because transactions may arrive at any point between two consecutive proposals, the expected waiting time until inclusion in a proposal is half of the inter-proposal time; the lowest transaction latency achieved by existing protocols is the 3.5Δ of Moonshot and Hydrangea++.

[1] HotStuff [52] also advertises to advance in rounds matching the actual network delay δ, decoupling progress from the pessimistic bound Δ; this property, known as *responsiveness*, is shared by several protocols [13, 27, 32, 25, 22]. Since the dominant strategy for profit-maximizing proposers is to exhaust their timeouts, blockchain protocols de-facto run non-responsively; we discuss this below and in Sec. 9.

Inter-Proposal Time: Stable Leaders. In classical atomic broadcast protocols, proposals are cryptographically chained: a leader can successfully propose a new batch of transactions only after receiving the previous leader's proposal. As a result, when the leader rotates at every slot, the inter-proposal time must be at least Δ to give time to the previous proposal to propagate to the next leader. Protocols achieving a sub-Δ inter-proposal time typically allow a single leader to remain in control for multiple consecutive slots and issue several proposals in sequence (stable or slowly rotating leaders). In this regime, the leader has local access to its own prior proposals and can therefore bypass network propagation delays. This is the case, for instance, of the original PBFT [13]. Keeping the same leader in power for extended periods is, however, problematic in blockchains, as it grants the leader disproportionate control over transaction ordering, exacerbating MEV extraction, censorship, and eroding fairness among nodes. A middle ground is achieved by some PBFT-like protocols instantiated with slowly rotating leaders: they allow a single leader to produce multiple proposals within a fixed number of consecutive slots (called a window), separated only by local processing delays (a few tens of milliseconds). When the leader rotates across windows, though, they incur a larger inter-proposal time of at least Δ (in practice, a few hundreds of milliseconds). Another example of such protocols is Alpenglow [5].

On Responsiveness. A substantial body of work has been devoted to the design of *responsive* Byzantine fault-tolerant protocols. Responsiveness refers to the ability of a protocol to make progress as a function of the actual network conditions (rounds of length δ), rather than operating with conservative timeouts that reflect the worst-case network bound (Δ). The goal is to reduce latency under $\delta \ll \Delta$ network conditions.

Recent work [41, 17] shows that, in blockchains, the dominant strategy of profit-maximizing proposers is to delay their proposal until their timeout expires: this allows them to gain more information about incoming transactions and craft a proposal that maximizes the Maximal Extractable Value (MEV). This behavior is often referred to as *timing games*, and has been observed in production systems such as Ethereum [41] and Solana [17]. As a result of economic incentives, blockchain protocols de-facto run non-responsively at Δ speed, and not at the δ speed that responsiveness would allow. For this reason, in this work, we consider protocol rounds of length Δ.

1.1 Our Contribution

Gatling. Gatling (Sec. 3) runs K parallel and independent instances of a closed-box atomic broadcast component protocol – e.g., PBFT, Tendermint, Simplex, Moonshot, or a DAG-based protocol such as Bullshark [47], Shoal [46], or Mysticeti [8]. Each component instance is parameterized with a fixed, regular proposal schedule that generates a proposal every T_{ipt} time, so when all nodes are honest each instance extends its log every T_{ipt}. In Gatling, K is a tunable parameter, and the proposal schedules of consecutive instances are offset by T_{ipt}/K. As illustrated in Fig. 1 for $K = 4$, the component protocols together produce a T_{ipt}/K inter-proposal time. By appropriately choosing K, Gatling achieves an arbitrarily small inter-proposal time.

Since we want Gatling to output a single transaction log rather than K independent parallel logs, the K component logs are combined via a deterministic merge rule. All nodes assign the same index from 1 to K to each component instance, and each node locally constructs the Gatling output log by interleaving the K component logs in order of increasing slot number and instance number. For example, when $K = 2$, the Gatling log alternates slot

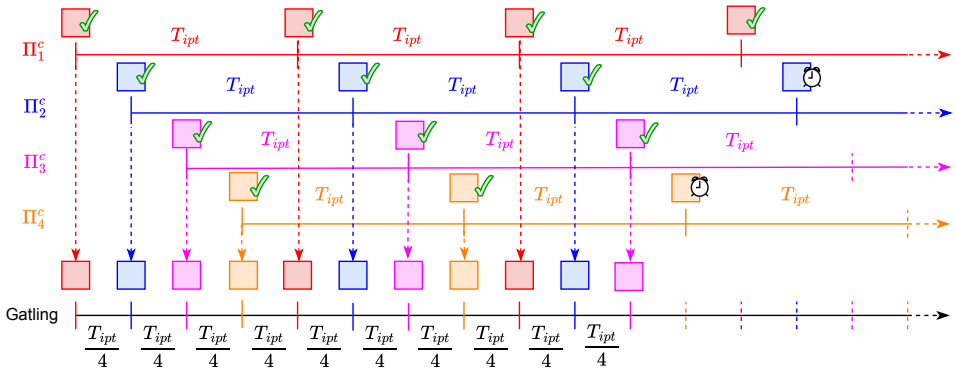


Figure 1 The Gatling protocol depicted in this figure runs $K = 4$ instances of a component protocol Π^c. The Gatling output log (at the bottom) is formed by deterministically interleaving blocks from all instances, in increasing order of slot number and instance number. A block is confirmed once all preceding blocks in this global order are confirmed. Checkmarks indicate blocks that are confirmed by their instance, whereas clocks indicate blocks that are not yet confirmed by their instance. The last red block is not yet included in the Gatling log because, in the interleaved ordering, it comes after the third orange block, which has not yet been confirmed. This does not happen in the good-case but it could happen in case of, e.g., a network hiccup.

by slot: the slot-1 block of instance 1, then the slot-1 block of instance 2, then the slot-2 block of instance 1, and so on. Gatling confirms a block once all slots that come before it have been confirmed.

We prove that Gatling implements secure atomic broadcast with transaction latency $(T_{\text{ipt}}/K, T_{\text{conf}})$, where T_{ipt}/K is the inter-proposal time – which can be made arbitrarily small (Sec. 4) – and T_{conf} is the confirmation time. We perform a stochastic latency analysis (Sec. 5) to characterize the impact of occasional proposal confirmation failures and identify the optimal inter-proposal time that minimizes expected end-to-end latency. We implement a Gatling proof of concept on top of a production-grade Simplex deployment [15, 42, 19] with a deliberately unoptimized 500 ms per-instance inter-proposal time, and run it on a global cluster with nodes in Asia, Europe, and North America. End-to-end transaction latency falls as K grows, reaching 244 ms at $K = 9$ under a 1% proposal-drop rate while driving the slot time down to 50 ms (Sec. 8).

Discussion and Related Work. We further discuss how the design of vanilla Gatling inherently loses what is sometimes called *predictable validity* [36, 37, 29] (Sec. 6) – a property that enables proposers to reason at proposal time about transaction execution outcomes – and we propose two variants that retain it (Sec. 7) under a slowly rotating leader schedule. The first variant retains predictable validity by mimicking the behaviour of, e.g., PBFT-style protocols. The second retains predictable validity at the execution level by creating two distinct tiers of block space and interleaving their execution (Sec. 7). One tier, named *subprime block space*, is reserved for transactions that are not time-critical and cannot be front-run (e.g., layer-2 data blob transactions); the other, *prime block space*, is reserved for time-critical and highly contentious transactions (e.g., DeFi).

A long line of work has sought to reduce consensus latency through the design of responsive protocols complemented with pipelining [52, 27, 32, 25, 22], speculative execution [27], optimistic assumptions [22], or fast-path techniques that trade resilience for fewer voting rounds [18, 5, 43, 45]. We compare Gatling with these approaches in more detail

in Sec. 9. Gatling improves latency along an orthogonal dimension: it achieves arbitrarily small inter-proposal times through composition, while each component protocol operates under conservative worst-case network assumptions and traditional Byzantine fault-tolerance thresholds. Notably, *Gatling shows that even starting from a simple, unoptimized protocol with poor standalone performance* – e.g., Simplex with a 500 ms inter-proposal time – *parallel composition alone is sufficient to achieve latencies improving upon those of highly optimized protocols*. This highlights that *extreme protocol-level optimization is not a prerequisite for ultra-low latency*. Moreover, Gatling is designed to remain compatible with responsive, pipelined, speculative, and fast-path techniques implemented within its component protocols, enabling these mechanisms to be seamlessly layered on top of it.

Conceptually, Gatling can be viewed as extending the parallelism underlying DAG-based protocols [33, 28, 47, 21, 46, 8]. Whereas DAG protocols support multiple concurrent ordering streams over a shared graph of proposals that reference one another, Gatling removes these references entirely by composing independent consensus instances and imposing staggered proposal schedules. The Gatling log is recovered with a deterministic merge rule that does not need proposals to reference one another, thereby allowing the inter-proposal time to drop below the network delay.

After completion of this work, it was brought to our attention that Shoal++ [6] employs a construction similar to Gatling in the context of DAG-based protocols to reduce the inter-proposal time. In particular, it operates three parallel DAG instances, time-staggered by a fixed Δ offset, in a not-fully-generic "ajar-box" manner. By contrast, Gatling explores regimes with inter-proposal times below the network delay and provides variants that retain predictable validity even when the offsets become arbitrarily small. Gatling is closed-box, and our experiments allow us to isolate the latency reduction achieved purely from time-staggered parallel composition.

2 Model and Preliminaries

Our overarching goal is to design an atomic broadcast protocol. We consider a setting with n nodes, each of which equipped with a cryptographic identity (typically, a public/secret key pair for digital signatures) known to all other nodes (public key infrastructure, PKI). Over time, the nodes receive transactions from the *environment* and, over time, they return a log of transactions to the environment. The nodes send each other messages through a *network* to reach agreement on an ordering of their input transactions into their output logs. The nodes' output logs are *confirmed*. Time proceeds in discrete *rounds*. Throughout the entire protocol execution, the nodes have *synchronized clocks*: all nodes always know which is the current round. In each round, each node receives messages from the network and possibly transactions from the environment. In every round, each node updates its internal state, produces messages to send to other nodes via the network, and outputs a log of confirmed transactions.

An *adversary* aims to disrupt the system by corrupting nodes and delaying network messages. We consider an adversary that corrupts a fixed set of at most f nodes at the start of the protocol execution, before any randomness is drawn (static corruption). The adversary learns the internal state of corrupted nodes and causes them to deviate arbitrarily from the protocol for the entire execution (permanent Byzantine faults). Corrupted nodes, also known as adversarial nodes, run any algorithm chosen by the adversary in time polynomial in the size of the inputs (computationally bounded adversary).

Nodes can exchange messages between each other via a fully-connected network of point-to-point links. The network is *partially synchronous* [23]: there exits an upper-bound message delay Δ that is known to all nodes and a global stabilization time $\mathsf{GST} < \infty$ that is adaptively chosen by the adversary. Before GST, the adversary can delay messages arbitrarily (asynchronous network). After GST, the adversary can delay messages by at most Δ (synchronous network). During synchronous periods, when a node instructs the network to send a message to another node, the message is enqueued in the recipient's pending message queue together with a countdown initialized to Δ; the countdown decreases by 1 with each round. The adversary can also decrease the countdown of every message at will. Once the countdown hits 0, the message is delivered to the recipient at the beginning of the next round. This means that every message sent by an honest node by time t is delivered to all honest nodes by time $\max(t, \mathsf{GST}) + \Delta$.

Atomic Broadcast Interface. The nodes run a Byzantine fault-tolerant consensus protocol of the atomic broadcast flavour to reach agreement on their output logs, i.e., a common ordering of their input transactions.

▶ **Definition 1** (Atomic Broadcast Interface). *An atomic broadcast protocol is run by n nodes. In every round, nodes possibly receive transactions from the environment and they return to the environment a confirmed sequence of transactions called output log.*

We consider atomic broadcast executions in which, regardless of the strategy of the adversary, all correct nodes output a log λ that is safe and live. We write $\lambda \preceq \tilde{\lambda}$ to denote that λ is a prefix of $\tilde{\lambda}$. We denote by $\mathsf{broadcast}$ the function that a node invokes to broadcast a transaction to the network, and by $\mathsf{deliveredLog}$ the function that a node invokes to retrieve its output log.

▶ **Definition 2** (Atomic Broadcast Security). *An atomic broadcast protocol is* secure with resilience τ *in partially synchronous networks iff, in every partially-synchronous execution with up to $f \leq \tau$ adversarial nodes, except with probability* $\mathrm{negl}(\kappa)$ *where κ is the security parameter, the following properties hold:*

- ***Safety**: For every two honest nodes p, q and for every two times t, t', if λ is the log output by p at t and $\tilde{\lambda}$ is the log output by q at t', then the two logs are consistent, i.e., $\lambda \preceq \tilde{\lambda}$ or vice versa.*
- ***Liveness**: For every t_0 and every transaction* tx*, if every honest node p calls* $\mathsf{broadcast}$ *with parameter* tx *by time t_0, then there exists a $t_1 \geq t_0$ such that for every $t_2 \geq t_1$ and for every p that calls* $\mathsf{deliveredLog}$ *at t_2,* $\mathsf{tx} \in \lambda$.

Enriched Atomic Broadcast Interface. As components of Gatling, we need an atomic broadcast protocol that proceeds in slots. Each *slot* is associated with a leader node and a fixed proposal time T^{prop}, during which the leader is expected to propose a payload of transactions to be appended to the output log, and broadcast it to the network. Slots typically consists of three phases: a proposal phase, and two voting phases known as notarization and confirmation. Each payload that is included in the output log is annotated with the slot number in which it was proposed. The slot number is non-decreasing along the log entries.

We additionally require the atomic broadcast protocol to implement an enriched interface. This interface is obtained by parameterizing the protocol with a sequence of proposal times, denoted T^{prop}, and by enriching the classical atomic broadcast interface with two additional functions, $\mathsf{broadcast}$ and $\mathsf{deliveredAnnotatedLog}$. Since the Gatling protocol runs several parallel instances of an atomic broadcast protocol, the two new functions make explicit when Gatling writes an input to an instance, or reads the output from an instance. For clarity, we now introduce the enriched interface of the atomic broadcast protocol.

▶ **Definition 3** (Enriched Atomic Broadcast Interface). *An enriched atomic broadcast protocol* Π *is run by* n *nodes, and parameterized by a sequence of proposal times* T^{prop}*. It provides a function* broadcast *that a node can call with a transaction* tx *as parameter and that returns nothing. It also provides a function* deliveredAnnotatedLog *which a node can call without any parameter and that returns an annotated output log, i.e., a sequence of transactions* λ *annotated with the slot number in which they were proposed.*

We emphasize that the enriched interface is merely a syntactic variant of the standard atomic broadcast interface, and does not alter its security semantics. We highlight that virtually all partially synchronous atomic broadcast protocols we are aware of can be straightforwardly adapted to provide the enriched interface. These protocols already operate over a notion of slot – called *view* in PBFT [13], Tendermint [10], HotStuff [52], and Moonshot [22], *heights* in Simplex [15], and *epochs* in Streamlet [16]. They can be easily modified to enforce a fixed, regular schedule of slot proposal times (e.g., every 2Δ time) and to annotate confirmed payloads with the slot number in which they were proposed, a value over which the protocols already establish consensus. Finally, let T_{conf} be the time it takes for a proposal to be confirmed after it has been proposed. We are interested in atomic broadcast protocols that fulfill a stronger notion of liveness, which can be articulated into two properties. We name the first property *slot-driven liveness*.

▶ **Definition 4** (Slot-Driven Liveness). *An atomic broadcast protocol has* slot-driven liveness *iff for any slot starting after* GST *whose associated leader is honest, a (possibly empty) proposal is produced and becomes confirmed by all honest nodes within* T_{conf} *time from the slot's start.*

The second property is *good-case* $(T_{\text{ipt}}, T_{\text{conf}})$*-latency*. Before stating this property, we need to define what it means for a sequence of proposal times to be T_{ipt}-spaced.

▶ **Definition 5** (T_{ipt}-Spaced Sequence). *A sequence* T *is* T_{ipt}-spaced *iff, for every two distinct* $t_1, t_2 \in T$*,* $|t_2 - t_1| \geq T_{\text{ipt}}$*.*

Informally, a protocol has good-case $(T_{\text{ipt}}, T_{\text{conf}})$-latency if, when all nodes are honest, transactions are confirmed $T_{\text{ipt}} + T_{\text{conf}}$ time after they are received by all nodes.

▶ **Definition 6** (Good-Case $(T_{\text{ipt}}, T_{\text{conf}})$-Latency). *A protocol has good-case* $(T_{\text{ipt}}, T_{\text{conf}})$*-latency iff, when the protocol is instantiated with a* T_{ipt}*-spaced proposal sequence* T^{prop}*, if* $\mathsf{GST} = 0$ *and all nodes are honest, then for every* $t \in T^{\text{prop}}$ *and for every transaction* tx*, if* tx *is received by every honest node by* t*, then* tx *is confirmed by every honest node by* $t + T_{\text{conf}}$*.*

3 Protocol

The goal of this work is to design an atomic broadcast protocol with an arbitrarily small inter-proposal time – in particular, smaller than the bound on the network delay. To this end, we design Gatling with a compositional approach. We describe the protocol step by step and we provide the pseudocode executed by the nodes in Alg. 1.

The Gatling protocol is parameterized by a positive interger $K > 0$, a positive integer $T_{\text{ipt}} > 0$, and a black-box atomic broadcast protocol Π^{c} that exposes the enriched interface of Def. 3 and satisfies the slot-driven liveness and good-case $(T_{\text{ipt}},T_{\text{conf}})$-latency of Defs. 4 and 6. For simplicity, we refer to Π^{c} as the component protocol. Nodes running Gatling execute K concurrent and mutually independent instances of the component protocol. Nodes uniquely identify these instances by assigning them indices from 1 to K; in particular, all nodes assign the same index to the same protocol instance. Each instance maintains its own

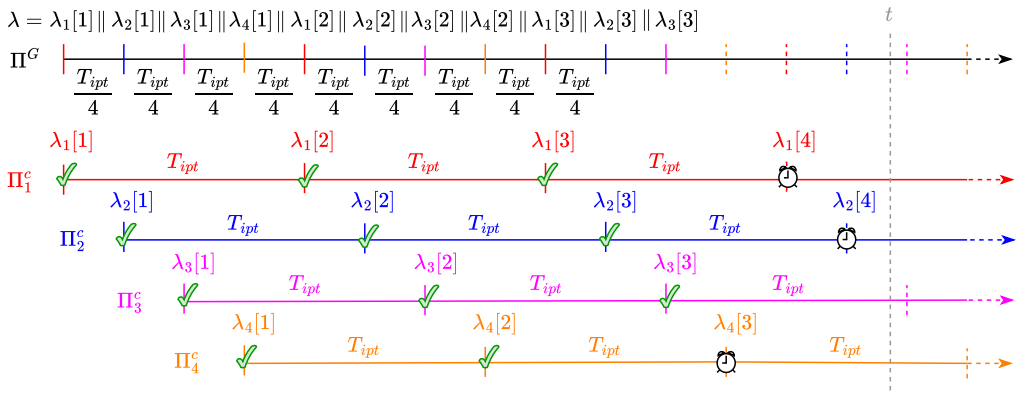


Figure 2 The Gatling protocol Π^{G} depicted in this figure runs $K = 4$ instances of a component protocol Π^{c}. The union of the proposal sequences of Π_1^{c} (red), Π_2^{c} (blue), Π_3^{c} (magenta), Π_4^{c} (orange) is the $T_{\mathrm{ipt}}/4$-spaced proposal sequence of Π^{G}. We denote by $\lambda_j[i]$ the payload confirmed in the i-th slot of the j-th instance. The Gatling log λ at time $t \geq \mathsf{GST}$ is reconstructed at the top as the concatenation of the payloads of the component protocols, ordered by increasing slot and increasing instance number. Checkmarks indicate blocks that are confirmed, whereas clocks indicate blocks that are not yet confirmed. Note that $\lambda_4[3], \lambda_1[4], \lambda_2[4]$, etc, are not yet confirmed by Gatling (not yet part of λ) because, at time t, the corresponding component protocols have not yet confirmed the payloads for those slots.

leader schedule, produces its own annotated output log, and is parameterized by a proposal schedule that generates a proposal every T_{ipt} time. The proposal schedule of each instance is offset by T_{ipt}/K relative to the previous one (Alg. 1, ln. 4). Consequently, the union of the proposal sequences of all K component protocols forms a single proposal sequence with inter-proposal time T_{ipt}/K.

When a node receives a transaction tx from the environment, it invokes the $\mathsf{broadcast}$ function (Alg. 1, ln. 18) that forwards tx to all K instances of the component protocol (Alg. 1, lns. 20 and 21). Then, tx is picked up by the leaders of the instances and included in a proposal. In particular, once the transaction has been received by all nodes, the instance proposing next has a proposal at most T_{ipt}/K time away. Therefore, the transaction is guaranteed to appear in some proposal within at most T_{ipt}/K time and confirmed after an additional T_{conf} time by the corresponding protocol instance.

It is not sufficient for tx to be confirmed by a component protocol: we need it to be confirmed by the Gatling protocol. Towards this, we describe how the output log of Gatling is built, and visually illustrate this in Fig. 2. The nodes construct the Gatling output log by concatenating the K output logs of the component protocols as follows: For each slot index i, the nodes define the i-th slot of the Gatling protocol as the tuple $s_i = (s_{i,1}, \ldots, s_{i,K})$, where $s_{i,j}$ denotes the i-th slot of the j-th instance. Then, they construct the Gatling output log as the sequence of payloads confirmed in the Gatling slots. In other words, the Gatling output log concatenates the payloads of the component protocols output logs, ordered by increasing slot and instance number. In Gatling, proposals are not chained. With this in mind, we now define the Gatling confirmation rule, which consists of three conditions. The payload of slot $s_{i,j}$ is confirmed if: (1) instance j has confirmed the payload of slot s_i, (2) all instances $j = 1, \ldots, K$ have decided for all slots $s_{i'}$ with $i' < i$, and (3) all instances j' with $j' < j$ have decided for slot s_i. As shown in Fig. 3, a component protocol that fails to confirm a proposal for a slot (e.g., due to an adversarial leader) still reaches consensus on the slot being empty.

Algorithm 1 Algorithm run by a node of the Gatling protocol.

function new $\Pi^{\mathrm{G}}(\Pi^{\mathrm{c}}, K, T_{\mathrm{ipt}})$
  // Start component protocols with proposal schedules s.t. the Gatling inter-proposal time is T_{ipt}/K
  for $k = 1, \ldots, K$
    $\pi_k \leftarrow$ new $\Pi^{\mathrm{c}}(T_k^{\mathrm{prop}} = \{(s + \frac{k-1}{K}) \cdot T_{\mathrm{ipt}} \mid s \in \mathbb{N}_0\})$
function Π^{G}.deliveredLog()
  // Get component protocols' output logs annotated with slot number
  for $k = 1, \ldots, K$
    $\lambda_k \leftarrow \pi_k$.deliveredLogAnnotated()
  // Find max slot number for which all component protocols have decided
  $s^* \leftarrow \max\, s \in \mathbb{N}_0$ so that $\forall s \leq s^* : \forall k \leq K : \lambda_k[s] \neq \perp$
  // Find max instance number that has already decided for slot $s^* + 1$
  $k^* \leftarrow \max\, k \in \mathbb{N}_0$ so that $\forall k \leq k^* : \lambda_k[s^* + 1] \neq \perp$
  // Output Gatling protocol's current log
  if $k^* = 0$
    return $\lambda_1[1] \| \ldots \| \lambda_k[1] \| \ldots \| \lambda_1[s^*] \| \ldots \| \lambda_K[s^*]$
  else
    return $\lambda_1[1] \| \ldots \| \lambda_k[1] \| \ldots \| \lambda_1[s^* + 1] \| \ldots \| \lambda_{k^*}[s^* + 1]$
function Π^{G}.broadcast(tx)
  // Forward Gatling protocol's inputs to component protocols
  for $k = 1, \ldots, K$
    π_k.broadcast(tx)

In the next section, we prove that the Gatling protocol in Alg. 1 is secure, and it has a good-case $(T_{\mathrm{ipt}}/K, T_{\mathrm{conf}})$-latency.

4 Analysis

We now state the main results of this work, namely the security and the good-case $(\frac{T_{\mathrm{ipt}}}{K}, T_{\mathrm{conf}})$-latency of Gatling. Proofs appear in the extended version of this work [40].

▶ **Theorem 7** (Gatling Security). *If Π^{c} is a secure atomic broadcast protocol as per Def. 2, then the Gatling protocol in Alg. 1 that uses Π^{c} as its component protocol is also secure as per Def. 2.*

▶ **Theorem 8** (Gatling Good-Case $(\frac{T_{\mathrm{ipt}}}{K}, T_{\mathrm{conf}})$-Latency). *If Π^{c} has good-case $(T_{\mathrm{ipt}}, T_{\mathrm{conf}})$-latency, then the composite protocol in Alg. 1 has good-case $(\frac{T_{\mathrm{ipt}}}{K}, T_{\mathrm{conf}})$-latency.*

5 Optimal Inter-Proposal Time

In the previous sections we established that in the good-case, after GST, Gatling achieves $(T_{\mathrm{ipt}}/K, T_{\mathrm{conf}})$-latency for any $K \geq 1$. Therefore, the Gatling inter-proposal time can be made arbitrarily small by adding component instances. In real world deployments, component instances may occasionally fail to confirm a proposal for a slot, both in the good-case due to network delays and in the bad-case due to the presence of adversarial nodes. The Gatling's confirmation rule turns these failures into head-of-line blocking events on the Gatling log: a delayed or failed confirmation in one component protocol slot can prevent subsequent proposals already confirmed at the component level from being appended to the log.

To see the effect, consider a single failure resolved at the affected instance's next slot (i.e., $T_{\mathrm{ipt}} = K\epsilon$ later), where $\epsilon = T_{\mathrm{ipt}}/K$ is the Gatling inter-proposal time. The proposal immediately following the failed one is ready ϵ after it, so it is held back by up to $T_{\mathrm{ipt}} - \epsilon = (K-1)\epsilon$, and its j-th successor by $(K-j)\epsilon$.

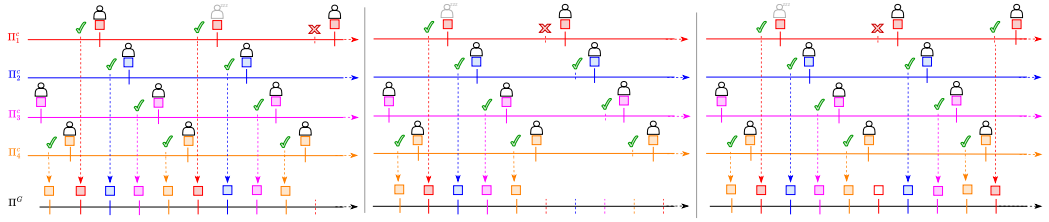


Figure 3 Behavior of Gatling in the presence of an adversarial proposer. Left: Each component instance assigns a leader per slot; confirmed blocks (green checkmarks) are immediately appended to the Gatling log (bottom), if all leaders are honest. If a leader is adversarial (second leader of the red instance, semitransparent), no block is confirmed for that slot (red cross). Center: The other instances (blue, magenta, orange) continue confirming blocks independently, but Gatling delays including them in the log because it waits for the red instance to reach consensus on the adversarial slot. Right: Once the red instance progresses under a subsequent honest leader and confirms a proposal, Gatling resumes extending the log, appending an empty block for the adversarial slot together with the pending confirmed blocks from the other instances.

This section gives an intuition of how to identify the Gatling's optimal inter-proposal time ϵ^* and, in turn, the optimal number of parallel instances K^*, that minimize the expected transaction latency in the presence of proposals that fail to confirm. The full analysis appears in the extended version of this work [40].

Stochastic Latency Model. The $(K-1)\epsilon$ value above assumes a single failure resolved at the next slot; expected latency instead depends on how often failures occur and on how much slack separates consecutive positions, which is what we model here. Optimizing ϵ involves a tension: packing proposals closer together (ϵ small) shortens the time a transaction waits for the next available proposal, but it also shortens the head start that each proposal's predecessors enjoy, thus holding up a longer queue of otherwise finalized proposals. When proposals are far apart, a slow component instance has plenty of time to decide before the proposals behind it are ready to be appended, and blocking is rare. Therefore, expected end-to-end latency is the sum of two terms that move in opposite directions as ϵ changes: the wait for inclusion, which grows with ϵ, and the delay caused by head-of-line blocking, which shrinks with ϵ.

To locate the sweet spot for ϵ, we adopt a stochastic model of component instances' behaviour: each instance attempts to decide its slot within the good-case confirmation time T_{conf}, each attempt succeeds with probability s independently of all other attempts and of all other slots and instances, and a failed attempt is followed by a new attempt on the next slot. The resulting per-slot decision time is geometric: a slot is decided after one attempt, or two, or three, so its distribution is a staircase that jumps at each multiple of T_{conf}. Those jumps obstruct a closed form. We therefore approximate the staircase by a smooth exponential curve shifted by T_{conf}, whose rate is fixed by requiring that it passes through the staircase at every attempt boundary. The curve is shifted by T_{conf}, because no slot can be decided before one attempt has elapsed. The approximation is exact at the boundaries and interpolates in between. It also turns the expected-latency expression into a classical q-series, which admits a closed-form optimization.

The next theorem shows that this trade-off has a single sweet spot, and gives both the optimal inter-proposal time and the resulting latency in closed form. For the full analysis that led to the theorem, as well as its proof, we refer to the extended version of this work [40].

▶ **Theorem 9** (Optimal Gatling Inter-Proposal Time). *Consider the model above, with per-attempt failure probability $p := 1 - s$ and smooth-approximation rate $\lambda_p := -\log p / T_{\text{conf}}$, and let $L(\epsilon)$ denote the resulting expected end-to-end transaction latency, comprising the wait for the next available proposal and the time from that proposal until it is appended to the Gatling log. Then L has a unique minimizer*

$$\epsilon^*(s) = -\lambda_p^{-1} \log q_p^* = T_{\text{conf}} \cdot \frac{-\log q_p^*}{-\log p}, \tag{1}$$

where $q_p^* \in (0,1)$ is the unique solution to

$$\int_0^1 (px;\, q_p^*)_\infty \sum_{j=1}^{\infty} \frac{p\, j\, (q_p^*)^j}{1 - px(q_p^*)^j}\, dx \;=\; \frac{1}{2} \tag{2}$$

and $(a;q)_\infty := \prod_{m\geq 0}(1 - aq^m)$ denotes the q-Pochhammer symbol. The corresponding minimum expected latency is

$$L^*(s) \;=\; T_{\text{conf}} + \lambda_p^{-1}\, \phi_p(q_p^*), \qquad \textit{with} \qquad \phi_p(q) \;:=\; -\tfrac{1}{2}\log q + \int_0^1 \frac{1 - (px;\, q)_\infty}{x}\, dx. \tag{3}$$

The theorem is stated in terms of q_p^* because the q-series form makes the optimization tractable; q is the decay of the smooth exponential curve over one inter-proposal time, and thus a reparametrization of ϵ. Equation (1) inverts it, giving an optimal ϵ^* that is a multiple of T_{conf} with a factor depending only on s. Equation (2) is the tension we described above, written as a balance. Its left-hand side is the rate at which increasing ϵ relieves head-of-line blocking, aggregated over all predecessors, and the $\frac{1}{2}$ on the right is the rate at which it grows the wait for inclusion, since a transaction waits on average $\epsilon/2$ for the next available proposal. The optimal ϵ^* is where the two rates meet. Finally, Equation (3) splits the achievable latency into the component protocol's good-case confirmation time T_{conf} and an overhead term that captures the cost of parallelism, which depends on s.

Numerical Results. We obtain q_p^* by numerically solving Equation (2) and then evaluate the closed-form expressions for $\epsilon^*(s)$ and $L^*(s)$ in Thm. 9. Fig. 4 showcases $\epsilon^*(s)$ and $L^*(s)$ in units of Δ as functions of the per-attempt success probability s. As $s \to 1$, the optimal inter-proposal time $\epsilon^*(s)$ shrinks to zero while the minimized expected latency $L^*(s)$ converges to T_{conf} (here 3Δ): Gatling recovers the component protocol's good-case confirmation latency with arbitrarily small inter-proposal time. For example, at $s = 0.99$ the optimal global inter-proposal time is $\epsilon^* \approx 0.09\Delta$ and the corresponding expected end-to-end latency is $L^* \approx 3.09\Delta$. At $s = 0.999$ these become $\epsilon^* \approx 0.02\Delta$ and $L^* \approx 3.02\Delta$. Translating ϵ^* into the design parameter K of Sec. 3, a Gatling protocol that runs parallel instances of a component protocol with inter-proposal time T_{ipt} should choose a number $K^*(s) = \lceil T_{\text{ipt}}/\epsilon^*(s) \rceil$ of parallel instances. Larger values of K increase head-of-line blocking without further reducing latency.

6 Transaction Deduplication

Since Gatling forwards every transaction to all K component protocol instances (Alg. 1, lns. 20 and 21), the same transaction may appear multiple times in the Gatling output log. As observed for multi-proposer protocols [39, 24, 47], duplicate transactions waste block space and reduce the effective throughput of the protocol. One might ask whether forwarding

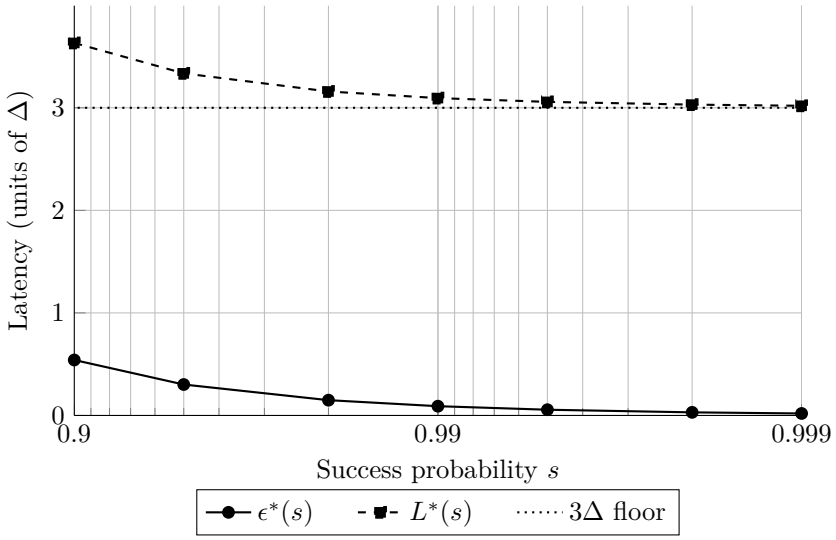


Figure 4 Optimal Gatling global inter-proposal spacing $\epsilon^*(s)$ and minimized expected end-to-end finality latency $L^*(s)$ under the model of Thm. 9.

each transaction only to the component instance next in line to propose, rather than to all K instances, would avoid duplicates and thereby improve what is sometimes called *goodput*. For practical purposes, this is indeed the case. Modern blockchains frequently rely on private mempools or relay networks, which disseminate transactions rapidly and preferentially to upcoming leaders, effectively ensuring that transactions are promptly available to the next leader to propose. In turn, for most consensus protocols, it suffices for liveness if an honest leader includes a transaction in its proposed block.

From a theoretical perspective, Def. 2 guarantees liveness of the component protocols only if a transaction is eventually input by all honest nodes. This requires that for Gatling liveness, honest nodes forward input transactions to all component protocols. On the other hand, strengthening the liveness guarantee so that input by one rather than all honest nodes suffices for liveness, would introduce complications in the definition of latency: under the current Def. 2, the latency clock starts when all honest nodes have broadcast the transaction, ensuring that the next leader has already received it. If the clock instead starts when only one honest node has broadcast the transaction, the transaction may still need to propagate to the next leader or wait for the node that broadcast it to become a leader, potentially adding up to the effective transaction latency. The theoretical latency metric then contains terms that in reality are not there due to how transaction dissemination and consensus are handled in modern systems.

7 Enhancing Gatling with Predictable Validity

An attentive reader may have observed that when Gatling is run with an inter-proposal time smaller than Δ and with successive blocks being proposed by different leader nodes (rotating leaders), a block does not have enough time to propagate from one leader to the next before the following proposal is issued. As a result, a leader cannot validate the transactions in its proposal against the most recent state of the log. Under these conditions, Gatling no longer satisfies a property commonly referred to as predictable validity [36, 37, 29]. Predictable validity guarantees that transactions deemed valid by a leader at proposal time

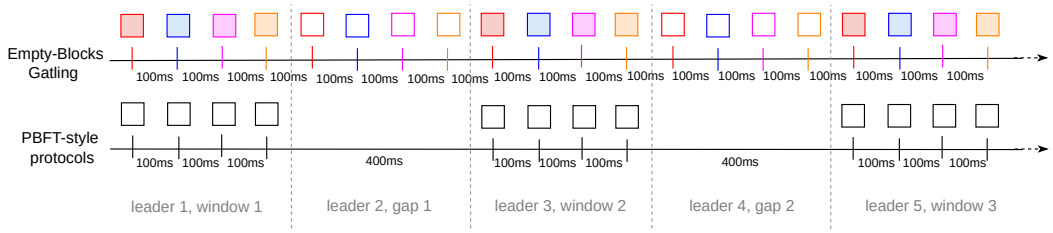


Figure 5 Comparison between the empty-blocks Gatling (top) and a PBFT-style protocol (bottom), both instantiated with a slowly rotating leader schedule. In this picture, $K = 4$, $\Delta = 400$ms, and the Gatling inter-proposal time is 100ms. The PBFT-style protocol proposes multiple (non-empty) blocks during a leader's window and separates consecutive windows by at least Δ to ensure block propagation and, therefore, predictable validity. The empty-blocks variant of Gatling reproduces this behavior by alternating proposal windows with explicit Δ-sized gaps with empty proposals.

remain semantically valid when they are eventually executed. Under predictable validity, proposers can therefore reason soundly about execution outcomes using only the locally available state. While most popular blockchain protocols such as Bitcoin, Ethereum, and Solana fulfill predictable validity, DAG-based logs such as Sui [9] and Aptos [46] as well as "lazy" logs such as Celestia [14] and Pod [3] have forgone it in favor of lower latency. For blockchains that want to preserve predictable validity while retaining the benefits of reduced inter-proposal time, we present two variants of Gatling– *empty-blocks* and *subprime-blocks* – with inter-proposal times smaller than Δ, operating under slowly rotating leaders.

7.1 Empty-Blocks Gatling Variant

Consider a Gatling protocol that achieves a sub-Δ inter-proposal time $\frac{T_{\text{ipt}}}{K}$ by running K parallel instances of a component protocol, each configured with an inter-proposal time of T_{ipt}. Time is partitioned into a sequence of contiguous slots, called windows, of length at least Δ. Each window is assigned to a randomly selected leader node which, within its assigned window, can issue a fixed number of consecutive proposals. For Gatling to achieve this slowly rotating leader schedule, the leader schedules of the component protocols are coordinated so that a single leader can issue K consecutive Gatling blocks by proposing in K consecutive component protocol instances. To enable this coordination, each atomic broadcast component protocol exposes an interface that accepts the proposer schedule as an explicit input, in addition to the sequence of proposal times.

Within a window, a sub-Δ inter-proposal time does not compromise predictable validity: because the same leader issues multiple consecutive proposals, it can locally maintain and update a consistent view of the log across proposals. Hence, when building a new proposal, the leader can validate the transactions against the most recent state of the log. The challenge arises at window boundaries. Since Gatling is instantiated with a sub-Δ inter-proposal time T_{ipt}, less than Δ time elapses (at least) between the last proposal of one window and the first proposal of the subsequent window. Consequently, the leader of the subsequent window has to issue a proposal(s) before having received one or more proposals of the previous window.

To preserve predictable validity across windows, this variant of Gatling introduces a "window-gap-window-gap" structure, with windows occurring in strict alternation with gaps of at least Δ (Fig. 5). In a window, the assigned leader includes a non-empty payload in all of its proposals. In contrast, in a gap, proposals carry empty payloads that do not modify the state of the log. These gaps provide sufficient time for proposal propagation between



Figure 6 Subprime-blocks variant of Gatling that restores predictable validity at execution. At the top, the consensus confirmation ordering. At the bottom, we illustrate two possible orderings in which blocks are executed: in one (top), blocks of windows and gaps are executed contiguously, with a window being executed before the preceding gap. In the other (bottom), for each window and the preceding gap, every block from the window is executed before the corresponding block from the gap.

consecutive windows, allowing the leader responsible for the next window to observe the preceding blocks. This ensures predictable validity. The empty proposals could be issued similarly to the ones in a window, or they could be issued by a leader so that all proposals are received by the next leader before it begins issuing proposals for its window (e.g., unique slot proposal time T^{prop} at the beginning of the window for all proposals). Alternatively, the protocol could mandate nodes to implicitly insert empty proposals in their local logs, without requiring explicit consensus on those (no QCs).

Fig. 5 illustrates that the empty-blocks variant of Gatling (top of the figure) closely mirrors the proposal pattern of traditional PBFT-style protocols with slowly rotating leaders (bottom of the figure). In PBFT-like protocols, a leader proposes multiple blocks within a window, and successive windows are separated by at least Δ time to allow blocks to propagate, thereby preserving predictable validity. Similarly, this Gatling variant alternates between proposal windows and Δ-sized gaps: proposals are issued at sub-Δ intervals within a window, while the intervening gaps ensure sufficient time for propagation. Gatling provides a general black-box composition framework for atomic broadcast that encompasses stable and slowly rotating leader protocols, such as PBFT and Alpenglow, as special cases. By adjusting its parameters, Gatling can reproduce their execution patterns, while also enabling alternative operating regimes in which proposals are issued at sub-Δ inter-proposal times and consistently carry transactions, trading predictable validity for higher proposal rates.

7.2 Subprime-Blocks Gatling Variant

While the empty-blocks variant of Gatling already captures many of the desired properties (sub-Δ inter-proposal times and predictable validity), the need to validate transactions against the most recent state introduces gaps that periodically stall progress and increase the inter-proposal time of non-empty blocks. To address this limitation, we introduce an advanced variant of Gatling named *subprime-blocks* Gatling – illustrated in Fig. 6 – that preserves predictable validity while eliminating the stalls.

The key observation is that empty blocks proposed in the gaps of the previous design can be repurposed as a distinct tier of block space, which we refer to as *subprime block space*. This tier is reserved for transactions that are not time-critical and are insensitive to ordering races, such as simple value transfers or layer-2 data blob transactions. In contrast, the *prime block space* tier is reserved for high-priority transactions – such as DeFi interactions – that

are sensitive to execution order and frequently participate in front-run races. In this variant, the designated leader of a window makes proposals for prime block space, by only including time-critical transactions that access highly contentious state. Instead, the designated leader of a window of a gap makes proposals for subprime block space, by only including non-time-critical transactions that access non-contentious state.

Additionally, we modify the interaction between the consensus and execution layers by allowing the log produced by consensus to differ from the log consumed by the execution layer. As per Fig. 6, consider window number 3, followed by gap number 3, followed by window number 4, followed by gap number 4, and so on. While the Gatling log output by consensus is the ordered sequence of blocks proposed in subsequent windows and gaps, the log consumed by the execution layer always executes a window before the immediately preceding gap. For example, the execution layer processes blocks from window 3, then blocks from gap 2, then blocks from window 4, followed by blocks from gap 3, and so on (see the execution order in Fig. 6). Because each gap has duration at least Δ, when the leader of a window issues its first proposal, all blocks that will be executed before it must have been proposed at least Δ time earlier and therefore already received and executed by that leader. Consequently, proposers in windows always construct their proposals on top of a known and up-to-date log state. In contrast, the leaders proposing during gaps have no knowledge of the state at the time their subprime blocks will be executed, since their blocks will be executed after the blocks proposed in the next window. Subprime-block transactions are not guaranteed to execute successfully. At proposal time, they only need to be well formed and pass all validity checks that do not depend on the eventual execution state (e.g., correct format, valid signatures, resource bounds, declared dependencies). State-dependent validity is evaluated later at execution time against the state determined by Gatling's execution order; if the transaction is invalid against that state, it (soft-)fails without invalidating the block. We can further refine the execution sequence by ordering blocks individually rather than executing all blocks from windows and gaps contiguously, as illustrated in the optimized execution ordering of Fig. 6: execution alternates between blocks from a window and blocks from the preceding gap. For instance, the first block of window 3 is executed first, then the first block of gap 2; next, the second block of window 3, followed by the second block of gap 2, and so on until all blocks of window 3 and gap 2 have been executed. The same alternating pattern is applied to window 4 and gap 3, etc.

In the variant of Sec. 7.1, nodes execute blocks only in during windows, while during gaps they wait for new blocks. This subprime-blocks Gatling variant, besides higher proposal rates, also offers a more uniformly distributed execution load.

8 Evaluation

We implemented a proof-of-concept of Gatling to measure its reduced transaction latency and its practicality in a realistic environment. Our experiment confirms the theoretical findings in Sec. 5. Our implementation is available at: `https://github.com/scaffino/Gatling`.

Implementation Overview. Our implementation extends Commonware's alto blockchain [19], a minimal but complete Rust implementation of a blockchain that relies on the Commonware monorepo [20], a collection of production-grade primitives including consensus, networking, and cryptographic libraries. We augment alto by enabling each validator to run $K \geq 1$ independent instances of the Simplex consensus. Each instance maintains its own log, follows an independent leader schedule, and communicates over dedicated network channels.

Importantly, each instance has an inter-proposal time of $T_{\text{ipt}} = 500$ms and is configured to be non-responsive. We set the leader deadline to 225ms (the time by which the validators must have received the block; otherwise, they will cast a skip vote) and the notarize deadline to 375ms (the time by which a validator must have received a notarize certificate to cast a finalization certificate). We implement the Gatling log reconstruction and confirmation rules described in Sec. 3, allowing the outputs of these parallel instances to be deterministically merged into a single global log.

We do not model transaction dissemination via gossip. Instead, transactions arrive according to a Poisson process at a rate of approximately 100 transactions per second. This transaction rate is below the system's saturation point and therefore does not introduce queuing effects that could distort latency measurements. Transactions are delivered directly to the validator expected to propose the next block, effectively modeling instantaneous transaction submission and emulating the private-mempool behavior commonly observed in real-world blockchain deployments. We note that when the Gatling inter-proposal time ϵ is sufficiently small relative to the global network delay, bypassing the designated next leader may reduce latency when that leader is geographically distant.

To emulate crash failures, we randomly suppress proposals with probability p. Varying p allows us to evaluate the effect of proposal failures on head-of-line blocking and transaction latency.

Measuring Latency. We deployed Gatling on a global 10 validator cluster with 1 node in London, 3 in Tokyo, 3 in Singapore, 2 in Dallas, and 1 in Miami. The largest latencies in the cluster were around 110-112ms between validators in Miami and Singapore. We compute the transaction latency for varying values of K. We denote by skip rate the percentage of slots that resulted in a skip vote, i.e., the leader missed the 225ms deadline, aggregated over all instances during the measurement window. Fig. 7 presents the transaction latency as a function of K, measured across the three proposal-drop rate regimes: 0% (top left), 1% (top right), 5% (bottom left), and 10% (bottom right).[2] Each bar is a stacked bar chart decomposing total latency into its two components: the wait time (blue), and the confirmation time (orange).

In the good case, where no proposal failures occur, increasing K reduces the inter-proposal time and thus lowers transaction waiting time. As a consequence, we expect the transaction latency to decrease monotonically with larger values of K. This is confirmed by Fig. 7; for $K = \{15, 40\}$ the average transaction latency shows a slight increase due to anomalously high skip rates. The 0% proposal-drop rate in Fig. 7 also reveals that end-to-end latency stabilizes around 215ms, consistent with the theoretical 3Δ confirmation time floor for $\Delta \approx 70$ms.

In the presence of proposal failures, transaction latency is expected to exhibit a non-monotonic dependence on K: it initially decreases as the inter-proposal time shrinks, but eventually increases as head-of-line blocking becomes the dominant source of delay (Sec. 5). This is consistent with Fig. 7: the transaction latency is minimized at $K = 9$ for the 1% drop rate, at $K = 4$ for the 5% drop rate, and at $K = 3$ for the 10% drop rate. In the right panel of Fig. 7 the blue component shrinks steadily from $K = 1$ to $K = 9$, and total latency falls from 447ms to a minimum of 244ms at $K = 9$. Beyond $K = 9$, however, a 1% proposal-drop probability is sufficient to trigger head-of-line blocking frequently enough that the confirmation time wait (orange component) grows, pushing total latency back up to 306ms at $K = 10$. The center and left panels of Fig. 7 illustrate how reliability governs the optimal K. With a 5% drop rate (center), the optimal number of instances shifts to

[2] The corresponding numerical results appear in the extended version of this paper [40].

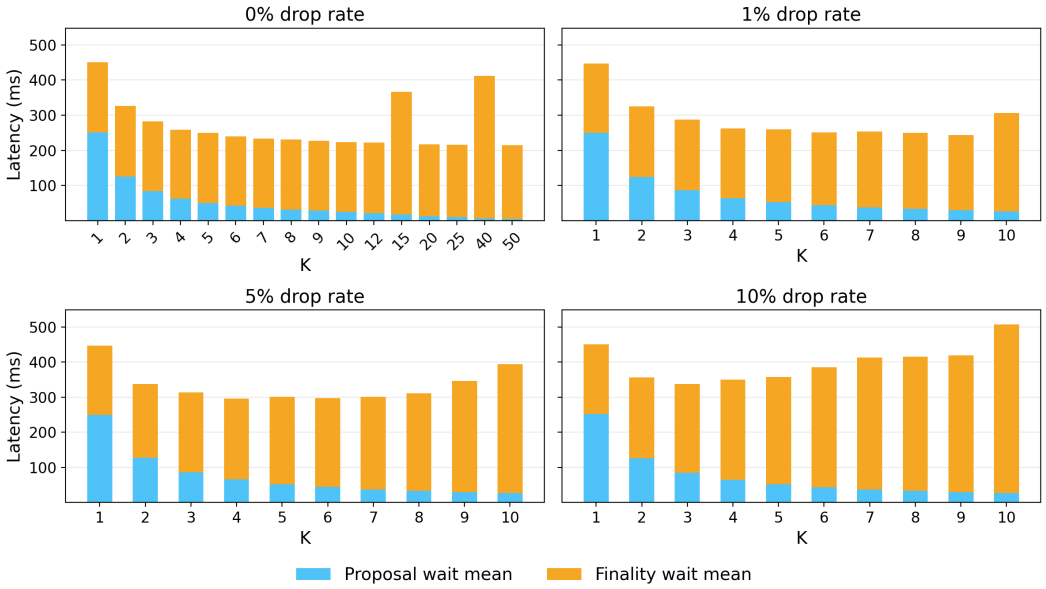


Figure 7 Transaction latency as a function of K across three proposal-drop rate regimes: 0% (top left), 1% (top right), 5% (bottom left), and 10% (bottom right). Each bar decomposes total latency into the average time transactions wait to be included in a proposal (proposal wait mean, blue), and the average time it takes for them to be confirmed after inclusion (finality wait mean, orange). The more reliable the validator set is, the more concurrent instances can be tolerated without introducing additional delay from head-of-line blocking.

$K = 4$, yielding a minimum latency of 296ms, while with a 10% drop rate the optimum is at $K = 3$ with 337ms. In both cases, the effect of head-of-line blocking increases with K. This confirms the core tradeoff identified in Sec. 5.

Our experiments also demonstrate that a blockchain can rotate leaders every $\epsilon = 50$ms or less on a global cluster where communication time between nodes exceeds 50ms.

Consistency with the Theoretical Optimum. The model of Sec. 5 makes three predictions about end-to-end latency as a function of the Gatling inter-proposal time ϵ; equivalently, as a function of the number of parallel instances $K = T_{\text{ipt}}/\epsilon$. First, the latency curve has a unique minimum at some $\epsilon^*(s)$. Second, the corresponding optimum $K^*(s)$ shifts toward smaller K as the per-attempt failure probability $p = 1 - s$ grows. Third, as $s \to 1$, the minimum expected latency $L^*(s)$ approaches the single-attempt confirmation floor T_{conf}. All three predictions are visible in Fig. 7: each panel exhibits a clear U-shape with a minimum; the optimum shifts from $K = 9$ at the 1% drop rate to $K = 4$ at the 5% drop rate and $K = 3$ at the 10% drop rate; and the minimum observed latency falls monotonically as the drop rate shrinks, from 337ms at 10% to 297ms at 5% and 244ms at 1%, approaching $T_{\text{conf}} \approx 197$ms, the time of one round of the component protocol, measured at $K = 1$ where head-of-line blocking is absent.

9 Related Work

Responsive Protocols and Timing Games. An important line of work in distributed systems and blockchains has focused on a class of protocols, called *responsive*, which improve latency by reducing dependence on worst-case timing assumptions [1, 7]. Protocols such as

Tendermint [10] and Streamlet [16] advance according to fixed delays chosen to tolerate the maximum network delay (Δ), causing nodes to wait for entire Δ timeout periods even when the actual message delay (δ) is much smaller.

In contrast, responsive protocols such as PBFT [13], Simplex [15], HotStuff [52], HotStuff-1 [27], HotStuff-2 [32], Jolteon [25], and Moonshot [22] are designed so that progress is driven by message arrivals rather than by timeout expirations, allowing nodes to reach a decision as quickly as the network conditions permit. As pointed out in [1, 38, 26], responsiveness is execution-dependent and cannot be guaranteed in all runs: when a leader is adversarial, correct nodes may be forced to delay commitment, reintroducing latency that is independent of the actual network conditions. As a result, responsiveness holds only for a subset of optimistic executions: those with a correct leader and sufficiently many honest participants.

For a long time, the design of responsive protocols has been carried out without considering timing incentives of the nodes. Only recently, the work of [41, 17], analyzed the incentives of responsive protocols in the blockchain setting, showing that when elected as proposer the best strategy for profit-maximising nodes is to delay the proposal for as long as possible, while still ensuring the timely inclusion in the ledger. This strategic delay in block proposals, often called timing game, is exploited to acquire additional information about pending transactions and optimize the Maximal Extractable Value (MEV) capture. Timing games have been observed in major blockchains such as Ethereum and Solana. Designing responsive protocols that explicitly account for timing games and strategic proposer behavior was first explored by [4]; however, this composition remains largely underexplored and represents a promising direction for future work. Our work improves latency via a mechanism that is orthogonal and complementary to the one of responsive protocols. While responsive protocols achieve low latency under favorable network delays (δ), Gatling attains arbitrarily small inter-block times using only conservative worst-case assumptions (Δ) in its component protocols, providing latency improvements independent of optimistic network conditions.

Pipelined and Optimistic Protocols. In traditional BFT-style protocols, block proposals are strictly sequential: a leader must wait for the previous proposal to be confirmed through multiple voting phases before issuing the next proposal. PBFT [13], for example, proceeds through a leader proposal phase (pre-prepare) followed by two voting phases (prepare, commit), resulting in a "propose-vote-vote" structure per proposal. With Casper-FFG [12] and then the HotStuff family of protocols [52, 27, 32], a *pipelined*, *chain-based* formulation is introduced: proposals are cryptographically linked to each other in a chain, and each proposal carries a quorum certificate attesting that a quorum of nodes has agreed on the previous proposal. This structure allows consensus phases to be pipelined: while nodes vote to form a certificate for the current proposal, the leader can already issue a new proposal. Conceptually, this results in an alternating "propose-vote-propose-vote" pattern, where making a new proposal is overlapped with voting on the previous one.

A complementary line of work explores optimistic/speculative techniques that further reduce the number of network round trips to reach confirmation in BFT consensus protocols. For instance, Moonshot [22] and Hydrangea++ [44] adopt an *optimistic proposal* approach allowing leaders to propose new blocks before the previous block has been quorum-certified, overlapping proposal and voting phases. This reduces latency under the assumption of consecutive honest leaders. This design effectively results in a "propose-propose-propose" structure, enabling more aggressive proposal pipelining. We observe that for $K = 2$, Gatling recovers the behavior of Moonshot in the optimistic case: when the leader is honest and includes a transaction in its block, and the previous leader was also honest, then that transaction is viewed as confirmed by everyone 3Δ later, even if at most f nodes are adversarial.

A different approach is taken by HotStuff-1 [27]: next to the traditional HotStuff-style commit rule, in the optimistic case, it introduces a speculative technique that reduces commit latency by sending clients early execution responses after a single quorum certificate, without waiting for the usual multi-phase commit rule.

Carnot [30] obtains block times below the network delay by adopting a multi-view leader schedule and it shows that, under bounded bandwidth, the equilibrium proposal interval is limited by transaction dissemination and proposal metadata transmission. Unlike Gatling, which supports rapidly rotating leaders through parallel composition and derives its optimal instance count from head-of-line blocking, Carnot employs a stable leader schedule and uses erasure coding to reduce the bandwidth bottleneck of the leader. Carnot's bandwidth analysis is complementary to ours and would impose an additional finite optimum on Gatling's number of instances in a bandwidth-constrained network.

Fast-Path Protocols. A long line of work that started with [34] has explored *fast-path* techniques that reduce latency by decreasing the number of voting rounds required to confirm a block, trading resilience for lower round complexity. Classical BFT protocols require two rounds of voting to tolerate up to f Byzantine faults, ensuring safety even under adversarial conditions. Fast-path designs relax these guarantees under optimistic assumptions – such as honest leader and a reduced number of adversarial nodes – to decide on a proposal with a single round of voting. This approach reduces transaction latency when the assumptions hold, but retain a fallback slow-path with two rounds of voting that restores full fault-tolerance when the optimistic assumptions are violated. Recently, there has been renewed interest in fast-path designs in blockchains and distributed systems which has led to the designs of Minimmit [18], Alpenglow [5], Kudzu [43], and Hydrangea [45]. While fast-path protocols reduce latency by weakening fault-tolerance assumptions in optimistic executions, Gatling preserves full resilience while improving block production rates through composition. As such, fast-path techniques are complementary to Gatling rather than competing: they could be layered on top of the component protocols to further reduce Gatling's confirmation latency.

Multiple Concurrent Proposals. Protocols that leverage multiple concurrent proposals (MCP) have recently gained renewed attention in the Ethereum and Solana communities, notably through the Ethereum's EIP-7805 (FOCIL) [50] and this recently proposed protocol [24]. MCP approaches allow multiple nodes to simultaneously propose batches of transactions for inclusion in the next block; while each round still produces a single block, that block aggregates contributions from several proposers, thereby reducing the leader's monopoly over transaction inclusion and ordering. While Gatling reduces the inter-proposal time, MCP protocols [24] require multiple rounds of communication among nodes when a proposal is constructed, which leads to longer inter-proposal times.

In DAG-based protocols [33, 28, 47, 21, 46, 8], in each round, every node can independently create and broadcast a proposal that back-references to typically $2f + 1$ proposals from the previous round, forming a directed acyclic graph (DAG). These protocols exploit parallelism by allowing multiple leaders to be active concurrently and they can be viewed as a form of parallel composition, in which multiple ordering streams with different leader schedules operate concurrently over a shared DAG. Unlike Gatling, however, these streams remain coupled through the DAG's reference structure, which imposes dependencies and constrains progress to the network delay.

Comparing Gatling with Multi-BFT Protocols. Recent work on Multi-BFT protocols explores running multiple PBFT instances in parallel to overcome the throughput limitations of single-leader designs.

For example, Mir-BFT [48] increases throughput by running multiple PBFT proposers in parallel, each responsible for a subset of log positions. To distribute work across proposers, transactions are partitioned into buckets and assigned according to a deterministic mapping. Confirmed proposals are merged into a single total order according to their assigned slot numbers. Mir-BFT further employs an epoch-based reconfiguration mechanism in which a designated epoch leader determines the assignment of transactions, slots, and ordering responsibilities across proposers. As a result, Mir-BFT's parallel streams share protocol state, including epoch configuration, slot ownership, and bucket ownership, and collectively realize a coordinated partitioning of a single global log. Gatling also exploits parallelism through multiple concurrent consensus instances, but does so through a black-box composition of atomic broadcast protocols. Each instance operates independently and need not share proposer assignments, transaction partitions, or reconfiguration state with other instances; the only coordination across instances is a deterministic merge rule applied to their outputs. Consequently, Gatling merges independent logs rather than by partitioning a single log. Furthermore, while Mir-BFT can achieve low effective inter-proposal times under sustained load as a consequence of having multiple active proposers, Gatling explicitly engineers a low and regular inter-proposal cadence by staggering proposal schedules across instances. This design enables reliable and arbitrarily small inter-proposal times, optimizing for low transaction latency.

ISS [49] generalizes Mir-BFT by introducing a protocol-independent framework for parallel consensus. Transactions are partitioned across multiple consensus instances, whose outputs are deterministically merged into a single replicated log. This abstraction removes the need for Mir's dedicated epoch leader and orderer-assignment mechanism while retaining the scalability benefits of parallel execution. Like ISS, Gatling composes multiple consensus instances and combines their outputs into a single log. However, ISS and Gatling target different goals. ISS uses parallelism to partition workload across independent streams and thereby increase throughput, whereas Gatling uses staggered proposal schedules to reduce transaction latency.

Ladon [31] follows a related approach, running parallel BFT protocols and deriving a single global order through monotonic block ranks. Leaders determine ranks using information collected across instances, and nodes confirm blocks according to rank-based thresholds. This cross-instance coordination distinguishes Ladon from Gatling in which the parallel instances operate independently as closed boxes.

Comparing Gatling with Shoal++. After completing this work, we were made aware that Shoal++ [6] uses a construction similar to Gatling in the context of DAG-based protocols to reduce the inter-proposal time. In particular, it runs three DAG instances in parallel, time-staggered by a fixed Δ offset, in a not-fully-generic "ajar-box" composition. By contrast, Gatling explores regimes with inter-proposal intervals below the network delay and introduces variants that preserve predictable validity even when the offsets become arbitrarily small. Moreover, Gatling is closed-box and works for general atomic broadcast component protocols, and our experiments evaluate the latency reduction solely attributable to time-staggered parallel composition.

References

1 Ittai Abraham and Kartik Nayak. What is responsiveness? https://decentralizedthoughts.github.io/2022-12-18-what-is-responsiveness/, 2022.

2 Ittai Abraham, Kartik Nayak, Ling Ren, and Zhuolun Xiang. Good-case latency of Byzantine broadcast: a complete categorization. In *PODC*, pages 331–341. ACM, 2021. doi:10.1145/3465084.3467899.

3 Orestis Alpos, Bernardo David, Jakov Mitrovski, Odysseas Sofikitis, and Dionysis Zindros. pod: An optimal-latency, censorship-free, and accountable generalized consensus layer. In *DISC*, volume 356 of *LIPIcs*, pages 4:1–4:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.DISC.2025.4.

4 Kaya Alpturer, Kushal Babel, and Aditya Saraf. Timing games in responsive consensus protocols, 2025. arXiv:2510.25144v1.

5 Anza. Alpenglow: A new consensus for solana. https://www.anza.xyz/blog/alpenglow-a-new-consensus-for-solana, 2024.

6 Balaji Arun, Zekun Li, Florian Suri-Payer, Sourav Das, and Alexander Spiegelman. Shoal++: High throughput DAG BFT can be fast and robust! In *NSDI*, pages 813–826. USENIX Association, 2025. URL: https://www.usenix.org/conference/nsdi25/presentation/arun.

7 Hagit Attiya, Cynthia Dwork, Nancy A. Lynch, and Larry J. Stockmeyer. Bounds on the time to reach agreement in the presence of timing uncertainty. *J. ACM*, 41(1):122–152, 1994. doi:10.1145/174644.174649.

8 Kushal Babel, Andrey Chursin, George Danezis, Anastasios Kichidis, Lefteris Kokoris-Kogias, Arun Koshy, Alberto Sonnino, and Mingwei Tian. Mysticeti: Reaching the latency limits with uncertified DAGs. In *NDSS*. The Internet Society, 2025. URL: https://www.ndss-symposium.org/ndss-paper/mysticeti-reaching-the-latency-limits-with-uncertified-dags/.

9 Sam Blackshear, Andrey Chursin, George Danezis, Anastasios Kichidis, Lefteris Kokoris-Kogias, Xun Li, Mark Logan, Ashok Menon, Todd Nowacki, Alberto Sonnino, Brandon Williams, and Lu Zhang. Sui Lutris: A blockchain combining broadcast and consensus. In *CCS*, pages 2606–2620. ACM, 2024. doi:10.1145/3658644.3670286.

10 Ethan Buchman. Tendermint: Byzantine fault tolerance in the age of blockchains. Master's thesis, University of Guelph, 2016. URL: https://atrium.lib.uoguelph.ca/items/5459099e-67aa-4a23-83ae-d3471d8d8336.

11 Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform. https://ethereum.org/whitepaper/, 2014.

12 Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget, 2019. arXiv:1710.09437v4.

13 Miguel Castro and Barbara Liskov. Practical Byzantine fault tolerance. In *OSDI*, pages 173–186. USENIX Association, 1999. URL: https://dl.acm.org/citation.cfm?id=296824.

14 Celestia. Celestia: Modular blockchain network. https://celestia.org/, 2023.

15 Benjamin Y. Chan and Rafael Pass. Simplex consensus: A simple and fast consensus protocol. In *TCC (4)*, volume 14372 of *Lecture Notes in Computer Science*, pages 452–479. Springer, 2023. doi:10.1007/978-3-031-48624-1_17.

16 Benjamin Y. Chan and Elaine Shi. Streamlet: Textbook streamlined blockchains. In *AFT*, pages 1–11. ACM, 2020. doi:10.1145/3419614.3423256.

17 ChorusOne. Post on X. https://x.com/ChorusOne/status/1955615856580301252, 2024.

18 Brendan Kobayashi Chou, Andrew Lewis-Pye, and Patrick O'Grady. Minimmit: Fast finality with even faster blocks, 2026. arXiv:2508.10862v1.

19 Commonware. Commonware alto. https://github.com/commonwarexyz/alto, 2024. GitHub repository.

20 Commonware. Commonware monorepo. https://github.com/commonwarexyz/monorepo, 2024. GitHub repository.

21 George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. Narwhal and Tusk: a DAG-based mempool and efficient BFT consensus. In *EuroSys*, pages 34–50. ACM, 2022. doi:10.1145/3492321.3519594.

22 Isaac Doidge, Raghavendra Ramesh, Nibesh Shrestha, and Joshua Tobkin. Moonshot: Optimizing block period and commit latency in chain-based rotating leader BFT. In *DSN*, pages 470–482. IEEE, 2024. doi:10.1109/DSN58291.2024.00052.

23 Cynthia Dwork, Nancy A. Lynch, and Larry J. Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, 1988. doi:10.1145/42282.42283.

24 Pranav Garimidi, Joachim Neu, and Max Resnick. Multiple concurrent proposers: Why and how. Cryptology ePrint Archive, Paper 2025/1772, 2025. URL: https://eprint.iacr.org/2025/1772.

25 Rati Gelashvili, Lefteris Kokoris-Kogias, Alberto Sonnino, Alexander Spiegelman, and Zhuolun Xiang. Jolteon and Ditto: Network-adaptive efficient consensus with asynchronous fallback. In *Financial Cryptography*, volume 13411 of *Lecture Notes in Computer Science*, pages 296–315. Springer, 2022. doi:10.1007/978-3-031-18283-9_14.

26 Amir Herzberg and Shay Kutten. Early detection of message forwarding faults. *SIAM J. Comput.*, 30(4):1169–1196, 2000. doi:10.1137/S0097539796312745.

27 Dakai Kang, Suyash Gupta, Dahlia Malkhi, and Mohammad Sadoghi. HotStuff-1: Linear consensus with one-phase speculation. *Proc. ACM Manag. Data*, 3(3):171:1–171:29, 2025. doi:10.1145/3725308.

28 Idit Keidar, Eleftherios Kokoris-Kogias, Oded Naor, and Alexander Spiegelman. All you need is DAG. In *PODC*, pages 165–175. ACM, 2021. doi:10.1145/3465084.3467905.

29 Lucianna Kiffer, Joachim Neu, Srivatsan Sridhar, Aviv Zohar, and David Tse. Nakamoto consensus under bounded processing capacity. In *CCS*, pages 363–377. ACM, 2024. doi:10.1145/3658644.3670347.

30 Andrew Lewis-Pye and Patrick O'Grady. The carnot bound: Limits and possibilities for bandwidth-efficient consensus, 2026. arXiv:2603.11797v3.

31 Hanzheng Lyu, Shaokang Xie, Jianyu Niu, Chen Feng, Yinqian Zhang, and Ivan Beschastnikh. Ladon: High-performance multi-BFT consensus via dynamic global ordering. In *EuroSys*, pages 226–242. ACM, 2025. doi:10.1145/3689031.3696102.

32 Dahlia Malkhi and Kartik Nayak. Extended abstract: HotStuff-2: Optimal two-phase responsive BFT. Cryptology ePrint Archive, Paper 2023/397, 2023. URL: https://eprint.iacr.org/2023/397.

33 Dahlia Malkhi, Chrysoula Stathakopoulou, and Maofan Yin. BBCA-Chain: Low latency, high throughput BFT consensus on a DAG. In *FC (1)*, volume 14744 of *Lecture Notes in Computer Science*, pages 51–73. Springer, 2024. doi:10.1007/978-3-031-78676-1_4.

34 Jean-Philippe Martin and Lorenzo Alvisi. Fast Byzantine consensus. *IEEE Trans. Dependable Secur. Comput.*, 3(3):202–215, 2006. doi:10.1109/TDSC.2006.35.

35 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, 2009.

36 Joachim Neu, Srivatsan Sridhar, Lei Yang, David Tse, and Mohammad Alizadeh. Longest chain consensus under bandwidth constraint. In *AFT*, pages 126–147. ACM, 2022. doi:10.1145/3558535.3559777.

37 Joachim Neu, Ertem Nusret Tas, and David Tse. The availability-accountability dilemma and its resolution via accountability gadgets. In *Financial Cryptography*, volume 13411 of *Lecture Notes in Computer Science*, pages 541–559. Springer, 2022. doi:10.1007/978-3-031-18283-9_27.

38 Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. In *DISC*, volume 91 of *LIPIcs*, pages 39:1–39:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.DISC.2017.39.

39 Alejandro Ranchal-Pedrosa, Benjamin Marsh, Lefteris Kokoris-Kogias, and Alberto Sonnino. Sedna: Sharding transactions in multiple concurrent proposer blockchains, 2025. arXiv:2512.17045v1.

40 Giulia Scaffino, Max Resnick, and Joachim Neu. Gatling: Rapid-fire consensus from parallel composition, 2026. arXiv:2606.18220v1.

41 Caspar Schwarz-Schilling, Fahad Saleh, Thomas Thiery, Jennifer Pan, Nihar Shah, and Barnabé Monnot. Time is money: Strategic timing games in proof-of-stake protocols. In *AFT*, volume 282 of *LIPIcs*, pages 30:1–30:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.AFT.2023.30.

42 Victor Shoup. Sing a song of Simplex. In *DISC*, volume 319 of *LIPIcs*, pages 37:1–37:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.DISC.2024.37.

43 Victor Shoup, Jakub Sliwinski, and Yann Vonlanthen. Kudzu: Fast and simple high-throughput BFT. In *DISC*, volume 356 of *LIPIcs*, pages 42:1–42:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.DISC.2025.42.

44 Nibesh Shrestha and Aniket Kate. Hydrangea++: Enhancing Hydrangea with optimistic proposals. https://supra.com/documents/hydrangea-plus-plus.pdf, 2025.

45 Nibesh Shrestha, Aniket Kate, and Kartik Nayak. Hydrangea: Optimistic two-round partial synchrony with improved fault resilience. Cryptology ePrint Archive, Paper 2025/1112, 2025. URL: https://eprint.iacr.org/2025/1112.

46 Alexander Spiegelman, Balaji Arun, Rati Gelashvili, and Zekun Li. Shoal: Improving DAG-BFT latency and robustness. In *FC (1)*, volume 14744 of *Lecture Notes in Computer Science*, pages 92–109. Springer, 2024. doi:10.1007/978-3-031-78676-1_6.

47 Alexander Spiegelman, Neil Giridharan, Alberto Sonnino, and Lefteris Kokoris-Kogias. Bullshark: DAG BFT protocols made practical. In *CCS*, pages 2705–2718. ACM, 2022. doi:10.1145/3548606.3559361.

48 Chrysoula Stathakopoulou, Tudor David, Matej Pavlovic, and Marko Vukolic. Mir-BFT: Scalable and robust BFT for decentralized networks. *J. Syst. Res.*, 2(1), 2022.

49 Chrysoula Stathakopoulou, Matej Pavlovic, and Marko Vukolic. State machine replication scalability made simple. In *EuroSys*, pages 17–33. ACM, 2022. doi:10.1145/3492321.3519579.

50 Thomas Thiery, Francesco D'Amato, Julian Ma, Barnabé Monnot, Terence Tsao, Jacob Kaufmann, and Jihoon Song. EIP-7805: Fork-choice enforced inclusion lists (FOCIL). Ethereum Improvement Proposals, 2024. URL: https://eips.ethereum.org/EIPS/eip-7805.

51 Anatoly Yakovenko. Solana: A new architecture for a high performance blockchain v0.8.13. https://solana.com/solana-whitepaper.pdf, 2018.

52 Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan-Gueta, and Ittai Abraham. HotStuff: BFT consensus with linearity and responsiveness. In *PODC*, pages 347–356. ACM, 2019. doi:10.1145/3293611.3331591.

The Consensus Number of Untraceable Cryptocurrencies

Christian Cachin
Institute of Computer Science, University of Bern, Switzerland

David Lehnherr
Institute of Computer Science, University of Bern, Switzerland

Juan Villacis
Institute of Computer Science, University of Bern, Switzerland

François-Xavier Wicht
Institute of Computer Science, University of Bern, Switzerland

Abstract

Privacy-preserving cryptocurrencies hide which account funds a transfer among a set of candidate accounts, called its *masking set*. To prevent the hidden account from being spent twice, classical schemes retain the candidates and append a *nullifier* that marks the spent account without revealing it. Their ledgers therefore grow with every transfer. Constant-state schemes instead consume and replace the entire masking set. Here, we study the synchronization cost of this second design.

We introduce the *constant untraceable asset transfer* (CUAT) object. A CUAT transfer atomically replaces every account in its masking set; hence two transfers with intersecting sets cannot both succeed. We capture this contention by a conflict graph on transfer invocations, whose edges join invocations with intersecting masking sets. In one-round protocols, the relevant sets form a clique at a critical configuration. In general protocols, only sets from opposite valency classes must intersect.

Under weak untraceability, which considers one transaction in isolation, CUAT has unbounded consensus number even for one-round protocols. Under strong untraceability, which considers the complete history, accounts in a common masking set must have equal incidence. This condition makes both the one-round consensus power and CUAT's consensus number grow quadratically with the masking-set size. Cyclic and projective-plane constructions establish the corresponding one-round lower bounds, while a recursive grid construction shows that CUAT's consensus number is exactly the square of the masking-set size. CUAT is also not starvation-free.

The full version studies the complementary linear untraceable asset transfer (LUAT) object, which retains its masking set and records a nullifier. Its state grows, but its consensus number is two for every masking-set size and under either untraceability notion, and it is starvation-free. Thus LUAT pays for untraceability in storage, whereas CUAT pays in synchronization and fairness.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Theory of computation → Concurrent algorithms; Security and privacy → Pseudonymity, anonymity and untraceability

Keywords and phrases Consensus number, untraceability, privacy-preserving cryptocurrencies, wait-freedom, masking sets, concurrent objects

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.5

Related Version *Full Version*: `https://eprint.iacr.org/2026/1503` [10]

Funding This work was supported by the Swiss National Science Foundation (SNSF) under grant agreement Nr. 219403 (Emerging Consensus) and by the Initiative for Cryptocurrencies and Contracts (IC3).



© Christian Cachin, David Lehnherr, Juan Villacis, and François-Xavier Wicht;
licensed under Creative Commons License CC-BY 4.0



1 Introduction

Cryptocurrencies enable peer-to-peer value transfer without trusted intermediaries. They maintain a replicated ledger that records transactions and the current state of all accounts. Public verifiability makes this ledger auditable, but also exposes a permanent record of users' financial activity. Pseudonymous addresses provide limited protection because transactions can be linked to one another and combined with off-ledger information. The resulting transaction graph may reveal payments, balances, business relationships, or personal associations. Privacy-preserving cryptocurrencies seek to retain public verification without making this financial history attributable to individual users.

One central goal is sender untraceability, which hides the sender of a transfer among a set of candidate accounts, its *masking set*. The difficulty is to prevent the hidden account from being spent twice. Classical systems retain every candidate and append a nullifier for the spent account. This preserves the masking set but makes the ledger grow with every transfer.

Another approach, used by *Quisquis* and related systems [16, 3, 29], replaces all accounts in a fixed-size masking set after every transfer. Only the current accounts and balances remain, but account replacement creates contention: two transfers using a common account cannot both succeed. Does this contention require consensus? How does the privacy requirement constrain its synchronization power? And can an eligible transfer be excluded indefinitely?

This tension is easy to miss when looking only at the cryptographic protocol. Both approaches hide one account among the same number of candidates, but they use those candidates differently. In the classical approach, the masking set is evidence used by a proof and remains available after the transfer. In the constant-state approach, it is also mutable ledger state: a successful transfer invalidates every account in the set and renders every proof about the old state stale. The mechanism that bounds storage thereby turns privacy choices into conflicts between otherwise independent users. We therefore ask:

> *What synchronization and progress costs arise when a cryptocurrency consumes its masking sets in order to keep the ledger state bounded?*

1.1 Our Contribution

We answer these questions in Herlihy's consensus hierarchy [23]. The consensus number of an object is the largest number of processes that can solve wait-free consensus using that object and atomic registers. We introduce the *constant untraceable asset transfer* (CUAT)[1] object, extending the asset-transfer abstraction of Guerraoui *et al.* [21]. A λ-CUAT transfer atomically replaces all λ accounts in its masking set while hiding the sender and the balance update. The object isolates the synchronization caused by masking-set replacement from the surrounding cryptocurrency protocol.

Consensus numbers measure the synchronization power inherent in the object, not the number of validators that every transaction must contact. A high consensus number means that a sufficiently contended use of the object can solve agreement among many processes. Disjoint operations may still proceed without coordination. This distinction is important for CUAT: the cost is determined by the intersections among masking sets, rather than by the total workload or the size of the validator set.

We distinguish two adversarial views. Weak untraceability hides the sender of each transfer in isolation. Under this notion, CUAT has unbounded consensus number, even when every process invokes only one transfer. Strong untraceability instead protects against

[1] Pronounced *Coo-at*.

an observer of the complete history. We characterize this notion by a uniform-incidence condition: accounts that occur in a common masking set must appear equally often in the history. This condition limits the conflict structures that a private execution may expose.

We determine the resulting consensus power in two protocol regimes. A *one-round* protocol models a payment attempt whose masking set is fixed before submission. For every $\lambda \geq 3$, its consensus number lies between $\lfloor \lambda^2/2 \rfloor$ and $\lambda^2 - \lambda + 1$; the upper bound is attained whenever $\lambda - 1$ is a prime power. General protocols may invoke CUAT repeatedly and choose later masking sets based on earlier outcomes. Their additional flexibility closes the gap: under strong untraceability, CUAT has consensus number exactly λ^2 for every $\lambda \geq 2$. Thus the full synchronization power of the object can be strictly larger than the contention exposed by a single payment attempt.

Finally, we connect these worst-case consensus bounds to execution behavior. For a fixed workload, the conflict graph determines both its minimum sequential depth and its maximum one-round throughput. The same conflict mechanism also precludes starvation-freedom under an unfair asynchronous scheduler: a process may complete every invocation yet never succeed. This establishes a formal denial-of-service limitation for constant-state masking-set replacement.

1.2 Technical Approach

Our analysis starts from the *conflict graph*. Its vertices are transfer invocations, and two vertices are adjacent when their masking sets share an account index. Non-adjacent transfers commute; adjacent transfers cannot both succeed. This graph separates the cryptographic content of a transaction from the concurrency structure relevant to consensus: once masking sets are fixed, only their intersections matter. We connect these intersections to consensus through the standard critical-state argument. At a critical configuration, every next step leads to a univalent configuration, and the processes split according to the decision value forced by that step. Pending transfers of opposite valency must conflict. Otherwise, the two steps commute and lead to indistinguishable configurations that must decide different values, which is impossible.

The resulting graph condition depends on the protocol. In a *one-round* protocol, each process invokes one transfer with a fixed masking set. Applying the critical-state argument with each process as the distinguished process forces all masking sets to pairwise intersect. Conversely, any such clique gives an attempt-and-adopt consensus protocol: the first transfer succeeds, every other transfer fails, and the losers adopt the winner's registered input. Thus one-round consensus power equals the largest admissible clique (Theorem 11). In a general protocol, processes may invoke several transfers and adapt their sets. The critical configuration then guarantees only a complete bipartite intersection between the two valency classes. This weaker structure permits more processes than any one-round protocol.

This distinction separates two interpretations of CUAT. The one-round regime models a payment attempt whose masking set is chosen before submission: each user gets one chance to contend on the object. The general regime asks for the full position of CUAT in the consensus hierarchy: a protocol may use several objects, observe earlier outcomes, and choose later masking sets adaptively. The two regimes are governed by different subgraphs (cliques in the first, complete bipartite graphs in the second) and consequently have different bounds.

We first consider weak untraceability. It considers one transaction in isolation and places no global constraint on masking-set reuse. We put one hub account in arbitrarily many masking sets. They form an arbitrarily large clique, while the cryptographic proof hides the sender within each individual set. The attempt-and-adopt construction therefore solves consensus for any number of processes, already in one round: $\mathsf{cons}(\mathsf{CUAT}^{-}_{\lambda}) = \infty$.

The hub construction also explains why the weak notion is insufficient for a history-aware observer. The hub appears in every masking set and therefore becomes recognizable as a likely decoy, even though each zero-knowledge proof reveals nothing about the sender of its transaction in isolation.

Strong untraceability changes the picture because it lets the adversary observe the complete history. Repeated appearances then reveal information that is invisible in an isolated transaction. For an adversary that treats account appearances as independent, we show that it holds exactly when two accounts in a common masking set have the same incidence in the history (Theorem 14); the direction our bounds use, that strong untraceability forces equal incidence, holds against any stronger adversary. For a connected masking-set family, every occurring account consequently has the same incidence (Corollary 15). This converts a probabilistic privacy condition into a combinatorial constraint on the conflict graph, and is what makes the consensus bounds finite.

For one-round protocols, the masking sets are both uniformly incident and pairwise intersecting. If every account has incidence m, distinct ownership and double counting give $m \leq \lambda$. Fixing one masking set, each of its λ accounts occurs in at most $\lambda - 1$ other sets; hence the clique has size at most $1 + \lambda(\lambda - 1) = \lambda^2 - \lambda + 1$. A projective plane meets this bound when $\lambda - 1$ is a prime power. For every $\lambda \geq 3$, a cyclic difference cover gives a clique of size $\lfloor \lambda^2/2 \rfloor$.

These constructions also explain the shape of the bound. Projective planes distribute intersections perfectly: every pair of masking sets meets once and every account occurs equally often. Cyclic difference covers retain uniform incidence for every λ, at the cost of a smaller family.

The general case uses the weaker bipartite intersection exposed by a critical configuration. Let the two valency classes have sizes s_0 and s_1. Every one of the $s_0 s_1$ cross-class pairs must share an account. If an account occurs i_a times on one side and j_a times on the other, it covers $i_a j_a$ pairs. Uniform incidence fixes $i_a + j_a = m \leq \lambda$. Summing over accounts and applying Cauchy–Schwarz yields $s_0 + s_1 \leq \lambda m \leq \lambda^2$. We match this bound recursively. The base protocol lets λ processes contend on one masking set. To combine protocols for λu and λw processes, we place the two groups on the rows and columns of a $\lambda \times \lambda$ account grid. Every row meets every column, each account has incidence $u + w$, and the winning group carries one value from its recursive protocol. Taking $u + w = \lambda$ gives $\mathsf{cons}(\mathsf{CUAT}^+_\lambda) = \lambda^2$ for every $\lambda \geq 2$.

The grid construction is recursive because a process first agrees with the members of its own group and then uses a row or column to choose between the two group values. The groups need not have equal size. This is essential when λ is odd and is why the exact quadratic bound holds without a parity or prime-power restriction.

Finally, we translate the conflict structure back to system behavior. The consensus number is a worst-case measure of contention, not a requirement to order every transfer. Disjoint masking sets commute and can be processed independently. For a fixed workload, the conflict graph makes this precise: its chromatic number is the minimum sequential depth, and its independence number is the maximum one-round throughput. Conflict also prevents fairness. An unfair scheduler can repeatedly commit one process's zero-value transfer before a competitor on the same masking-set lineage, making every invocation of the competitor fail. Thus CUAT is not starvation-free, formalizing a conjectured denial-of-service behavior of *Quisquis* [9].

Together, these results identify a three-way connection between privacy, storage, and synchronization. Increasing λ enlarges the anonymity set, but under strong untraceability it also permits a conflict structure with quadratic consensus power. This is a worst-case

statement: it does not force disjoint transfers through consensus. It does show that a constant-state design must control how masking sets are selected and how conflicting transactions are scheduled. Fair ordering, unpredictable set selection, and contention control therefore complement the cryptographic privacy mechanism rather than merely optimize it.

The full version [10] studies the complementary linear untraceable asset transfer (LUAT) object, which retains masking sets and grows the ledger. LUAT has consensus number 2 for every masking-set size and is starvation-free. The comparison isolates the design cost: LUAT pays for untraceability in storage, whereas CUAT pays in synchronization and fairness.

Section 2 discusses related work, and Section 3 gives the model. Section 4 defines CUAT and proves the consensus bounds. Section 5 studies parallelism and starvation; Section 6 concludes.

2 Related Work

Our analysis of CUAT's synchronization power connects three distinct research areas: consensus numbers of cryptocurrency objects and privacy-preserving primitives, quorum systems and combinatorial design theory, and multi-word compare-and-swap primitives.

Guerraoui *et al.* [21] challenge the assumption that consensus is necessary to transfer assets. By modeling asset transfers as concurrent objects, they show the consensus number of standard asset transfer is 1, which means that transactions can be processed asynchronously without total ordering. The result concerns the asset-transfer object alone, and not a cryptocurrency as a whole: issuance mechanisms such as mining, as well as smart-contract state, lie outside the object and need not have consensus number 1. Building on this insight, modern cryptocurrencies like Sui [8], Libra/Diem, and Zef [5] adopt their k-shared asset transfer model, where only the k users sharing an account need consensus, thereby reducing synchronization overhead. Beyond basic transfers, Ethereum's ERC20 token standard introduces additional complexity. As Alpos *et al.* [2] observe, ERC20 tokens allow account owners to approve multiple spenders, which in turn creates a dynamic consensus number where synchronization requirements depend on the number of authorized spenders. Importantly, this demonstrates that consensus requirements in smart contracts are state-dependent rather than fixed, thereby enabling more efficient synchronization mechanisms.

Privacy-preserving systems, however, face distinct synchronization challenges. Frey *et al.* [18] analyze the stronger allow-list and deny-list objects, showing that while allow-lists need no synchronization, deny-lists require consensus among verifiers performing set-non-membership proofs. More broadly, their framework applies to decentralized identity management and e-voting.

CUAT's pairwise-intersecting masking sets bear structural similarities to quorum systems [33, 20]. Both use intersections among sets of elements to provide synchronization in distributed systems. The key distinction is that quorum systems are designed to tolerate process failures and must satisfy constraints derived from failure assumptions, whereas masking sets in CUAT need not account for such failures. Despite this difference, both serve as synchronization abstractions. We show that CUAT requires additional conditions beyond pairwise intersection. First, to implement consensus, the intersections between masking sets must be specifically structured to identify the winner of a transfer operation. Second, to maintain sender untraceability across multiple transactions, we require uniformity, where each account must appear in the same number of masking sets.

CUAT's transfer operation exhibits structural similarities to multi-word compare-and-swap (MCAS) primitives. Harris *et al.* [22] introduce practical MCAS implementations using single-word CAS operations. Like MCAS, CUAT transfers atomically update multiple

memory locations (account representations in a masking set) and succeed only if all locations match expected values. However, CUAT differs fundamentally due to its privacy requirements and the conditions to maintain sender untraceability.

Christ and Bonneau [12] introduce revocable proof systems (RPS) as a unifying abstraction for authenticated data structures, including stateless blockchains and set commitments. They prove a fundamental trade-off: any scheme with sublinear global state must incur near-linear proof-update costs when statements are revoked (e.g., when coins are spent). This information-theoretic lower bound focuses on storage and witness maintenance rather than concurrency.

3 Preliminaries

This section establishes the theoretical foundations for our analysis. We present the system model, consensus numbers (which measure synchronization power of shared objects), and the consensus hierarchy theorem.

Notation. We use $\leftarrow$ for assignment in algorithms, $=$ for equality testing, $:=$ for definitions, and standard set notation ($\in$ for membership, $\subseteq$ for subset, $\cap$ for intersection, $\cup$ for union).

System Model. We consider N asynchronous sequential processes $p_1, \ldots, p_N$ in a set $\mathcal{P}$ that may crash (crash-failure model). Throughout, Π denotes a protocol; the process set is $\mathcal{P}$. Processes interact through shared objects supporting operations that produce responses. Objects satisfy *linearizability* [25], where operation histories appear as if executed in a single, consistent order respecting real-time constraints. A *register* is a shared-memory abstraction supporting *read* and *write* operations. We assume *atomic registers* with linearizable semantics, where each operation appears to take effect instantaneously between invocation and response.

Wait-Freedom. An implementation of a shared object is *wait-free* if every operation invoked by a correct process completes in a finite number of steps, regardless of the behavior of other processes.

Adversarial Model. External adversary $\mathcal{E}$ can *eavesdrop* on network ciphertexts, record shared object states, and analyze process interactions to infer relationships. Though computationally unbounded, the adversary faces idealized cryptographic primitives secure against all attacks.

Consensus Numbers and the Consensus Hierarchy. The consensus number $\mathsf{cons}(O)$ of an object O is the largest number n such that it is possible to wait-free implement a consensus object from atomic registers and objects of type O, in a system of n processes [23]. The consensus hierarchy shows that objects with higher consensus numbers are strictly more powerful and cannot be implemented from objects with lower consensus numbers. For an individual protocol Π we write $\mathsf{cn}(\Pi)$ for the largest n such that Π solves n-consensus.

Valency. Following the standard terminology [23], a configuration C of a binary consensus protocol is *v-valent* (with $v \in \{0, 1\}$) if every execution extending C decides v, *univalent* if it is v-valent for some v, and *bivalent* if both decision values are still reachable from C. We write $C \cdot p$ for the configuration reached when process p takes one step from C, and chain steps as $C \cdot p \cdot q$ for the configuration reached when p then q each take one step. A bivalent configuration is *critical* if every immediate successor $C \cdot p$ is univalent.

Ownership. Both objects we study attach an owner to each account through a map $\mu : \mathcal{A} \to 2^{\mathcal{P}} \setminus \{\emptyset\}$, and only a process in $\mu(a)$ may spend a or read its balance. An account a is *k-shared* if $|\mu(a)| = k$, and *unshared* if $|\mu(a)| = 1$; an object is unshared if all of its accounts are. Following Guerraoui *et al.* [21], sharing is the mechanism by which several processes contend for one account. The CUAT_λ objects considered below are unshared (Definition 4). Since a process funds its transfer from an account of its own, at most λ processes can then carry the same masking set.

4 Constant Untraceable Asset Transfer

Quisquis and similar systems [16, 3, 29] address the monotonic growth problem through account replacement: each transfer invalidates all accounts in a masking set and creates new ones, while cryptographic proofs hide the sender. For contrast, the full version [10] describes the linear untraceable asset transfer (LUAT) object, which uses the classical growing allow-set and deny-set approach. We model these constant-state systems through the *constant untraceable asset transfer* (CUAT) object, a shared-memory abstraction building on the asset transfer formalism of Guerraoui *et al.* [21]. CUAT maintains sender untraceability through a masking-set mechanism where each transfer invalidates old accounts and creates new accounts for all members within a fixed-size set. Before presenting the formal model, we first define the untraceability notions that CUAT must satisfy.

4.1 Untraceability

Untraceability requires the adversary's posterior on the sender to remain uniform over the masking set. We write $P(a_i \mid \cdot)$ for the posterior probability that a_i funded the transfer under consideration; the underlying probability model is specified in Section 4.3.4. We consider both the standard single-transaction guarantee [31, 6] and protection against history-aware deanonymization [28, 32, 15].

▶ **Definition 1** (Weak untraceability)**.** CUAT_λ *satisfies* weak untraceability *if, for every transfer τ with masking set S and every $a_i \in S$, $P(a_i \mid \tau) = 1/|S| = 1/\lambda$.*

Weak untraceability restricts the adversary's view to the transfer under consideration. Strong untraceability instead conditions on everything the adversary has recorded, including the masking sets of failed transfers.

▶ **Definition 2** (Strong untraceability)**.** CUAT_λ *satisfies* strong untraceability *if, for every transfer τ with masking set S in any history H and every $a_i \in S$, $P(a_i \mid H) = 1/|S| = 1/\lambda$.*

4.2 Formal Model

We now formalize the λ-CUAT object. Each transfer atomically invalidates an old masking set and creates a new one of size λ, with a zero-knowledge proof witnessing ownership of the invalidated accounts.

▶ **Definition 3** (Masking sets and λ-masking)**.** *A* masking set *is any $S \subseteq \mathcal{A}$. The system is* λ-masking *if every masking set used by the protocol has size exactly λ.*

With masking sets defined, we can specify the CUAT object. The definition below formalizes its state, operations, and transition rules.

▶ **Definition 4** (CUAT_λ object)**.** CUAT_λ *is a shared-memory asset transfer object specified by the tuple* $\mathsf{CUAT}_\lambda = (Q, q_0, O, R, \Delta, U)$ *where:*

- ***States:*** Q *consists of pairs* $q = (\mathcal{A}, \beta)$ *where* $\mathcal{A}$ *is a finite set of currently valid accounts and* $\beta : \mathcal{A} \to \mathbb{N}$ *maps each account to its balance. Each account* $a \in \mathcal{A}$ *carries two fixed fields assigned at creation: an owner set* $\mu(a) = \{p\}$ *for a single process* p*, and an index* $\mathrm{idx}(a) \in \mathbb{N}$ *also denoted by a subscript. The initial state is* $q_0 = (\mathcal{A}_0, \beta_0)$ *for some finite* $\mathcal{A}_0$*.*
- ***Operations:*** *The set* O *consists of:* $\mathrm{read}(a)$ *for any account* a*, and* $\mathrm{transfer}(S_{\beta'}, \pi)$ *where* $S_{\beta'} = \{a'_1, \ldots, a'_\lambda\}$ *is a set of* λ *new accounts and* π *is a zero-knowledge proof. The transfer carries an implicit bijection* $\phi : S_\beta \to S_{\beta'}$ *between the old and new masking set that preserves both account fields:* $\mu(\phi(a)) = \mu(a)$ *and* $\mathrm{idx}(\phi(a)) = \mathrm{idx}(a)$ *for every* $a \in S_\beta$*.*
- ***Responses:*** R *consists of:*
  - *For* $\mathrm{read}(a)$*: either* $\beta(a)$ *if the invoking process owns* a*, or* $\perp$ *otherwise.*
  - *For* $\mathrm{transfer}(S_{\beta'}, \pi)$*: a pair* $(\mathcal{A}, b)$ *where* $\mathcal{A}$ *is the current account set and* $b \in \{\texttt{TRUE}, \texttt{FALSE}\}$ *indicates success or failure.*
- ***Transitions:*** $\Delta \subseteq Q \times O \times R \times Q$ *specifies allowed state transitions. For any* $q = (\mathcal{A}, \beta)$ *and* $q' = (\mathcal{A}', \beta')$*:*
  - ***Account read:*** *If* $o = \mathrm{read}(a)$ *invoked by process* p*, then* $q' = q$ *and*

$$r = \begin{cases} \beta(a) & \text{if } a \in \mathcal{A} \text{ and } p \in \mu(a), \\ \perp & \text{otherwise.} \end{cases}$$

  - ***Transfer success:*** *If* $o = \mathrm{transfer}(S_{\beta'}, \pi)$ *with* $S_\beta \subseteq \mathcal{A}$*,* $S_{\beta'} \cap \mathcal{A} = \emptyset$*,* $|S_{\beta'}| = \lambda$*, and* π *is a proof demonstrating* $\exists a_s, a_r \in S_\beta, v \in \mathbb{N}$ *such that the invoking process lies in* $\mu(a_s)$ *and* $\beta(a_s) \geq v$*, then the new account set is*

$$\mathcal{A}' = (\mathcal{A} \setminus S_\beta) \cup S_{\beta'},$$

and the new balance map $\beta' : \mathcal{A}' \to \mathbb{N}$ *is*

$$\beta'(a) = \begin{cases} \beta(\phi^{-1}(a)) - v & \text{if } a = \phi(a_s), \\ \beta(\phi^{-1}(a)) + v & \text{if } a = \phi(a_r), \\ \beta(\phi^{-1}(a)) & \text{if } a \in S_{\beta'} \setminus \{\phi(a_s), \phi(a_r)\}, \\ \beta(a) & \text{if } a \in \mathcal{A} \setminus S_\beta. \end{cases}$$

  Return $r = (\mathcal{A}', \texttt{TRUE})$*.*
  - ***Transfer failure:*** *In all other cases of* $\mathrm{transfer}(\cdot)$*, set* $q' = q$ *and* $r = (\mathcal{A}, \texttt{FALSE})$*.*
- ***Untraceability:*** U *specifies the untraceability notion required for the object. We consider two notions: weak untraceability (Definition 1), giving the object* CUAT^-_λ*, and strong untraceability (Definition 2), giving* CUAT^+_λ*. We write* CUAT_λ *for either variant when the result or argument applies to both. The notion restricts the valid executions of the object, as some use patterns lead to untraceability loss, especially under strong untraceability.*

The λ-CUAT models privacy-preserving cryptocurrencies through account invalidation and creation. A transfer takes a set $S_{\beta'} = \{a'_1, \ldots, a'_\lambda\}$ of λ new accounts and a zero-knowledge proof π attesting ownership and valid balance updates. On success it atomically invalidates every account of S_β and creates its image under ϕ with updated balances, so old and new accounts occupy the same logical positions under different cryptographic representations. The

proof hides which accounts of S_β are the sender and the recipient, and balance confidentiality restricts balances to account owners. We write τ for the transfer operation and $\tau(S)$ for an invocation that consumes the masking set S; the operation is invoked with the replacement accounts $S_{\beta'}$, while S_β is what it consumes and hence what governs its conflicts.

A transfer therefore blocks every concurrent transfer that uses any account of S_β, while transfers on disjoint masking sets proceed in parallel. This atomicity is what gives λ-CUAT non-trivial synchronization power, which the next section quantifies.

4.3 Consensus Number

This section characterizes the wait-free consensus number of λ-CUAT. Protocols may use any finite number of λ-CUAT instances together with atomic registers. By Definition 4, the bijection ϕ of every successful transfer preserves each account's index $\mathrm{idx}(a)$, so two accounts with equal index represent the same logical position across rerandomizations; assuming distinct processes never produce identical cryptographic representations,[2] two masking sets conflict precisely when they share an idx value, and the synchronization structure of a protocol is captured by a graph on its masking sets (Definition 5 below). Two modeling choices distinguish us from earlier work on asset transfer. Guerraoui *et al.* [21] encode the winning process in account balances, a path that CUAT's consensus cannot take since only an account's owner may query its balance; synchronization therefore runs through the masking sets themselves. Untraceability is a runtime constraint [13]: we admit only executions consistent with the notion in force, and the consensus number is taken over that restricted family.

We distinguish protocols by the number of transfers each process may invoke. We first consider the one-round regime, which most directly models the use of CUAT in payment systems, and then turn to the more general multi-round regime.

Section 4.3.1 treats *one-round* protocols (one τ-invocation per process). This models how CUAT is used in payment systems [16, 3, 29]: a user submits a single transaction whose masking set is fixed before submission, and validators either commit it or reject it. The one-round upper bound therefore captures the synchronization cost faced inside one such transaction: validators process non-conflicting transfers independently and only need to coordinate when two transfers contend over a shared masking set, and the upper bound sizes the worst-case coordination slice.

Section 4.3.2 lifts the analysis to *multi-round* protocols, where CUAT is used as a synchronization primitive inside a layered agreement protocol whose outer rounds adapt their masking sets to earlier outcomes. This captures CUAT's theoretical synchronization power but is not the typical operational mode of payment systems. To start our analysis, we first formalize the conflict graph and its clique number.

▶ **Definition 5** (Conflict graph and clique number). *Two masking sets S and S' on a common* CUAT_λ *object* conflict *when there exist $a \in S$ and $a' \in S'$ with $\mathrm{idx}(a) = \mathrm{idx}(a')$. Let $\mathcal{S}(\Pi)$ denote the family of τ-invocations occurring in some execution of Π admissible under the untraceability notion in force, each carrying its masking set. The* conflict graph *G of Π has a vertex v_S for each invocation in $\mathcal{S}(\Pi)$, labelled by the masking set S it carries, and an edge $(v_S, v_{S'}) \in E$ whenever S and S' conflict; two invocations carrying the same masking set are distinct vertices, and they are adjacent. A* clique *in G is a set of pairwise-conflicting invocations, and the* clique number *$\omega(G)$ is the size of the largest clique.*

[2] Negligible in the security parameter for any concrete instantiation.

The family $\mathcal{S}(\Pi)$ captures every τ-invocation Π may make, including across executions where processes adapt their choices to the schedule. Any single execution realizes a subgraph of G, with clique number at most $\omega(G)$. The upper-bound arguments below inspect the invocation each process is about to make at a critical configuration; these invocations lie in $\mathcal{S}(\Pi)$ by definition, so the bounds hold whether Π chooses masking sets statically or dynamically. The matching lower bounds are realized by protocols whose masking-set assignment is fixed in advance. Alongside the conflict graph, a second invariant tracks how many masking sets contain a given account.

▶ **Definition 6** (Incidence number). *The* incidence *of an account $a \in \mathcal{A}$ is the number of vertices $v_S \in V(G)$ such that $a \in S$. The* incidence number *of* Π *is* $r(\Pi) = \max_{a \in \mathcal{A}} \text{incidence}(a)$.

Cliques in G implement consensus. When d invocations pairwise conflict, at most one of them can return `TRUE`: any pair shares an index, and the second transfer to commit finds that index already invalidated and fails. Losing transfers inspect the published CUAT state, locate the successful transfer by the new accounts it created, and adopt the value its issuer wrote to a single-writer register. Pairing each of d processes with one clique member yields a d-process consensus protocol that attempts the transfer, decides on success, and adopts on failure; hence $\omega(G)$ lower-bounds the consensus power of λ-CUAT.

The matching upper bounds, both at $\omega(G)$ and beyond, rest on a binary-input specialization of the standard critical-state construction [17, 23].

▶ **Lemma 7** (Different valencies). *Let Π be a deterministic wait-free consensus protocol for $n \geq 2$ processes in the asynchronous shared-memory model. For any process p and any $v \in \{0, 1\}$, the input assignment I with $I(p) = v$ and $I(s) = 1 - v$ for $s \neq p$ admits a reachable critical configuration C such that $C \cdot p$ is v-valent and $C \cdot s$ is $(1 - v)$-valent for every $s \neq p$.*

Proof. By symmetry of 0 and 1, take $v = 0$; process p then has input 0 and every other process has input 1. Let σ be the schedule in which p takes no steps and every other process runs to completion. By wait-freedom each $s \neq p$ decides along σ, and every such decision is 1: as p takes no step, σ is indistinguishable to those processes from σ run on the assignment that also gives p input 1, where validity forces 1. Hence σ ends in a 1-valent configuration. The initial configuration is bivalent (σ decides 1 while running p alone decides 0), so σ has a last bivalent prefix, and the standard critical-state construction extends that prefix p-silently to a reachable critical configuration C.

Fix $s \neq p$ and suppose, toward a contradiction, that $C \cdot s$ is 0-valent. Every extension of $C \cdot s$ then decides 0. Consider the extension that runs every process in $\mathcal{P} \setminus \{p\}$ to completion before p takes a step: wait-freedom guarantees termination, and the same indistinguishability argument forces the decision 1, since every process decides before p takes a step. This contradicts 0-valence, so $C \cdot s$ is 1-valent for every $s \neq p$. By bivalence of C, the remaining successor $C \cdot p$ is 0-valent. ◀

In each subsequent application, we extract a reachable critical configuration whose successors split processes by valency, then exploit the commutativity of λ-CUAT transfers on disjoint account indices: two masking sets on opposite valency sides must conflict, since otherwise their transfers commute and produce indistinguishable configurations forced to decide differently. The next lemma packages this technique.

▶ **Lemma 8** (Bipartite intersection at a critical configuration). *Let* Π *be a wait-free binary consensus protocol for* $n \geq 3$ *processes on* CUAT_λ *objects and atomic registers, and let* C *be a reachable critical configuration of* Π*. There exist a* CUAT_λ *object* O *and a non-trivial partition* $\mathcal{P}_0 \sqcup \mathcal{P}_1 = \mathcal{P}$ *such that every pending step at* C *is a* τ*-invocation on* O*, and, writing* S_p *for the masking set carried by* p*'s pending invocation:*

(i) $C \cdot p$ *is* 0*-valent for every* $p \in \mathcal{P}_0$ *and* $C \cdot q$ *is* 1*-valent for every* $q \in \mathcal{P}_1$*;*

(ii) $S_p \cap S_q \neq \emptyset$ *on* O *for every* $(p, q) \in \mathcal{P}_0 \times \mathcal{P}_1$*.*

Proof. At a critical configuration neither pending step is a register operation [23]: a read leaves the shared state unchanged, writes to distinct registers commute, and of two writes to one register the second overwrites the first, leaving a configuration indistinguishable to every process except the overwritten writer. Each case makes two oppositely valent successors indistinguishable to some process that decides in both. Every pending step at C is therefore a τ-invocation. Define $\mathcal{P}_0 = \{p \in \mathcal{P} : C \cdot p \text{ is 0-valent}\}$ and $\mathcal{P}_1 = \mathcal{P} \setminus \mathcal{P}_0$; both are non-empty by bivalency of C, and property (i) follows.

Fix $p \in \mathcal{P}_0$ and $q \in \mathcal{P}_1$, and suppose toward a contradiction that $\tau(S_p)$ and $\tau(S_q)$ are on distinct λ-CUAT objects, or on a common object with $S_p \cap S_q = \emptyset$. Each transfer's outcome is then unaffected by the other: the two masking sets are disjoint, and by account-representation uniqueness neither transfer's new accounts can appear in the other's masking set. The object states after the pair of steps therefore agree in either order, and neither step writes a register, so the shared state at $C \cdot p \cdot q$ equals that at $C \cdot q \cdot p$. The two configurations are not indistinguishable to p and q themselves, whose transfer responses carry the current account set and hence reveal the order; but they are indistinguishable to every other process. Since $n \geq 3$, choose a spectator $s \notin \{p, q\}$ and run it alone from each configuration. By wait-freedom s decides in both runs, and indistinguishability makes the two decisions equal; yet every extension of $C \cdot p \cdot q$ decides 0, as $C \cdot p$ is 0-valent, and every extension of $C \cdot q \cdot p$ decides 1, as $C \cdot q$ is 1-valent. This contradiction shows that p and q invoke τ on a common λ-CUAT object with $S_p \cap S_q \neq \emptyset$ on that object. Iterating over $(p, q) \in \mathcal{P}_0 \times \mathcal{P}_1$, all cross-valency pairs share a single object O; transitivity through such pairs forces every process in $\mathcal{P}$ to invoke τ on O, and property (ii) holds with masking sets on O. ◀

In particular, applying the lemma to the critical configuration produced by Lemma 7 with (p, v) gives the *star* specialization, with $\mathcal{P}_v = \{p\}$ and $\mathcal{P}_{1-v} = \mathcal{P} \setminus \{p\}$: p's pending masking set conflicts on a common object with every other process's pending masking set. Iterating over p is what the one-round upper bound relies on.

4.3.1 One-Round Protocols

Throughout this subsection, Π denotes a wait-free one-round consensus protocol on λ-CUAT objects and atomic registers: each process makes a single τ-invocation and identifies a contested winner from the published CUAT state. Before its τ-step a process has access only to the CUAT's initial state and its own identity, so its masking set is determined by the protocol from these two pieces of data alone. A process may also read registers, but atomic registers have consensus number 1 [23] and add no synchronization power on top of λ-CUAT. The analysis pairs a lower bound from constructive cliques in G with an upper bound from iterating Lemma 8 over each process.

Any clique of G admits a wait-free consensus protocol of matching size, by the standard attempt-and-adopt construction.

▶ **Lemma 9** (One-round lower bound from cliques). *Let G be a conflict graph admissible under the untraceability notion in force, and let $K = \{S_1, \ldots, S_n\} \subseteq V(G)$ be a clique of size n whose members can be assigned to distinct processes so that S_i contains an account owned by p_i. Then there is a wait-free one-round n-consensus protocol Π over a CUAT_λ object in which process p_i uses masking set S_i. Consequently $\mathsf{cons}(\mathsf{CUAT}_\lambda) \geq \omega(G)$ whenever a largest clique admits such an assignment.*

Proof.

Construction. Assign S_i as p_i's masking set. Each p_i writes its input to a single-writer register r_i (proposal) and invokes $\tau(S_i)$ on the λ-CUAT object. On success (τ returns `TRUE`), p_i decides its own input. On failure, p_i reads the published CUAT state to identify which transfer has been committed, reads the corresponding r_j, and decides that value. Algorithm 1 gives the full pseudocode.

Wait-freedom. The protocol contains no loops: τ is invoked once, and identification of the winner is a single inspection of the published state. Each process completes in a bounded number of steps regardless of the schedule.

Validity. The decided value is either p_i's own input (on success) or r_{j^*} for some j^*, which is p_{j^*}'s input.

Agreement. Linearize the n τ-invocations on K and let the one issued by p_{j^*} be the first to return `TRUE`. For every other $S_i \in K$, $S_i \cap S_{j^*} \neq \emptyset$ by the clique property of K; the account shared between S_i and S_{j^*} is invalidated when p_{j^*} commits, so $\tau(S_i)$ fails. Process p_i then identifies p_{j^*}'s transfer in the published state, by the new accounts it created, and decides r_{j^*}. Process p_{j^*} also decides r_{j^*} on success. Every process therefore decides the same value r_{j^*}. ◀

Algorithm 1 Wait-free implementation of d-consensus using a λ-CUAT object.

Shared state: atomic registers $R[1..d]$, $A[1..d]$, a λ-CUAT object CUAT on a finite account set $\mathcal{A}_0$ with $\beta_0 \equiv 0$, a masking-set assignment $i \mapsto S_i \subseteq \mathcal{A}_0$ such that $|S_i| = \lambda$, $S_i \cap S_j \neq \emptyset$ for all $i \neq j$, and p_i owns some $a_i \in S_i$

function propose(v) *at process* p_i**:**
- $R[i].write(v)$ — `// write proposal`
- $S_{\text{old}} \leftarrow S_i$ — `// old masking set`
- $(S_{\text{new}}, \pi) \leftarrow Compute_{p_i}(S_{\text{old}}, a_i, a_i', 0)$ — `// new accounts and proof`
- $A[i].write(S_{\text{new}})$ — `// write new account set`
- $(\mathcal{A}, b) \leftarrow \mathsf{CUAT}.transfer(S_{\text{new}}, \pi)$
- **if** $b =$ *TRUE* **then**
  - **return** v — `// won: decide own value`
- **else**
  - $S_{\text{winner}} \leftarrow \mathcal{A} \setminus \mathcal{A}_0$ — `// new accounts in` $\mathcal{A}$
  - **for** $j \in \{1, \ldots, d\}$ **do**
    - **if** $A[j].read() = S_{\text{winner}}$ **then**
      - **return** $R[j].read()$ — `//` p_j `won: adopt its value`

Conversely, Lemma 8 turns wait-freedom into a clique witness in G.

▶ **Lemma 10** (One-round upper bound from pairwise intersection). *Let* Π *be a wait-free one-round consensus protocol on* CUAT_λ *objects and atomic registers, with* G *its conflict graph. If* $\mathsf{cn}(\Pi) \geq 3$*, then* $\mathsf{cn}(\Pi) \leq \omega(G)$.

Proof. Let $n = \mathsf{cn}(\Pi) \geq 3$ and write S_i for the masking set carried by p_i's single τ-invocation, which the protocol fixes from p_i's identity and the initial state alone.

Fix $p \in \mathcal{P}$ and apply Lemma 7 to the input assignment giving p input 0 and every other process input 1: it yields a reachable critical configuration C at which $C \cdot p$ is 0-valent and $C \cdot s$ is 1-valent for every $s \neq p$. The valency classes at C are therefore $\mathcal{P}_0 = \{p\}$ and $\mathcal{P}_1 = \mathcal{P} \setminus \{p\}$, the star specialization of Lemma 8: every process invokes τ on one common object O, and S_p conflicts with S_s on O for every $s \neq p$.

Iterating over $p \in \mathcal{P}$ makes every two of the n invocations conflict. Distinct processes make distinct invocations, so these are n distinct vertices of G, pairwise adjacent by Definition 5, hence an n-clique. Therefore $\mathsf{cn}(\Pi) = n \leq \omega(G)$. ◀

Combining the two bounds gives equality.

▶ **Theorem 11** (One-round consensus number). *If* Π *is a one-round protocol on* CUAT_λ *with* $\mathsf{cn}(\Pi) \geq 3$*, then* $\mathsf{cn}(\Pi) = \omega(G)$*. Consequently, the largest* n *achievable by a one-round* n*-consensus protocol on* CUAT_λ *equals the supremum of* $\omega(G)$ *over conflict graphs admissible under the untraceability notion in force.*

Proof. Combine Lemma 9 and Lemma 10. ◀

Theorem 11 matches the one-round upper bound to $\omega(G)$, and the bound is the limit of the one-round regime: a committed τ permanently invalidates its masking set, and the surviving masking sets that conflicted with it can no longer fire, so the conflict graph is exhausted after one round of contention. Two follow-up questions remain. Section 4.3.2 relaxes the one-round restriction and asks how far multi-round protocols can push the consensus number past $\omega(G)$. Sections 4.3.3 and 4.3.4 take the orthogonal question of how large $\omega(G)$ can be made under each untraceability notion: weak untraceability leaves $\omega(G)$ unbounded, while strong untraceability caps it at $\lambda^2 - \lambda + 1$.

4.3.2 Multi-Round Protocols

A multi-round protocol lets each process invoke τ more than once and adapt later invocations to earlier outcomes. The structural gain over $\omega(G)$ is not larger cliques but a weaker requirement on the conflict graph: pairwise intersection at the critical configuration can be relaxed once the pending processes split by valency, so the next τ-round need only enforce conflict across the partition. Lemma 8 already extracts such a partition $\mathcal{P} = \mathcal{P}_0 \sqcup \mathcal{P}_1$ with a complete bipartite intersection between the two sides. At incidence 2 a λ-block biclique supports 2λ vertices, exceeding the one-round upper bound.

▶ **Theorem 12** (Upper bound). *A wait-free binary consensus protocol* Π *over* CUAT_λ *objects and atomic registers can solve consensus among at most* $\max\big(2,\, 2\lambda(r(\Pi) - 1)\big)$ *processes.*

Proof. Suppose Π solves n-consensus, so $|\mathcal{P}| = n$; for $n \leq 2$ the bound is immediate, so assume $n \geq 3$. Take C, $\mathcal{P}_0$, $\mathcal{P}_1$ as in Lemma 8; the partition covers $\mathcal{P}$, and cross-valency pending masking sets conflict on a common CUAT_λ object.

Distinct processes make distinct invocations, so $\{v_{S_p} : p \in \mathcal{P}_0\}$ and $\{v_{S_q} : q \in \mathcal{P}_1\}$ are disjoint sets of vertices, of sizes $|\mathcal{P}_0|$ and $|\mathcal{P}_1|$, whether or not two processes happen to carry the same masking set. Property (ii) of Lemma 8 makes every cross pair adjacent, so the two sets span a complete bipartite subgraph $K_{|\mathcal{P}_0|,|\mathcal{P}_1|} \subseteq G$.

Fix $p \in \mathcal{P}_0$. Each account $a \in S_p$ has incidence at most $r(\Pi)$ in G, hence lies in at most $r(\Pi) - 1$ vertices distinct from v_{S_p}. Summing over the λ accounts of S_p, at most $\lambda(r(\Pi) - 1)$ distinct vertices in $V(G)$ are adjacent to v_{S_p} via a shared account. Since every v_{S_q} with $q \in \mathcal{P}_1$ is adjacent to v_{S_p}, we get $|\mathcal{P}_1| \leq \lambda(r(\Pi) - 1)$. By symmetry $|\mathcal{P}_0| \leq \lambda(r(\Pi) - 1)$, so $n = |\mathcal{P}_0| + |\mathcal{P}_1| \leq 2\lambda(r(\Pi) - 1)$. ◀

In the following two sections, we discuss how the incidence number $r(\Pi)$ relates to privacy and, through it, to the consensus number.

4.3.3 Weak Untraceability

Under weak untraceability the incidence number $r(\Pi)$ is unconstrained: an account may appear in every masking set without violating per-transaction anonymity. The hub construction places a shared hub account a_1 (owned by p_1) in all d masking sets, making the conflict graph a d-clique, and satisfies weak untraceability for any d. By Theorem 11 this yields $\mathsf{cn}(\Pi) = d$, so the consensus number is unbounded.

▶ **Theorem 13** (Consensus number of CUAT^-_λ)**.** *For every $d \geq 2$, CUAT^-_λ supports a wait-free one-round d-consensus protocol. Hence $\mathsf{cons}(\mathsf{CUAT}^-_\lambda) = \infty$.*

Proof. We instantiate Lemma 9 on the hub family and verify that the family satisfies weak untraceability.

Construction. Put $M = \max(d, \lambda)$ and take $\mathcal{A}_0 = \{a_1, \ldots, a_{M+1}\}$ with $\mu(a_{i+1}) = \{p_i\}$ for $i \leq d$. Let S_i consist of the hub a_1 (owned by p_1) together with the $\lambda - 1$ accounts starting at a_{i+1} and taken cyclically through $a_2, \ldots, a_{M+1}$, so that S_i contains p_i's own account a_{i+1}. Since all d sets share a_1, they pairwise intersect, so $K = \{S_1, \ldots, S_d\}$ is a d-clique in the conflict graph G. Lemma 9 then implements wait-free d-consensus on this clique.

Weak untraceability. The cryptographic proof π of each τ-invocation is zero-knowledge: an observer of τ learns the masking set S_{old} and the published state, but obtains no advantage over the uniform distribution on S_{old} for the sender's identity. Hence $P(a_i \mid \tau) = 1/|S_{\text{old}}| = 1/\lambda$ for every $a_i \in S_{\text{old}}$, regardless of d or of the hub-account structure. Therefore Π achieves wait-free d-consensus and preserves weak untraceability for every $d \geq 2$. ◀

The hub construction leaves a_1 in every masking set, so a history-aware adversary identifies it as a near-certain decoy; this statistical leakage motivates strong untraceability.

4.3.4 Strong Untraceability

Strong untraceability protects against an adversary that observes the complete transaction history. We first show that it is equivalent to a local uniformity condition on account incidence. We then use this condition to bound both $\omega(G)$ and $r(\Pi)$. Each account $a \in \mathcal{A}$ has an unknown sender probability q_a and an unknown receiver probability r_a; decoys are sampled uniformly. The prior is symmetric across accounts. We use the independent-inclusion model: account appearances are independent across accounts and transfers. Consequently, the likelihood of a history depends on each account only through its incidence $f_a(H)$. An adversary with additional information, such as account co-occurrences or side channels, is only stronger; the consensus upper bounds below use only the necessity direction of Theorem 14. Let $\mathcal{A}(H) = \{a : f_a(H) \geq 1\}$ be the accounts that occur in H.

▶ **Theorem 14** (Uniformity). *Denote by $f_i(H)$ the number of masking sets in which account a_i appears in the history of transactions H. Under the adversary model above,* CUAT_λ *satisfies* strong untraceability *on H if and only if $f_i(H) = f_j(H)$ for every two accounts a_i, a_j lying in a common masking set of H.*

Proof. Write $A = \mathcal{A}(H)$. An account occurs in a masking set as the sender, the receiver, or one of the $\lambda - 2$ uniformly sampled decoys. Under the independence assumption, the likelihood factorizes over accounts. Hence there is a function g, independent of the account identity, such that

$$\Pr\left[H \mid (q_a, r_a)_{a\in A}\right] \;\propto\; \prod_{a\in A} g\big(q_a, r_a, f_a(H)\big).$$

Thus $f_a(H)$ is sufficient for (q_a, r_a). By symmetry of the prior, accounts with equal incidence are posterior-exchangeable. Accounts with different incidence are not: uniform decoy sampling contributes equally to all accounts, and $\mathbb{E}[q_a \mid H]$ is strictly increasing in $f_a(H)$. For sufficiency, suppose $f_a(H) = f_b(H)$ whenever a and b lie in a common masking set of H. Fix $\tau \in H$ and $a_i, a_j \in S_\tau$, and note $S_\tau \subseteq A$. Let π swap a_i and a_j and fix every other account. Since $f_{a_i}(H) = f_{a_j}(H)$ by hypothesis and no other count moves, π fixes S_τ and preserves the incidence vector, while mapping the event "a_i sent τ" to "a_j sent τ". Posterior exchangeability gives the two events equal probability. Exactly one member of S_τ sent τ, so these $|S_\tau| = \lambda$ probabilities sum to 1 and each equals $1/\lambda$. Strong untraceability holds on H.

For necessity, suppose strong untraceability holds on H, and fix $\tau \in H$ with $a_i, a_j \in S_\tau$. Both have posterior probability $1/\lambda$ of having sent τ. Were $f_i(H) \neq f_j(H)$, strict monotonicity of $\mathbb{E}[q_a \mid H]$ in $f_a(H)$ would give the accounts different posterior sender probabilities for τ. Consequently, some $a^\star \in S_\tau$ would satisfy $P(a^\star \mid H) > 1/\lambda$. Hence $f_i(H) = f_j(H)$: incidence is constant on every masking set of H. ◀

The theorem gives a local condition, comparing accounts that occur in a common masking set. Connected masking-set families satisfy a global form of the same condition.

▶ **Corollary 15** (Uniform incidence). *Let H be a history whose masking sets are* connected under intersection*, meaning that their intersection graph is connected; this holds in particular when they pairwise intersect. Then* CUAT_λ *satisfies strong untraceability on H if and only if $f_i(H) = f_j(H)$ for all $a_i, a_j \in \mathcal{A}(H)$.*

Proof. If incidence is constant on $\mathcal{A}(H)$ it is constant on every masking set of H, and Theorem 14 gives strong untraceability. Conversely, let strong untraceability hold on H and let $a, b \in \mathcal{A}(H)$. The theorem makes incidence constant on each masking set of H. Each of a, b lies in some masking set, and by hypothesis there are masking sets $S^{(1)}, \ldots, S^{(m)}$ of H with $a \in S^{(1)}$, $b \in S^{(m)}$ and $S^{(l)} \cap S^{(l+1)} \neq \emptyset$ for each $l < m$. Consecutive sets share an account at which their two constants agree, so one value runs along the whole chain and $f_a(H) = f_b(H)$. ◀

Connectedness is necessary. For example, two disjoint masking-set components may have different incidences while remaining uniform within each component. The consensus constructions below use connected families, so the corollary applies. We quantify only over $\mathcal{A}(H)$ because newly created accounts outside the history have incidence zero.

We next bound the clique number of a one-round protocol. Its masking sets are λ-uniform and pairwise intersect, and strong untraceability makes their account incidence uniform.

▶ **Theorem 16** (Clique bound). *If the masking-set family of a one-round protocol Π has uniform incidence, then $\omega(G) \leq \lambda^2 - \lambda + 1$.*

Proof. Let $F = \{S_1, \ldots, S_n\}$ be a clique of G of size $\omega(G)$; by definition of G, the S_i pairwise intersect, and by hypothesis every account in $\bigcup_i S_i$ lies in a common number m of the invocations of F, each carrying λ accounts. A one-round protocol gives each process a single invocation, determined by its identity, so the n invocations of F belong to n distinct processes; each funds its transfer from an account of its own, and distinct processes own distinct accounts, so $n \leq |\bigcup_i S_i|$, and double counting $|\bigcup_i S_i|\, m = n\lambda$ gives $m \leq \lambda$. Now fix S_1. Every other member of F meets it, and each of the λ accounts of S_1 lies in at most $m-1 \leq \lambda-1$ further members, so $n \leq 1+\lambda(\lambda-1)$. Hence $\omega(G) = n \leq \lambda^2-\lambda+1$; Füredi [19] and Ihringer and Kupavskii [26] characterize the families attaining this bound. ◀

Combining Lemma 10 with Theorem 16 gives the one-round consensus bound under strong untraceability.

▶ **Corollary 17** (One-round bound). *A wait-free one-round consensus protocol on* CUAT^+_λ *can solve consensus among at most* $\lambda^2 - \lambda + 1$ *processes.*

Proof. Let Π be a d-consensus protocol of the class of Section 4.3.1, with $d \geq 3$. Lemma 10 gives $d \leq \omega(G)$: the d masking sets must lie on a common object and form a d-clique in G, hence pairwise intersect. Strong untraceability forces every account in the family to appear in a common number of masking sets, i.e., uniform incidence (Corollary 15). Theorem 16 then caps the clique number, $\omega(G) \leq \lambda^2 - \lambda + 1$. Combining, $d \leq \lambda^2 - \lambda + 1$. ◀

When $\lambda - 1$ is a prime power, the upper bound of Corollary 17 is matched by a projective-plane construction.

▶ **Theorem 18** (Projective plane). *If* $\lambda - 1$ *is a prime power,* CUAT^+_λ *supports a wait-free one-round* $(\lambda^2 - \lambda + 1)$*-consensus protocol, instantiated by attempt-and-adopt on the line set of* $\mathrm{PG}(2, \lambda - 1)$*.*

Proof. Let $q = \lambda - 1$. The projective plane $\mathrm{PG}(2, q)$ has $\lambda^2 - \lambda + 1$ points and the same number of lines; each line contains $\lambda = q + 1$ points, every two lines meet in exactly one point, and each point lies on exactly λ lines [26]. Interpreting points as accounts and lines as masking sets yields a λ-uniform pairwise-intersecting family of size $\lambda^2 - \lambda + 1$ over $\lambda^2 - \lambda + 1$ accounts.

To run the attempt-and-adopt protocol of Lemma 9 (Algorithm 1) on this family, each of the $\lambda^2 - \lambda + 1$ processes must be assigned a distinct masking set containing one of that process's accounts. This is a perfect-matching problem in the bipartite graph between accounts and masking sets, with an edge for each containment. Each account appears in exactly λ masking sets and each masking set contains exactly λ accounts, so the graph is λ-regular; König's theorem [34] yields a perfect matching. Lemma 9 then implements $(\lambda^2 - \lambda + 1)$-consensus on this family. ◀

For general λ, a cyclic difference cover gives a one-round family of the size needed below.

▶ **Theorem 19** (Cyclic family). *For every* $\lambda \geq 3$*,* CUAT^+_λ *supports a wait-free one-round* $\lfloor \lambda^2/2 \rfloor$*-consensus protocol.*

Proof. Put $g = \lfloor \lambda/2 \rfloor$, $h = \lceil \lambda/2 \rceil$, and $n = \lfloor \lambda^2/2 \rfloor$, so $g + h = \lambda$ and $gh = n/2$. Identify the accounts and processes with $\mathbb{Z}_n$, let $\mu(t) = \{t\}$, and define

$$D = \{0, 1, \ldots, g-1\} \cup \{g, 2g, \ldots, hg\}, \qquad S_t = D + t$$

for $t \in \mathbb{Z}_n$. The two parts of D are disjoint, so $|S_t| = |D| = g + h = \lambda$. Moreover, $x \in D + t$ exactly when $t \in x - D$; hence every account lies in exactly λ translates, and the family has uniform incidence.

It remains to show that the translates pairwise intersect. We have $(D+t)\cap(D+t')\neq\emptyset$ exactly when $t'-t\in D-D$. The consecutive part of D contributes the differences $[-(g-1),g-1]$. For each $j\in\{1,\dots,h\}$, subtracting $\{0,\dots,g-1\}$ from jg contributes $[(j-1)g+1,jg]$. These intervals tile $[1,hg]$, and their negatives cover $[-hg,-1]$. Since $hg=n/2$, these differences cover all of $\mathbb{Z}_n$. Thus the S_t pairwise intersect.

Finally, the translates are distinct. If a nonzero subgroup H stabilized D, then, since $0\in D$, we would have $H\subseteq D\subseteq[0,n/2]$. Its least positive generator d would satisfy $d\geq n/2$ because $n-d\in H$; the only remaining case $d=n/2$ would force D to be a union of pairs $\{x,x+n/2\}$, possible here only for $D=\{0,n/2\}$ and $\lambda=2$, which is excluded. Therefore the stabilizer is trivial. Since $0\in D$, process t owns $t\in S_t$, and Lemma 9 applied to the resulting n-clique gives the claim. ◀

Thus, for every $\lambda\geq 3$, the one-round consensus number lies between $\lfloor\lambda^2/2\rfloor$ and $\lambda^2-\lambda+1$. The upper bound is tight when $\lambda-1$ is a prime power. For general protocols, Lemma 8 requires intersection only across the two valency classes. Counting how many cross-class pairs each account can cover gives the following bound.

▶ **Theorem 20** (General upper bound). *For every $\lambda\geq 2$, a wait-free consensus protocol on $\mathsf{CUAT}^{+}_{\lambda}$ can solve consensus among at most λ^2 processes.*

Proof. Suppose Π solves n-consensus; if $n\leq 2$ the bound holds trivially, since $\lambda^2\geq 4$, so assume $n\geq 3$. Take C, $\mathcal{P}_0$, $\mathcal{P}_1$, and O as in Lemma 8, and write $s=|\mathcal{P}_0|+|\mathcal{P}_1|$. Uniformity first bounds the incidence of an account by λ: if d masking sets of size λ cover v accounts with common incidence m, then $vm=d\lambda$ by double counting, while $d\leq v$ because distinct processes own distinct accounts and each funds its transfer from an account of its own masking set; hence $m\leq\lambda$. For every account a on O, let i_a and j_a count its incidences among the pending masking sets of $\mathcal{P}_0$ and $\mathcal{P}_1$, respectively, and write $s_0=|\mathcal{P}_0|$ and $s_1=|\mathcal{P}_1|$, so $s=s_0+s_1$. Every cross-class pair must share an account, so $s_0s_1\leq\sum_a i_aj_a$. Uniformity makes $i_a+j_a=m$ for every account occurring at C, and each of the s pending masking sets has λ members, so $\sum_a i_a=s_0\lambda$, $\sum_a j_a=s_1\lambda$, and the number of such accounts is $v=s\lambda/m$. Cauchy–Schwarz gives $\sum_a i_a^2\geq(\sum_a i_a)^2/v$, whence

$$\sum_a i_aj_a=\sum_a i_a(m-i_a)=ms_0\lambda-\sum_a i_a^2\leq ms_0\lambda-\frac{(s_0\lambda)^2}{v}=s_0\lambda m\,\frac{s_1}{s}.$$

Combining with $s_0s_1\leq\sum_a i_aj_a$ and dividing by $s_0s_1>0$ gives $s\leq\lambda m\leq\lambda^2$. ◀

Equality requires $m=\lambda$, a constant split $i_a=c$ and $j_a=\lambda-c$ of every account between the classes, hence $s_0=\lambda c$ and $s_1=\lambda(\lambda-c)$, and every cross-pair covered exactly once. The following recursive construction realizes these conditions.

▶ **Lemma 21** (Coincident sets). *For every $\lambda\geq 2$, $\mathsf{CUAT}^{+}_{\lambda}$ supports a wait-free one-round λ-consensus protocol in which every process carries the same masking set.*

Proof. Take an object with $\mathcal{A}_0=T=\{a_1,\dots,a_\lambda\}$, $\mu(a_i)=\{p_i\}$ and $\beta_0\equiv 0$, and let every p_i carry the masking set T and the value $v=0$. Each p_i owns $a_i\in T$, and the λ invocations pairwise conflict, so they form a λ-clique of G and Lemma 9 applies: the first τ to linearize commits and replaces T by its new accounts, every later τ finds $T\not\subseteq\mathcal{A}$ and fails, and a loser identifies the committed transfer by the new accounts it created and adopts its issuer's proposal. All λ invocations carry T, so every account of T has incidence λ and Theorem 14 gives strong untraceability. ◀

A grid construction composes two such protocols while adding their sizes.

▶ **Theorem 22** (Grid composition). *For every $\lambda \geq 2$ and every $m \in \{1, \ldots, \lambda\}$, CUAT^+_λ supports a wait-free λm-consensus protocol. In particular $\mathsf{cons}(\mathsf{CUAT}^+_\lambda) \geq \lambda^2$.*

Proof. Induction on m; the case $m = 1$ is Lemma 21. For $m \geq 2$ write $m = u + w$ with $u, w \geq 1$ and split the λm processes into classes $\mathcal{P}_0$ of size λu and $\mathcal{P}_1$ of size λw.

Round 1. Each class runs the protocol supplied by the induction hypothesis, on its own objects. Every process of class v thereby holds a common *class value* w_v, which is the input of some process of that class.

Round 2. Partition $\mathcal{P}_0$ into λ blocks $X_1, \ldots, X_\lambda$ of size u and $\mathcal{P}_1$ into λ blocks $Y_1, \ldots, Y_\lambda$ of size w. On a new object with accounts $\{a_{ij}\}_{i,j\in[\lambda]}$, let every $p \in X_i$ carry the row $\{a_{ij} : j \in [\lambda]\}$ and every $q \in Y_j$ the column $\{a_{ij} : i \in [\lambda]\}$. Each masking set has λ accounts, and a_{ij} occurs in the invocations of the u processes of X_i and the w processes of Y_j, so incidence is $u + w = m$ on every account and Theorem 14 admits the round. Each process writes its class value to a single-writer register and invokes τ.

Row i and column j meet exactly in a_{ij}, so once some transfer commits, every transfer of the other class finds one of its accounts invalidated and fails; the committing transfers therefore all belong to one class, and at least one commits. A process that commits decides its own class value; one that fails identifies a committed transfer in the published state, reads off the class of its issuer, and adopts that class's value. Every process thus decides the winning class's value, which round 1 made common to that class and which is some process's input, so agreement and validity hold, and neither round loops.

Finally, the bipartite graph between the λm processes and the λ^2 accounts of the round-2 object has degree λ on the left and $m \leq \lambda$ on the right, so $|N(X)| \geq \lambda|X|/m \geq |X|$ for every set X of processes and Hall's theorem assigns each process a distinct account of its own masking set to fund from. Taking $m = \lambda$ gives λ^2 processes. ◀

The families of Lemma 21 and Theorem 22 satisfy more than uniform incidence: any two accounts of a masking set are exchanged by a symmetry of the whole family, a transposition of two grid columns or of two grid rows. Their sender posteriors are therefore uniform for every adversary whose prior is symmetric in the accounts, independently of the independent-inclusion idealization.

▶ **Theorem 23** (Consensus number of CUAT^+_λ). *For every $\lambda \geq 2$, $\mathsf{cons}(\mathsf{CUAT}^+_\lambda) = \lambda^2$.*

Proof. The upper bound is Theorem 20 and the lower bound is Theorem 22. ◀

For every $\lambda \geq 2$, $\lambda^2 > \lambda^2 - \lambda + 1$; hence multi-round protocols are strictly more powerful than one-round protocols under strong untraceability. The prime-power condition remains relevant only to attaining the one-round upper bound through a projective plane, the multi-round value carrying no arithmetic condition on λ.

Having characterized CUAT's consensus power through the conflict graph structure, we now examine how this structure affects parallelization opportunities and fairness guarantees.

5 Parallelization and Fairness

The consensus analysis uses the conflict graph at a critical configuration. For a fixed workload, the same graph determines sequential depth and parallel throughput. It also exposes a fairness limitation: an unfair scheduler can indefinitely prevent a conflicting transfer from succeeding.

5.1 Workload Parallelism

Fix a transfer workload $\mathcal{T}$. Its conflict graph $G_{\mathcal{T}} = (V, E)$ has one vertex per transfer, including distinct transfers with the same masking set, and an edge between transfers whose masking sets share an index. Adjacent transfers cannot both succeed in one round; non-adjacent transfers commute. Between rounds, a pending transfer may be *rebased* by replacing each account with the current representation of the same index.

▶ **Theorem 24** (Sequential depth). *Suppose there is no external interference and every transfer remains balance-eligible after rebasing. Then $\chi(G_{\mathcal{T}})$ is the minimum number of rounds in which all transfers in $\mathcal{T}$ can succeed.*

Proof. A proper k-coloring partitions V into independent sets $V_1, \ldots, V_k$. Rebase and execute V_i in round i. Its transfers use disjoint indices and therefore all succeed. Conversely, color each transfer by its round in any successful k-round schedule. Transfers in one round cannot share an index, so this is a proper k-coloring. Hence $\chi(G_{\mathcal{T}}) \leq k$. ◀

The independence number gives the corresponding one-round throughput.

▶ **Theorem 25** (Parallel throughput). *The maximum number of transfers in $\mathcal{T}$ that can succeed in one round is $\alpha(G_{\mathcal{T}})$.*

Proof. The transfers in an independent set use disjoint indices, commute, and all succeed in one round. Thus $\alpha(G_{\mathcal{T}})$ is achievable. Conversely, transfers that all succeed in one round are pairwise non-conflicting and therefore form an independent set. ◀

5.2 Starvation

Wait-freedom guarantees that an invocation terminates, not that it succeeds. An unfair asynchronous scheduler may repeatedly resolve conflicts against the same process. This formalizes a denial-of-service behavior conjectured for *Quisquis* by Bünz *et al.* [9].

▶ **Definition 26** (Starvation-freedom). *An implementation of* CUAT_λ *is* starvation-free *if, for every correct process p, every infinite execution in which p invokes infinitely many eligible transfers contains infinitely many successful invocations by p. Eligibility is evaluated at invocation: the sender account is owned by p, its balance covers the chosen value, and the masking set is valid. Successive invocations may follow the current representations of the same logical accounts through rerandomization.*

This definition requires neither bounded waiting nor proportional success, only infinitely many successes for a process that remains active and eligible.

▶ **Theorem 27** (Starvation). *No* CUAT_λ *implementation is starvation-free under an asynchronous, unfair scheduler. A correct process may fail infinitely often while another correct process succeeds infinitely often on the same masking-set lineage. This holds even for zero-value transfers.*

Proof. Fix a masking set S containing accounts a and $a^\star$, owned by processes p and $p^\star$, respectively. Let $b, b^\star \in S$ be arbitrary destinations. Both processes repeatedly invoke transfers on the current representations of these accounts. We construct an infinite execution in which $p^\star$ wins every conflict.

Round r. Process p invokes $\tau_r = \mathit{transfer}(S'_p, \pi_p)$ with $\pi_p = (S, a, b, v, p)$ for some $v \leq \beta(a)$. The transfer is eligible when invoked. The scheduler pauses p before its linearization point and lets $p^\star$ complete its zero-value transfer $\tau_r^\star = \mathit{transfer}(S'_\star, \pi^\star)$ with witness $\pi^\star = (S, a^\star, b^\star, 0, p^\star)$. This invocation succeeds: S is valid and $\beta(a^\star) \geq 0$ at its linearization point. It replaces S by $S'_\star$ and succeeds.

The scheduler then resumes p. Its old masking set is invalid, so τ_r returns `FALSE`.

Both processes rebase before the next round. Because $p^\star$ transfers value zero, its balance is unchanged and it remains eligible. Repeating the schedule makes every invocation of p fail while $p^\star$ succeeds once per round. ◀

5.3 Implications and Mitigations

The graph parameters give an operational interpretation of contention. $\chi(G_\mathcal{T})$ is the minimum latency, in sequential rounds, for a fixed eligible workload, whereas $\alpha(G_\mathcal{T})$ is its maximum one-round throughput. These quantities depend on the placement of masking sets, not only on their size. Two workloads with the same number of transfers and the same anonymity parameter may therefore have very different execution costs. They also concern only the conflicting portion of the workload: isolated vertices and independent components can be processed without a common order.

This observation exposes two denial-of-service surfaces. First, a sender that can influence its masking set can introduce edges incident to selected honest transfers, increasing sequential depth and reducing parallel throughput. Second, control over scheduling can repeatedly resolve an existing edge against the same transfer. Zero-value transfers make the second attack particularly cheap because the adversary can rerandomize the masking-set lineage without depleting its balance. The starvation theorem requires neither an invalid proof nor insufficient funds; every adversarial transfer is valid when it commits. These results apply when a transfer consumes its entire masking set. They do not apply when a transfer only references its anonymity set. Monero rings and Zcash note-commitment sets retain their decoys; a repeated key image or nullifier indicates a double spend rather than masking-set-wide contention. Anonymous Zether [9, 14] similarly avoids masking-set-wide invalidation by deriving the relevant nonce from the sender's secret key. These mechanisms are closer to LUAT, analyzed in the full version [10]. Sui [8] and Zef [5] similarly process disjoint public transfers independently and coordinate only contending operations.

Several complementary mechanisms can limit these attacks. First, the protocol can derive masking sets from public, unpredictable randomness rather than let senders choose them freely. A verifiable random function or randomness beacon can make the selection auditable while preventing an adversary from targeting a particular transfer [30]; the design must also limit grinding over candidate sets. This controls how adversarially the conflict graph can be formed, but does not resolve conflicts that occur naturally.

Second, admission control can raise the cost of repeated contention. Fees, rate limits, per-account quotas, or restrictions on zero-value transfers make an infinite sequence of interfering transfers expensive or impossible within a fixed interval. Such measures bound the rate of an attack rather than eliminate the underlying schedule, and they may also penalize honest users who retry after a conflict.

Third, a fair broadcast or ordering layer can ensure that a transfer which remains pending is eventually placed before later conflicting transfers. This directly addresses starvation, but changes the assumptions of Theorem 27: the scheduler is no longer unfair, and the system may require synchrony, randomized leader election, or Byzantine agreement. Order-fair Byzantine consensus constrains adversarial transaction reordering [27, 11], while causal or

atomic group broadcast preserves dependencies between related messages [7, 4]. Causal order alone does not prevent starvation between concurrent transfers, so an inclusion or order-fairness condition is still needed. Randomized contention management offers a lighter alternative under a weaker progress condition [24], with probabilistic rather than deterministic success guarantees.

Finally, validators can use the conflict graph constructively. Independent sets can be batched and verified in parallel, while a coloring gives a schedule for rebasing and executing a contended batch. Overall, these measures call for a holistic design that jointly controls masking-set selection, admission, ordering, and execution.

6 Conclusion

We introduced λ-CUAT to study the synchronization cost of constant-state sender untraceability. Under weak untraceability, CUAT has unbounded consensus number, even for one-round protocols. Under strong untraceability, accounts in a common masking set must have uniform incidence. For one-round protocols, this condition bounds the consensus number by $\lambda^2-\lambda+1$; projective planes attain this bound whenever $\lambda-1$ is a prime power, and cyclic constructions give the general lower bound $\lfloor\lambda^2/2\rfloor$. For general protocols, our recursive grid construction matches the upper bound for every $\lambda \geq 2$, yielding $\mathsf{cons}(\mathsf{CUAT}^{+}_{\lambda}) = \lambda^2$.

The conflict graph connects these worst-case bounds to concrete workloads: its coloring and independent-set structure determine sequential depth and one-round throughput. It also exposes a progress cost, since CUAT cannot guarantee starvation-freedom under an unfair asynchronous scheduler. These results answer the question posed in the introduction. Consuming a masking set to bound ledger state turns a privacy mechanism into mutable shared state and thereby couples sender untraceability to synchronization and fairness. The consensus numbers measure the agreement power available in the worst case, not a coordination requirement for every transfer; transfers with disjoint masking sets commute and can be processed independently.

The full version [10] analyzes the complementary λ-LUAT abstraction, which captures the classical growing-state nullifier approach used by systems such as Monero [1] and Zcash [6]. It proves that LUAT has consensus number 2 for every masking-set size and under both untraceability notions, and that LUAT is starvation-free. It also studies partitioning objects and decoy-pruning mechanisms for reclaiming the state accumulated by LUAT. The comparison isolates the design trade-off: LUAT pays for untraceability through growing state, whereas CUAT keeps state constant at the cost of stronger synchronization and weaker fairness.

A central direction for future work is an end-to-end constant-state ledger that addresses privacy, progress, and efficiency jointly. Such a design should derive masking sets from unpredictable, publicly verifiable randomness while limiting grinding; guarantee fair inclusion of conflicting transfers under explicit scheduling and fault assumptions; and control repeated contention without creating a new denial-of-service vector. Compact proofs, aggregation, and conflict-aware batching may also be needed to keep communication practical as masking sets grow. Combining these mechanisms would clarify whether a ledger can bound its state without sacrificing fairness or practical throughput.

References

1 Kurt M. Alonso and The Monero Community. *Zero to Monero: Second Edition — Second Edition: First Principles and Cryptonote Cryptography*. The Monero Project, 2020. Version 2.0.0, April 2020. URL: https://web.getmonero.org/library/Zero-to-Monero-2-0-0.pdf.

2 Orestis Alpos, Christian Cachin, Giorgia Azzurra Marson, and Luca Zanolini. On the synchronization power of token smart contracts. In *41st IEEE International Conference on Distributed Computing Systems, ICDCS 2021, Washington DC, USA, July 7-10, 2021*, pages 640–651. IEEE, 2021. doi:10.1109/ICDCS51616.2021.00067.

3 Jayamine Alupotha, Mathieu Gestin, and Christian Cachin. Nopenena untraceable payments: Defeating graph analysis with small decoy sets. *IACR Cryptol. ePrint Arch.*, page 903, 2024. URL: https://eprint.iacr.org/2024/903.

4 Mariarosaria Barbaraci and Christian Cachin. Byzantine reliable broadcast with causal ordering: (research statement). In Dennis Olivetti and Eric Ruppert, editors, *ApPLIED*, pages 13:1–13:5. ACM, 2026. doi:10.1145/3820355.3820392.

5 Mathieu Baudet, Alberto Sonnino, Mahimna Kelkar, and George Danezis. Zef: Low-latency, scalable, private payments. arXiv:2201.05671, 2022. arXiv:2201.05671.

6 Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 459–474. IEEE Computer Society, 2014. doi:10.1109/SP.2014.36.

7 Kenneth P. Birman, André Schiper, and Pat Stephenson. Lightweight causal and atomic group multicast. *ACM Trans. Comput. Syst.*, 9(3):272–314, 1991. doi:10.1145/128738.128742.

8 Sam Blackshear, Andrey Chursin, George Danezis, Anastasios Kichidis, Lefteris Kokoris-Kogias, Xun Li, Mark Logan, Ashok Menon, Todd Nowacki, Alberto Sonnino, Brandon Williams, and Lu Zhang. Sui lutris: A blockchain combining broadcast and consensus. arXiv:2310.18042, 2023. doi:10.48550/arXiv.2310.18042.

9 Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. In Joseph Bonneau and Nadia Heninger, editors, *Financial Cryptography and Data Security - 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10-14, 2020 Revised Selected Papers*, volume 12059 of *Lecture Notes in Computer Science*, pages 423–443. Springer, 2020. doi:10.1007/978-3-030-51280-4_23.

10 Christian Cachin, David Lehnherr, Juan Villacis, and François-Xavier Wicht. The consensus number of untraceable cryptocurrencies. Cryptology ePrint Archive, Paper 2026/1503, 2026. URL: https://eprint.iacr.org/2026/1503.

11 Christian Cachin, Jovana Micic, Nathalie Steinhauer, and Luca Zanolini. Quick order fairness. In *Financial Cryptography*, pages 316–333, 2022. doi:10.1007/978-3-031-18283-9_15.

12 Miranda Christ and Joseph Bonneau. Limits on revocable proof systems, with implications for stateless blockchains. In Foteini Baldimtsi and Christian Cachin, editors, *Financial Cryptography and Data Security - 27th International Conference, FC 2023, Bol, Brač, Croatia, May 1-5, 2023, Revised Selected Papers, Part II*, volume 13951 of *Lecture Notes in Computer Science*, pages 54–71. Springer, 2023. doi:10.1007/978-3-031-47751-5_4.

13 Eli Daian, Giuliano Losa, Yehuda Afek, and Eli Gafni. A wealth of sub-consensus deterministic objects. In Ulrich Schmid and Josef Widder, editors, *32nd International Symposium on Distributed Computing, DISC 2018, New Orleans, LA, USA, October 15-19, 2018*, volume 121 of *LIPIcs*, pages 17:1–17:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.DISC.2018.17.

14 Benjamin E. Diamond. Many-out-of-many proofs and applications to anonymous zether. In *SP*, pages 1800–1817, 2021. doi:10.1109/SP40001.2021.00026.

15 Christoph Egger, Russell W. F. Lai, Viktoria Ronge, Ivy K. Y. Woo, and Hoover H. F. Yin. On defeating graph analysis of anonymous transactions. *Proc. Priv. Enhancing Technol.*, 2022(3):538–557, 2022. doi:10.56553/POPETS-2022-0085.

16 Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio Orlandi. Quisquis: A new design for anonymous cryptocurrencies. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 649–678. Springer, 2019. doi:10.1007/978-3-030-34578-5_23.

17 Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, April 1985. doi:10.1145/3149.214121.

18 Davide Frey, Mathieu Gestin, and Michel Raynal. The synchronization power (consensus number) of access-control objects: the case of allowlist and denylist. In Rotem Oshman, editor, *37th International Symposium on Distributed Computing, DISC 2023, October 10-12, 2023, L'Aquila, Italy*, volume 281 of *LIPIcs*, pages 21:1–21:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.DISC.2023.21.

19 Zoltán Füredi. Maximum degree and fractional matchings in uniform hypergraphs. *Combinatorica*, 1(2):155–162, 1981. doi:10.1007/BF02579271.

20 David K. Gifford. Weighted voting for replicated data. In Michael D. Schroeder and Anita K. Jones, editors, *Proceedings of the Seventh Symposium on Operating System Principles, SOSP 1979, Asilomar Conference Grounds, Pacific Grove, California, USA, 10-12, December 1979*, pages 150–162. ACM, 1979. doi:10.1145/800215.806583.

21 Rachid Guerraoui, Petr Kuznetsov, Matteo Monti, Matej Pavlovic, and Dragos-Adrian Seredinschi. The consensus number of a cryptocurrency. *Distributed Comput.*, 35(1):1–15, 2022. doi:10.1007/S00446-021-00399-2.

22 Timothy L. Harris, Keir Fraser, and Ian A. Pratt. A practical multi-word compare-and-swap operation. In Dahlia Malkhi, editor, *Distributed Computing, 16th International Conference, DISC 2002, Toulouse, France, October 28-30, 2002 Proceedings*, volume 2508 of *Lecture Notes in Computer Science*, pages 265–279. Springer, 2002. doi:10.1007/3-540-36108-1_18.

23 Maurice Herlihy. Wait-free synchronization. *ACM Trans. Program. Lang. Syst.*, 13(1):124–149, 1991. doi:10.1145/114005.102808.

24 Maurice Herlihy, Victor Luchangco, and Mark Moir. Obstruction-free synchronization: Double-ended queues as an example. In *23rd International Conference on Distributed Computing Systems (ICDCS 2003)*, pages 522–529. IEEE Computer Society, 2003. doi:10.1109/ICDCS.2003.1203503.

25 Maurice Herlihy and Jeannette M. Wing. Axioms for concurrent objects. In *Conference Record of the Fourteenth Annual ACM Symposium on Principles of Programming Languages, Munich, Germany, January 21-23, 1987*, pages 13–26. ACM Press, 1987. doi:10.1145/41625.41627.

26 Ferdinand Ihringer and Andrey Kupavskii. Regular intersecting families. *Discrete Applied Mathematics*, 270:142–152, 2019. doi:10.1016/j.dam.2019.07.009.

27 Mahimna Kelkar, Fan Zhang, Steven Goldfeder, and Ari Juels. Order-fairness for byzantine consensus. In *CRYPTO (3)*, pages 451–480, 2020. doi:10.1007/978-3-030-56877-1_16.

28 Amrit Kumar, Clément Fischer, Shruti Tople, and Prateek Saxena. A traceability analysis of monero's blockchain. In *ESORICS (2)*, pages 153–173, 2017. doi:10.1007/978-3-319-66399-9_9.

29 Varun Madathil and Alessandra Scafuro. sfpri tt fhesfte: Achieving full-privacy in account-based cryptocurrencies is possible. In Goichiro Hanaoka and Bo-Yin Yang, editors, *Advances in Cryptology - ASIACRYPT 2025 - 31st International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, VIC, Australia, December 8-12, 2025, Proceedings, Part VII*, volume 16251 of *Lecture Notes in Computer Science*, pages 199–229. Springer, 2025. doi:10.1007/978-981-95-5122-4_7.

30 Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *FOCS*, pages 120–130, 1999. doi:10.1109/SFFCS.1999.814584.

31 Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 397–411. IEEE Computer Society, 2013. doi:10.1109/SP.2013.34.

32 Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, and Nicolas Christin. An empirical analysis of traceability in the monero blockchain. *Proc. Priv. Enhancing Technol.*, 2018(3):143–163, 2018. doi:10.1515/POPETS-2018-0025.

33 Robert H. Thomas. A majority consensus approach to concurrency control for multiple copy databases. *ACM Trans. Database Syst.*, 4(2):180–209, 1979. doi:10.1145/320071.320076.

34 Douglas B West. *Introduction to graph theory*. Prentice Hall, Upper Saddle River, NJ, 2 edition, August 2000.

Reserve Depletion and Security Runway in Proof-of-Stake Systems

Paolo Penna
IOG, Zurich, Switzerland

Manvir Schneider
Cardano Foundation, Zurich, Switzerland

Abstract

Many proof-of-stake protocols finance validator rewards from two sources: transaction fees and a finite reserve of native tokens. This creates a dynamic hand-off problem. Early in the life of the system, fees may be too small to fund the target level of security; later, fees may become sufficient. The central question is whether the reserve provides enough runway for the protocol to remain secure until this fee-only region is reached. We study this problem in a discrete-time stochastic model of validator participation. Token price and transaction demand fluctuate over time, while validators choose participation strategically in each state. We solve the validator entry game in closed form and derive an exact state-dependent reserve threshold: for every token price and demand state, the threshold is the minimal reserve stock necessary and sufficient to sustain a target security level. This threshold separates three regions: infeasibility, reserve-dependent security, and fee-only security. The threshold turns reserve adequacy into a hitting-time problem. Security fails when the reserve first falls below the state-dependent threshold, and a successful hand-off occurs exactly when the fee-only region is reached before that failure time. We derive conservative finite-horizon stress-test guarantees that convert lower confidence bands for token price and demand into reserve requirements, and under lognormal price-demand dynamics we obtain explicit failure-probability and expected hand-off-time bounds. Finally, we extend the model to forward-looking validators and derive the Markov participation condition that captures how current participation affects future reserve-funded rewards. The main implication is that reserve policy should not be evaluated by nominal depletion dates or steady-state reward ratios alone. A protocol can have a large nominal reserve and still be close to security failure after adverse price or demand shocks. Conversely, once demand crosses the fee-only threshold, the reserve becomes redundant for security. This paper provides a tractable equilibrium framework for stress-testing this transition.

2012 ACM Subject Classification Theory of computation → Algorithmic game theory and mechanism design; Security and privacy → Economics of security and privacy; Mathematics of computing → Stochastic processes

Keywords and phrases Blockchain, Proof-of-Stake, Reserve Depletion, Security Runway, Validator Incentives, Tokenomics, Economic Security, Transaction Fees, Nash Equilibrium, Markov-Perfect Equilibrium, Stochastic Games

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.6

Related Version *Full Version*: `https://arxiv.org/abs/2606.03587` [22]

Acknowledgements We thank members of the Cardano Foundation and IOG for helpful discussions.

1 Introduction

Proof-of-stake blockchains typically pay validators from two sources: current transaction fees and a reserve of previously minted or retained tokens. The reserve is meant to bridge the early phase of the system, when adoption is still low and fees alone do not support the amount of active stake needed for security. Cardano is a canonical example of such a design: its reward system combines transaction fees with a controlled drawdown of reserves [4].



© Paolo Penna and Manvir Schneider;
licensed under Creative Commons License CC-BY 4.0



Other examples include Avalanche [3], Algorand [1], Ripple [2], and Hedera [11]. The policy problem is therefore not simply whether fee revenue is positive in the long run. The relevant question is whether the protocol can remain secure *along the entire transition path*.

That transition problem has two distinct sources of fragility. First, fee income is uncertain because transaction demand is uncertain. Second, even if the reserve is large in token units, its purchasing power is stochastic because validator costs are incurred in an external numeraire while the reserve is held in the native token. A reserve that looks ample at one token price can become inadequate after a price drawdown.

This creates a governance problem that is not visible from the reserve balance alone. Suppose two protocols have the same number of reserve tokens. The first has strong current demand and a high token price; the second faces weak demand and a price drawdown. Their nominal reserves are identical, but their ability to fund validator participation is not. Conversely, a protocol with a declining reserve may be safe if fee demand has already grown enough to support the target security level without subsidies. The relevant object is therefore a state-contingent reserve requirement.

The security-runway perspective is designed to capture exactly this issue. A protocol has runway at time t if the reserve stock at that date is large enough, given the current state of demand and token price, to keep equilibrium security above the required minimum. Runway ends not when the reserve literally reaches zero, but when it first becomes too small to finance the security shortfall.

This paper studies the following fundamental question:

> *Can a blockchain protocol guarantee secure operation throughout the transition from reserve-funded to fee-only validator rewards, and under what conditions does this transition fail?*

We answer this question by deriving a reserve threshold that can be interpreted as a state-contingent runway criterion: given price, demand, and a target security level, it tells whether the current reserve is sufficient to sustain equilibrium security. This allows reserve policy to be stress-tested state by state.

Our Contribution

We study the reserve-to-fee transition in a stochastic state model with two exogenous state variables: token price and user demand for blockspace. Validators choose participation strategically, so equilibrium security and fees respond to the current reserve, price, and demand state. Our contributions are as follows.

- **(i)** We solve the validator participation game in closed form and prove existence and uniqueness of a symmetric Nash equilibrium (Theorem 3). The equilibrium shows that the security value of a given reserve stock depends on the current token price.
- **(ii)** We derive an exact state-dependent reserve threshold (Theorem 10). For each price-demand state, the threshold is the minimal reserve stock necessary and sufficient to sustain a target security level.
- **(iii)** We use this threshold to formulate the dynamic hand-off problem (Section 4). Security persists while the reserve remains above the threshold, and hand-off succeeds when the fee-only region is reached before failure (Proposition 13, Theorem 15, Corollary 17).
- **(iv)** We extend the baseline model to a forward-looking Markov environment (Section 5). We prove finite-horizon Markov perfect equilibrium existence and characterize how continuation values alter current validator participation (Theorem 22, Propositions 23, 25, and 27).

(v) We specialize the runway analysis to geometric Brownian token price and discrete-time lognormal demand (Section 6). This yields explicit finite-horizon failure-probability guarantees and bounds on hand-off timing (Theorem 30, Proposition 31).

Related Literature

Our analysis is related to three strands of literature. First, the Ouroboros line of work studies the protocol and security foundations of proof-of-stake systems and provides the incentive-theoretic background for stake-based consensus [18, 9]. Second, economic analyzes of proof-of-stake study equilibrium incentives, reward design, and the distributional effects of staking rewards [23, 10, 15]. Third, the blockchain fee-market literature studies how congestion pricing and transaction fees support decentralized infrastructure [13, 25]. Dynamics models for cryptocurrencies include [17, 14, 6, 16], while geometric Brownian motion is used, e.g., in [7, 8]. To the best of our knowledge, existing work has not isolated the reserve hand-off problem studied here: a finite native-token reserve, strategic validator participation, stochastic token price and demand, and a state-dependent threshold for sustaining a target security level until fee-only operation becomes possible. Earlier work models the effects of fee-only rewards in Bitcoin: First, higher variance induces "forking" strategies of miners competing for transactions with high tips [5]. [24] further show that mining gaps can emerge even before block rewards disappear, with adverse effects on security and decentralization. Moreover, attackers can bribe honest participants by offering tips for mining a forked chain and increase the chances of successfully performing double spending [19].

Contribution Relative to the Literature

The present paper differs from these strands by focusing on a dynamic reserve-funded transition problem. The object of interest is not simply equilibrium rewards at a point in time, nor fee formation in isolation, but the reserve threshold required to sustain security until fee funding becomes sufficient on its own. That is the role of the security-runway concept developed below.

Our model has some similarities with the validator side in [17]: there, validators decide the amount of blockspace (or security) to provide, and their utility depends on the per-unit fee of the system as equilibrium. That paper, however, does not consider reserve constraints, and it models users demand differently.

In our model, both token prices and demand are exogenous quantities. In [14, 16] instead, the system value is determined by the protocol evolution (at some cost function) and the token price is an endogenous quantity which is part of the equilibrium (the price at which tokens are exchanged in a spot market involving validators and users). In this respect, our model exhibits greater flexibility and generality, while additionally incorporating the aforementioned reserve constraint, which is absent in [14, 16].

Deferred Proofs

All proofs are available in the appendix of the full version of the paper [22].

2 Model

Time is discrete and indexed by $t = 0, 1, 2, \ldots$. All random variables are defined on a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0}, \mathbb{P})$. The protocol state at time t is given by a triple $X_t = (R_t, Price_t, FeeMax_t)$, where:

- $R_t \in \mathbb{R}_+$ is the reserve stock measured in native-token units.
- $Price_t \in \mathbb{R}_{++}$ is the token price in an external numeraire. Validator costs are paid in this numeraire.
- $FeeMax_t \in \mathbb{R}_{++}$ is a demand state governing willingness to pay for block space. Intuitively, this is the largest fee that at least some user is willing to pay (any higher fee results in zero demand).

The policy parameters are fixed ex ante:

$$\theta \in [0, 1), \qquad \rho \in (0, 1),$$

where θ is the share of fee revenue diverted into the reserve and ρ is the fraction of the reserve paid out each period. Token price and demand are stochastic, exogenous, and positive quantities:

▶ **Assumption 1** (State dynamics). *The token price process $(Price_t)_{t\geq 0}$ and the demand process $(FeeMax_t)_{t\geq 0}$ are strictly positive and adapted to the filtration $(\mathcal{F}_t)_{t\geq 0}$.*

2.1 Game Form and Economic Scope

The players are the N validators. Their date-t actions are participation levels, interpreted as active stake or security supply. Users are not modeled as strategic players; instead, user behavior is summarized by an inverse-demand schedule for block space. Likewise, token-market trading is not modeled strategically; it enters through the stochastic price process $Price_t$. This separation isolates the security-runway question: how the reserve interacts with validator incentives when current fee conditions and the external token price fluctuate over time.

This modeling choice has two implications that should be kept distinct throughout the paper. First, the token price $Price_t$ is externally given to validators at date t, so the model does not claim to explain token valuation. Second, the transaction fee f_t is endogenous, because it is the market-clearing fee induced by validator participation and the current demand state $FeeMax_t$. Thus demand and token price are state variables, while validator participation and fees are equilibrium outcomes.

2.2 A Reduced-Form Security Technology

There are $N \geq 2$ validators. Validator i chooses an amount of active stake, or security supply,

$$a_{i,t} \in \mathbb{R}_+,$$

and aggregate security and blockspace is

$$s_t := \sum_{i=1}^{N} a_{i,t}. \tag{1}$$

The baseline model is symmetric across validators: every validator has the same feasible action set, the same cost function, and the same proportional reward-sharing rule. [1]

The action $a_{i,t}$ should be interpreted as validator i's effective supply of secure blockspace at date t. It combines the validator's active participation with the amount of transaction-processing capacity that the validator makes available to users. Thus, a validator who is

[1] The model therefore suppresses ex ante heterogeneity in stake, scale, or operating cost.

online but produces empty or only partially filled blocks is represented as supplying a lower effective $a_{i,t}$. The aggregate s_t is therefore the quantity of secure blockspace supplied in equilibrium. It enters the security side of the model because larger effective participation raises the security level, and it enters the demand side because it is the quantity of blockspace cleared at the market fee. This convention keeps the model focused on the funding problem: how much effective secure capacity validators are willing to supply when rewards come from fees and reserve payouts.

Users are described by an *inverse demand curve* expressed in terms of the external numeraire (for example, in dollars):

$$P(s_t, FeeMax_t) := FeeMax_t - Sec \cdot s_t \ . \tag{2}$$

Intuitively, given the available blockspace provided by the validators (1), Equation 2 gives the *market-clearing* fee:

- Users demand equals the supplied blockspace s if the protocol charges $P(s, FeeMax)$ dollars per unit of blockspace. Note that we are assuming each user to consume one unit of blockspace – say one transaction. Hence, the demand corresponding to (2) is of the form $D(p) = (FeeMax - p)/Sec$ for $p \in [0, FeeMax]$, and $D(p) = 0$ for $p > FeeMax$.
- The parameter $Sec > 0$ determines the rate at which market-clearing fees decline as total blockspace supply increases (intuitively, the protocol must lower fees to accommodate higher user demand).

As already mentioned above, while $FeeMax_t$ is an exogenous parameter controlling users demand, blockspace s_t is strategic and results from validators aiming at maximizing their own utilities. If the supplied blockspace is s_t, the protocol posts a token-denominated fee

$$f_t(s_t, FeeMax_t, Price_t) := \frac{P(s_t, FeeMax_t)}{Price_t} = \frac{FeeMax_t - Sec \cdot s_t}{Price_t} \ . \tag{3}$$

The resulting fee revenue, in external numeraire, is thus

$$Price_t \cdot f_t \cdot s_t = (FeeMax_t - Sec \cdot s_t) \cdot s_t \ .$$

The model restricts attention to action profiles that yield nonnegative clearing fees, reflecting the natural requirement that the system does not subsidize users for consuming blockspace. Since $P(s_t, FeeMax_t) \geq 0$ requires $s_t \leq FeeMax_t/Sec$, we impose the symmetric feasibility bound

$$a_{i,t} \in \left[0, \frac{FeeMax_t}{N \cdot Sec}\right] . \tag{4}$$

This condition implies that $0 \leq s_t \leq FeeMax_t/Sec$, ensuring that clearing fees are always nonnegative. Furthermore, in the context of symmetric equilibria – where all validators select an identical strategy – these two conditions are equivalent.

2.3 Reserve Dynamics and Validator Rewards

The reserve absorbs a fraction θ of current fee revenue and a fraction ρ of the reserve is paid out each period. Because the reserve is held in tokens, its law of motion is

$$R_{t+1} = (1 - \rho) \cdot R_t + \theta \cdot \frac{(FeeMax_t - Sec \cdot s_t) \cdot s_t}{Price_t}. \tag{5}$$

The total reward pool paid to validators in period t, measured in the external numeraire, is

$$W_t := (1-\theta)\cdot(FeeMax_t - Sec\cdot s_t)\cdot s_t + \rho\cdot Price_t\cdot R_t. \tag{6}$$

The first term is the portion of fees not diverted to the reserve; the second term is the external value of the reserve payout.

Each validator has quadratic operating cost

$$\frac{\kappa}{2}a_{i,t}^2, \qquad \kappa > 0\ . \tag{7}$$

The quadratic cost reflects increasing marginal costs: providing additional secure blockspace becomes more costly as a validator's supply grows. Kiayias et al. [17] use a general convex cost for blockspace provision, while Houy [12] uses a quadratic cost. The quadratic form also keeps the first-order condition tractable whenever the capacity bound does not bind.

2.4 Stage-Game Equilibrium

In this section, we consider a single-stage game which is fully specified by the current state: [2] the current reserve R_t, token price $Price_t$, and demand $FeeMax_t$. As we focus only on a generic stage t, we suppress t and consider a generic state $x = (R, Price, FeeMax)$. The stage game at state x is played only by the N validators. A pure action for validator i is a feasible action a_i satisfying (4), that is, $a_i \in [0, FeeMax/(N\cdot Sec)]$. Actions are chosen simultaneously. If the aggregate action profile is $\boldsymbol{a} = (a_1,\dots,a_N)$ and $s = \sum_i a_i > 0$, validator i receives its pro rata share of the reward pool (6) given the current state and action profile,

$$W = W(\boldsymbol{a}; x) := \frac{a_i}{s}\cdot\left((1-\theta)\cdot(FeeMax - Sec\cdot s)\cdot s + \rho\cdot Price\cdot R\right).$$

The resulting utility (reward minus incurred cost) is thus

$$u_i(\boldsymbol{a}; x) := \frac{a_i}{s}\cdot W - \frac{\kappa}{2}a_i^2 = \frac{a_i}{s}\left((1-\theta)\cdot(FeeMax - Sec\cdot s)\cdot s + \rho\cdot Price\cdot R\right) - \frac{\kappa}{2}a_i^2. \tag{8}$$

When all validators choose zero, we set $u_i(0,\dots,0;x) = 0$.

▶ **Definition 2** (Symmetric Nash equilibrium). *Fix a state $x = (R, Price, FeeMax)$. A* symmetric Nash equilibrium *is a profile $\boldsymbol{a}^*(x) = (a^*(x),\dots,a^*(x))$ such that*

$$u_i\big(a^*(x),\dots,a^*(x); x\big) \geq u_i\big(a_i, \boldsymbol{a}^*_{-i}(x); x\big)$$

for every validator i and every feasible deviation $a_i \in [0, FeeMax/(NSec)]$, where $(a_i, \boldsymbol{a}^*_{-i}(x))$ denotes the vector obtained by replacing the ith entry in profile $\boldsymbol{a}^*(x)$ with a_i.

The induced equilibrium security, fee, and next-period reserve are given by:

$$\begin{aligned} s^*(x) &= N\cdot a^*(x), \qquad f^*(x) \overset{(3)}{=} \frac{FeeMax - Sec\cdot s^*(x)}{Price}, \\ R^+(x) &= (1-\rho)\cdot R + \theta\cdot f^*(x)\cdot s^*(x) = (1-\rho)\cdot R + \theta\cdot\frac{(FeeMax - Sec\cdot s^*(x))\cdot s^*(x)}{Price}. \end{aligned}$$

[2] Later, we extend this game and its analysis to a dynamic "multi-stage" version.

Symmetric equilibria arise naturally in the context of blockchain systems (see, e.g., [14, 16]), as they constitute the intended design objective of many protocols: they correspond to a regime of maximal decentralization [21, 20].

The definition above is for a generic state and yields a state by state evolution in the natural way: Given the current state $x_t = (R_t, Price_t, FeeMax_t)$, the reserve at the next state is $R_{t+1} = R^+(x_t)$, and therefore the next state is $x_{t+1} = (R_{t+1}, Price_{t+1}, FeeMax_{t+1})$. Dynamic incentives are encoded only through the reserve stock carried to the next date; validators do not solve an intertemporal control problem. The analysis focuses on whether the current state provides enough funding to sustain current security, not on dynamic reputation effects.

Observe that the state is composed of three variables, of which two are exogenous: The reserve stock R is an endogenous state variable inherited from past protocol outcomes. The token price $Price$ and the demand shifter $FeeMax$ are exogenous state variables from the perspective of the validator game. Conditional on a state x, equilibrium participation $a^*(x)$ and the clearing fee $f^*(x)$ are endogenous objects solved within the model.

3 Static Equilibrium Analysis

This section solves the stage game exactly. The main result is a closed-form symmetric Nash equilibrium, which then becomes the building block for the dynamic runway analysis.

3.1 Existence and Uniqueness

Define the constants

$$A_N := \frac{\kappa}{N} + \frac{N+1}{N} \cdot (1-\theta) \cdot Sec\ , \qquad B_N := \frac{N-1}{N} \cdot \rho\ . \tag{9}$$

The next theorem shows the uniqueness of symmetric equilibria in the stage game. This result carries over to the multi-round dynamic game, as we prove below (Proposition 23). Note that the number of validators N does affect the equilibrium strategies through the two quantities defined above.

▶ **Theorem 3** (Unique symmetric Nash equilibrium). *For every state $x = (R, Price, FeeMax) \in \mathbb{R}_+ \times \mathbb{R}^2_{++}$, the stage game admits a unique symmetric Nash equilibrium. Its aggregate security level is*

$$s^*(x) = \min\left\{ \frac{(1-\theta)\cdot FeeMax + \sqrt{(1-\theta)^2 \cdot FeeMax^2 + 4A_N B_N \cdot Price \cdot R}}{2A_N}, \frac{FeeMax}{Sec} \right\}. \tag{10}$$

The corresponding equilibrium fee is $f^*(x) = \frac{FeeMax - Sec \cdot s^*(x)}{Price}$.

▶ Remark 4. The equilibrium formula shows exactly how the reserve enters incentives. Current fee funding depends on demand $FeeMax$ and congestion Sec, while reserve funding enters only through the product $Price \cdot R$, the reserve's value in *the external numeraire*. The same reserve stock therefore has different security consequences at different token prices. Hence, the reserve balance alone does not determine how much security it can support; its value must be considered together with the token price.

Theorem 3 implies that, in any symmetric equilibrium, the validators' utilities are given by the following expression.

▶ **Corollary 5.** *For every state $x = (R, Price, FeeMax) \in \mathbb{R}_+ \times \mathbb{R}^2_{++}$, the equilibrium utilities of the validators in the corresponding stage game are*

$$u_i^*(x) := \frac{1}{N} \cdot \big((1-\theta) \cdot (FeeMax - Sec \cdot s^*(x)) \cdot s^*(x) + \rho \cdot Price \cdot R\big) - \frac{\kappa}{2} \cdot \left(\frac{s^*(x)}{N}\right)^2 \tag{11}$$

$$= s^*(x) \cdot \frac{(1-\theta) \cdot FeeMax + \rho \cdot Price \cdot R}{N} - (s^*(x))^2 \cdot \left(\frac{(1-\theta) \cdot Sec}{N} + \frac{\kappa}{2N^2}\right) . \tag{12}$$

In particular, in any non-interior equilibrium, i.e. $s^*(x) = \frac{FeeMax}{Sec}$, the utilities are equal to

$$u_i^*(x) = \frac{1}{N} \cdot \big(\rho \cdot Price \cdot R\big) - \frac{\kappa}{2} \cdot \left(\frac{FeeMax}{N \cdot Sec}\right)^2 . \tag{13}$$

The second part of the corollary above implies that, for sufficiently large R, the utility of each validator is entirely reserve-driven. This is because the resulting equilibrium must be non-interior, and the corresponding market-clearing fee (3) is zero. The next corollary deals with the opposite case of small reserve (intuitively, when R is not large enough to make the utility in (13) nonnegative).

▶ **Corollary 6.** *For sufficiently small R, the unique symmetric equilibrium must be interior, that is,*

$$s^*(R, Price, FeeMax) < \frac{FeeMax}{Sec}.$$

In particular, this holds true for any $R < R_{\min} := \left(\frac{FeeMax}{N \cdot Sec}\right)^2 \cdot \frac{\kappa N}{2\rho \cdot Price} = \frac{\kappa FeeMax^2}{2\rho \cdot Price \cdot N \cdot Sec^2}$.

Proof. By contradiction, if $s^*(R, Price, FeeMax) \geq \frac{FeeMax}{Sec}$, then $u_i^*(x) = \frac{1}{N}(0 + \rho \cdot Price \cdot R) - \frac{\kappa}{2} \cdot \left(\frac{FeeMax}{N \cdot Sec}\right)^2 < 0$. But validator i can deviate to $a_i = 0$, yielding utility 0. Hence the boundary profile cannot be a Nash equilibrium. Therefore the equilibrium must be interior. ◀

Note that for any interior equilibria and for $Q := N \cdot (1-\theta) \cdot FeeMax$, we have

$$s^*(R, Price, FeeMax) = \frac{Q + \sqrt{Q^2 + 4 \cdot (\kappa + (N+1) \cdot (1-\theta) \cdot Sec) \cdot (N-1) \cdot \rho \cdot Price \cdot R}}{2(\kappa + (N+1) \cdot (1-\theta) \cdot Sec)} . \tag{14}$$

▶ **Example 7** (No reserve)**.** Consider $R = 0$, so validator rewards are funded entirely by fees. By Corollary 6, the equilibrium is interior. Hence, for $x = (0, Price, FeeMax)$ we have

$$s^*(x) \overset{(14)}{=} \frac{N \cdot (1-\theta) \cdot FeeMax}{\kappa + (N+1) \cdot (1-\theta) \cdot Sec} \overset{(9)}{=} \frac{(1-\theta) \cdot FeeMax}{A_N} . \tag{15}$$

Next, we calculate the utilities of the validators. By Corollary 5, with $R = 0$

$$\begin{aligned} u_i^*(x) =& \frac{1}{N} \cdot (1-\theta) \cdot (FeeMax - Sec \cdot s^*(x)) \cdot s^*(x) - \frac{\kappa}{2} \cdot \left(\frac{s^*(x)}{N}\right)^2 \\ \overset{(15)}{=}& \frac{1}{N} \cdot s^*(x) \cdot (A_N - (1-\theta) \cdot Sec) \cdot s^*(x) - \frac{\kappa}{2} \cdot \left(\frac{s^*(x)}{N}\right)^2 \\ =& \left(\frac{s^*(x)}{N}\right)^2 \cdot (N \cdot A_N - N \cdot (1-\theta) \cdot Sec) - \frac{\kappa}{2} \cdot \left(\frac{s^*(x)}{N}\right)^2 \\ \overset{(9)}{=}& \left(\frac{s^*(x)}{N}\right)^2 \cdot (\kappa + (N+1) \cdot (1-\theta) \cdot Sec - N \cdot (1-\theta) \cdot Sec) - \frac{\kappa}{2} \cdot \left(\frac{s^*(x)}{N}\right)^2 \\ =& \left(\frac{s^*(x)}{N}\right)^2 \cdot (\frac{\kappa}{2} + (1-\theta) \cdot Sec) > 0 \ . \end{aligned}$$

Thus, when $R = 0$, the equilibrium is interior and each validator obtains strictly positive utility.

The next result shows that equilibrium security increases when validator rewards become easier to finance. Higher demand raises current fee revenue, and a higher token price makes reserve payouts more valuable in external terms. By contrast, higher operating costs, stronger fee compression, or diverting a larger fraction of the fees into the reserve reduce the validators' current incentives to supply security and blockspace.

▶ **Proposition 8** (Monotonicity). *The equilibrium security level $s^*(R, Price, FeeMax)$ from Theorem 3 is weakly increasing in R, in $Price$, and in $FeeMax$, and weakly decreasing in κ, in Sec, and in θ.*

3.2 Security Threshold

In this section, we analyze the conditions under which the system sustains an equilibrium level of security (blockspace) above a threshold. Let $\underline{s} > 0$ denote the target security requirement.

▶ **Definition 9** (Security-feasible state). *A state $(R, Price, FeeMax)$ is* security feasible *for target $\underline{s}$ if*

$$s^*(R, Price, FeeMax) \geq \underline{s}.$$

It is fee-only feasible *if the same inequality holds at $R = 0$.*

The next theorem is the static core of the paper. It converts the equilibrium formula into a reserve threshold.

▶ **Theorem 10** (Exact reserve threshold). *For every target $\underline{s} > 0$, define*

$$SecureR(Price, FeeMax; \underline{s}) := \begin{cases} \dfrac{\left[A_N \cdot \underline{s}^2 - (1-\theta) \cdot FeeMax \cdot \underline{s}\right]_+}{B_N \cdot Price}, & \text{if } \underline{s} \leq FeeMax/Sec, \\ \infty, & \text{if } \underline{s} > FeeMax/Sec. \end{cases} \tag{16}$$

Then the following are equivalent:

(i) *$(R, Price, FeeMax)$ is security feasible for target $\underline{s}$.*

(ii) *$R \geq SecureR(Price, FeeMax; \underline{s})$.*

Moreover, $(R, Price, FeeMax)$ is fee-only feasible if and only if $SecureR(Price, FeeMax; \underline{s}) = 0$.

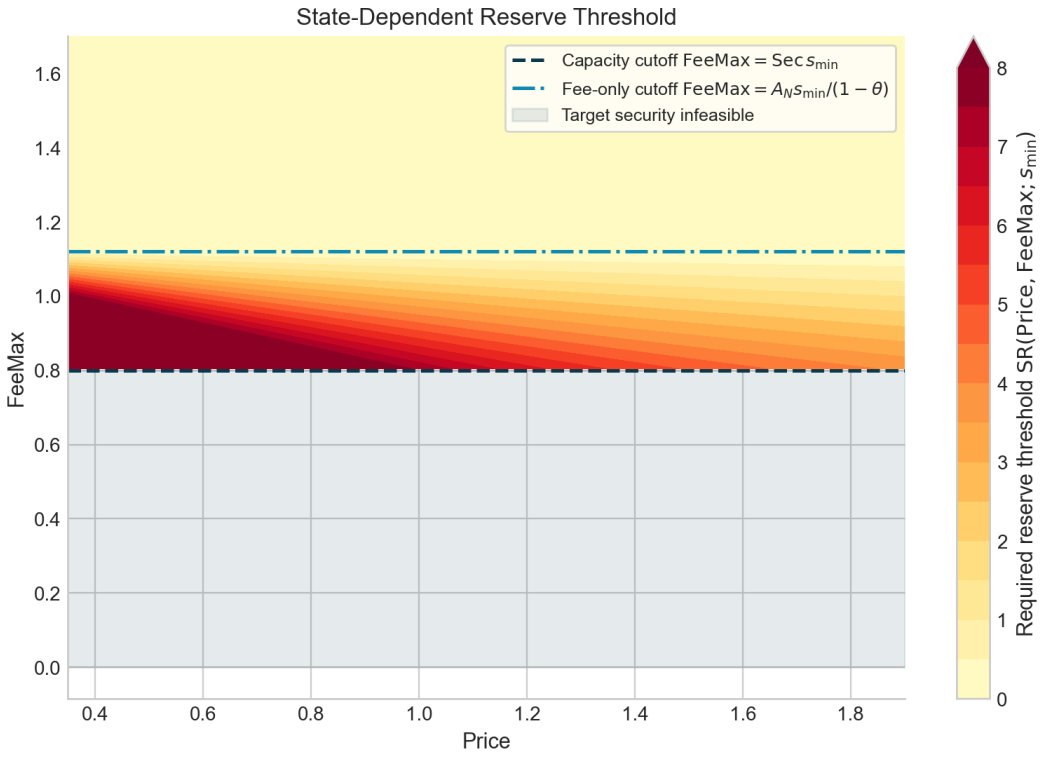


Figure 1 Exact reserve threshold as a function of token price $Price$ and demand state $FeeMax$. For any point in the $(Price, FeeMax)$ plane, the color gives the minimal reserve stock required to sustain the target security level $\underline{s}$: darker regions require larger reserves. The gray lower region lies below the capacity cutoff $FeeMax = Sec \cdot \underline{s}$; there the target is infeasible at any finite reserve level because aggregate supply cannot be increased enough while keeping fees nonnegative. Above the fee-only cutoff $FeeMax = A_N \cdot \underline{s}/(1-\theta)$, the reserve threshold is zero because current fees alone can sustain the target. The figure uses the illustrative parameter values $N = 5$, $\theta = 0.25$, $\rho = 0.04$, $\kappa = 0.6$, $Sec = 0.8$, and $\underline{s} = 1$.

Theorem 10 isolates the object that governance actually needs to monitor: not the reserve level alone, but the reserve level relative to a state-dependent threshold $SecureR(\cdot)$. When demand is strong or the token price is high, the threshold falls because either current fees or reserve purchasing power are more favorable. When demand is too weak to support the target capacity, no reserve level can repair the shortfall. The threshold becomes infinite when the target security level $\underline{s}$ exceeds current demand capacity $FeeMax/Sec$ (recall that by (4) this ratio is the largest feasible s, i.e., the protocol cannot buy more secure throughput at any nonnegative fee).

▶ Remark 11. The threshold $SecureR(\cdot)$ is easiest to interpret geometrically. Figure 1 plots the exact threshold over the $(Price, FeeMax)$ plane for an illustrative parameterization. The figure makes visible the three economically distinct regions identified by Theorem 10: (i) an infeasible low-demand region, (ii) a strictly reserve-dependent transition region, and (iii) a fee-only region where the threshold collapses to zero.

4 Dynamic Security Runway

The static threshold becomes dynamic once price and demand evolve over time and the reserve follows Equation (5). This section formalizes the hand-off problem.

4.1 Failure Time and Hand-Off Time

In this section, we study the hand-off time for myopic players, whose equilibrium is characterized at every step t by Theorem 3. In particular, the reserve evolution in (5) becomes

$$R_{t+1} = (1-\rho)R_t + \theta \frac{\big(FeeMax_t - Sec \cdot s^*(R_t, Price_t, FeeMax_t)\big) s^*(R_t, Price_t, FeeMax_t)}{Price_t}. \tag{17}$$

▶ **Definition 12** (Runway stopping times). *Fix a target security level $\underline{s} > 0$. Define*

$$\tau_{\text{fail}} := \inf\{t \geq 0 : R_t < SecureR(Price_t, FeeMax_t; \underline{s})\}, \tag{18}$$

$$\tau_{\text{hand}} := \inf\{t \geq 0 : SecureR(Price_t, FeeMax_t; \underline{s}) = 0\}. \tag{19}$$

We say that the protocol achieves a successful hand-off *if $\tau_{\text{hand}} < \tau_{\text{fail}}$.*

Next, we turn the hand-off problem into a hitting-time problem. Security lasts exactly as long as the reserve process remains inside the safe region, and the transition succeeds exactly when the protocol reaches the fee-only region before leaving that safe region. While a successful handoff is a necessary condition, it does not provide any guarantee for the system to be self-sustainable (i.e. stay in the fee-only region) after the handoff. As we prove below, we shall provide sufficient conditions under which the system does not fail before any target period T (Theorem 15).

▶ **Proposition 13** (Exact runway criterion). *For every horizon $T \in \mathbb{N}$ and every sample path:*

(i) *Security is maintained at all dates $t = 0, \ldots, T$ if and only if*

$$R_t \geq SecureR(Price_t, FeeMax_t; \underline{s}) \qquad \text{for all } t = 0, \ldots, T.$$

(ii) *A successful hand-off occurs if and only if the process reaches the fee-only region before the failure region, that is,*

$$\tau_{\text{hand}} < \tau_{\text{fail}}.$$

Proof. Part (i) follows directly from Theorem 10 applied state by state. Part (ii) is the definition of successful hand-off expressed in terms of the stopping times in Definition 12. ◀

4.2 A Conservative Stress-Test Bound

The exact criterion is pathwise. For protocol design one often wants a finite-horizon sufficient condition that can be checked before the system is launched. The next result gives such a bound.

▶ **Lemma 14** (Pure-decay lower bound). *Along every sample path and for every $t \geq 0$,*

$$R_t \geq (1-\rho)^t \cdot R_0.$$

Proof. The reserve recursion in Equation (17) has a nonnegative inflow term. Hence $R_{t+1} \geq (1-\rho) \cdot R_t$. Iterating yields the claim. ◀

Lemma 14 is the worst-case reserve benchmark: even if future fee inflows are ignored entirely, the reserve cannot decay faster than pure payout at rate ρ. This simple lower bound is what makes conservative stress testing possible.

The next theorem provides a robust sufficient condition: if the reserve is large enough to survive a pessimistic lower envelope for price and demand, then the protocol is safe on every path inside that envelope. This form is useful for ex ante certification and governance stress tests.

▶ **Theorem 15** (Finite-horizon stress-test guarantee). *Fix a horizon $T \in \mathbb{N}$ and deterministic lower envelopes $(\underline{Price}_t)_{t=0}^{T}$ and $(\underline{FeeMax}_t)_{t=0}^{T}$. Define the event*

$$E_T := \{Price_t \geq \underline{Price}_t,\ FeeMax_t \geq \underline{FeeMax}_t \quad \text{for all } t = 0, \ldots, T\}.$$

If

$$(1-\rho)^t R_0 \geq SecureR(\underline{Price}_t, \underline{FeeMax}_t; \underline{s}) \qquad \text{for all } t = 0, \ldots, T, \tag{20}$$

then event E_T implies that failure occurs after T, that is, $\tau_{\text{fail}} > T$. Consequently, for any $\alpha \in [0,1]$, if $\mathbb{P}(E_T) \geq 1-\alpha$, then $\mathbb{P}(\tau_{\text{fail}} > T) \geq 1-\alpha$.

▶ Remark 16. Theorem 15 is intended for protocol governance. One may estimate lower confidence bands for $Price_t$ and $FeeMax_t$, plug them into Equation (20), and obtain a conservative reserve requirement for a target horizon T. The bound is conservative because it ignores future reserve inflows from fees; it treats the reserve as if it only decayed. That makes it suitable for adverse-scenario certification.

4.3 Fee-Only Region and Long-Run Design

We identify the economic moment at which the reserve becomes redundant, that is, current demand is enough to finance the target security level. In this case, reserve policy no longer determines feasibility.

▶ **Corollary 17** (Fee-only region). *Suppose $\underline{s} \leq FeeMax/Sec$. Then fee-only feasibility at target $\underline{s}$ holds if and only if*

$$(1-\theta) \cdot FeeMax \geq A_N \cdot \underline{s}. \tag{21}$$

Proof. By Theorem 10, fee-only feasibility is equivalent to $SecureR(Price, FeeMax; \underline{s}) = 0$. Under $\underline{s} \leq FeeMax/Sec$, this is equivalent to

$$A_N \cdot \underline{s}^2 - (1-\theta) FeeMax \cdot \underline{s} \leq 0.$$

Because $\underline{s} > 0$, division by $\underline{s}$ yields Equation (21). ◀

▶ Remark 18. Equation (21) says that fee-only sustainability is governed by demand, *not* by token price. This is because current transaction fees paid by users are already measured in the external numeraire. Token-price risk matters only because the reserve is a stock of tokens (carried from the past).

5 Dynamic Validator Participation: A Markov Extension

The baseline model in considered so far is myopic: At state X_t, validators choose current participation to maximize current payoff only. Intuitively, this is what makes the reserve threshold explicit. In this section, we consider a fully strategic dynamic extension that endogenizes continuation values while preserving a precise equilibrium concept.

There are two technical issues. First, once validators become forward looking, a deviation at time t changes the future reserve stock and therefore future incentives. Second, proving equilibrium existence in the original continuous-state model requires a substantial stochastic-game apparatus that would obscure the runway results. We therefore proceed in two steps. We first formulate a finite-horizon finite-state Markov game and prove existence of a Markov perfect equilibrium. We then return to the continuous model and derive the exact first-order condition that any differentiable pure Markov equilibrium must satisfy.

5.1 Finite-Horizon Markov Game

Fix a horizon $T \in \mathbb{N}$ and a discount factor $\delta \in (0, 1)$.

▶ **Assumption 19** (Finite-state Markov extension)**.** *There is a finite set of exogenous states*

$$\mathcal{Z} = \{z^1, \dots, z^M\} \subset \mathbb{R}^2_{++}, \qquad z^m = (Price^m, FeeMax^m),$$

and a Markov transition matrix $P = (f_{mn})_{m,n=1}^M$ on $\mathcal{Z}$. There is also a finite reserve grid

$$\mathcal{R} = \{R^1, \dots, R^L\} \subset \mathbb{R}_+, \qquad 0 = R^1 < \cdots < R^L = \bar{R},$$

with $Price^{\min} := \min_{1 \le m \le M} Price^m$, $FeeMax^{\max} := \max_{1 \le m \le M} FeeMax^m$ and

$$\bar{R} \ge \max\left\{R_0, \frac{\theta \cdot (FeeMax^{\max})^2}{4Sec \cdot \rho \cdot Price^{\min}}\right\}.$$

At state $x = (R, z) = (R, Price, FeeMax) \in \mathcal{R} \times \mathcal{Z}$, each validator chooses an action in the compact interval

$$\mathcal{A}(x) := \left[0, \frac{FeeMax}{NSec}\right].$$

The next-period reserve is computed by first determining the intermediate value

$$\widehat{R}'(x, a) := (1 - \rho) \cdot R + \theta \cdot \frac{(FeeMax - Sec \cdot s) \cdot s}{Price}, \qquad s := \sum_{i=1}^{N} a_i,$$

and subsequently projecting this value onto the discrete reserve grid via

$$\Pi_{\mathcal{R}}(r) := \max\{R^\ell \in \mathcal{R} : R^\ell \le r\}.$$

Accordingly, the next-period reserve is given by

$$R'(x, a) := \Pi_{\mathcal{R}}(\widehat{R}'(x, a)).$$

▶ Remark 20. Assumption 19 is a computational extension of the baseline model. The exogenous Markov chain is a finite approximation of the original price-demand process, and the reserve grid is the standard discretization used in dynamic policy computation. The bound on $\bar{R}$ guarantees that the reserve state is invariant: because $(FeeMax - Sec \cdot s) \cdot s \le FeeMax^2/(4Sec)$ and $Price \ge Price^{\min}$,

$$\widehat{R}'(x, a) \le (1 - \rho) \cdot \bar{R} + \frac{\theta \cdot (FeeMax^{\max})^2}{4Sec \cdot Price^{\min}} \le \bar{R}.$$

▶ **Definition 21** (Markov strategy and Markov perfect equilibrium). *For each date $t \in \{0, \dots, T\}$ and each validator i, a (mixed)* Markov strategy *is a mapping*

$$\sigma_t^i : \mathcal{R} \times \mathcal{Z} \to \Delta(\mathcal{A}(x)),$$

where $\Delta(\mathcal{A}(x))$ denotes the set of Borel probability measures on $\mathcal{A}(x)$.

Given a strategy profile $\sigma = (\sigma_t^i)_{i,t}$, define continuation values recursively by

$$V_{i,T+1}^{\sigma}(x) := 0,$$

and, for $t = T, T-1, \dots, 0$,

$$V_{i,t}^{\sigma}(x) = \mathbb{E}_{a_1 \sim \sigma_t^1(\cdot|x),\, \dots,\, a_N \sim \sigma_t^N(\cdot|x)} \left[u_i(a; x) + \delta \sum_{z' \in Z} P(z, z')\, V_{i,t+1}^{\sigma}(R'(x, a), z') \right]. \tag{22}$$

A strategy profile σ is a Markov perfect equilibrium (in mixed strategies) *if for every date t, every state x, every validator i, and every alternative mixed action $\mu \in \Delta(\mathcal{A}(x))$,*

$$\begin{aligned} &\mathbb{E}_{a_1 \sim \sigma_t^1(\cdot|x),\, \dots,\, a_N \sim \sigma_t^N(\cdot|x)} \left[u_i(a; x) + \delta \sum_{z' \in Z} P(z, z')\, V_{i,t+1}^{\sigma}(R'(x, a), z') \right] \\ &\qquad \geq \mathbb{E}_{\substack{a_i \sim \mu \\ a_j \sim \sigma_t^j(\cdot|x),\, j \neq i}} \left[u_i(a; x) + \delta \sum_{z' \in Z} P(z, z')\, V_{i,t+1}^{\sigma}(R'(x, a), z') \right]. \end{aligned} \tag{23}$$

The next theorem shows that once the state space is discretized and the horizon is finite, forward-looking validator behavior can still be analyzed with a well-defined equilibrium concept.

▶ **Theorem 22** (Existence of finite-horizon Markov perfect equilibrium). *Under Assumption 19, the finite-horizon dynamic validator game admits a Markov perfect equilibrium in mixed strategies.*

The proof of the above theorem is a backward-induction argument: each date-t problem becomes an ordinary continuation game once later continuation values are fixed.

The next result says that dynamic multiplicity does not arise if every continuation game is already pinned down locally. Once each state-date problem has a unique symmetric pure best-response fixed point, backward induction propagates that uniqueness through the full dynamic game.

▶ **Proposition 23** (Purity and uniqueness under state-by-state uniqueness). *Define the date-t continuation-game payoff by*

$$g_{i,t}(x, a) := u_i(a; x) + \delta \sum_{z' \in \mathcal{Z}} P(z, z') V_{i,t+1}^{\sigma}\big((R'(x, a), z')\big). \tag{24}$$

Suppose that, for every date $t \in \{0, \dots, T\}$ and every state $x \in \mathcal{R} \times \mathcal{Z}$, the continuation game with payoff Equation (24) admits a unique symmetric pure Nash equilibrium action $a_t^*(x) \in \mathcal{A}(x)$. Then the dynamic validator game admits a unique symmetric pure Markov perfect equilibrium, namely the profile that assigns action $a_t^*(x)$ to every validator at every state-date pair.

▶ **Example 24** (Illustrating Proposition 23). We construct a simple finite-horizon example in which the hypothesis of Proposition 23 is satisfied. Consider the finite-horizon Markov game with

$$N = 2, \qquad T < \infty, \qquad \delta \in (0, 1),$$

and let the exogenous state space Z and transition matrix P be arbitrary but finite, as in Assumption 19. Let the reserve grid $\mathcal{R}$ also be arbitrary and finite. Now choose $\theta = 0$. Under this choice, the reserve transition becomes

$$R'(x, a) = (1 - \rho)R,$$

so the next-period reserve no longer depends on the current action profile a. Hence, for every date t and state $x = (R, z)$, the continuation-game payoff

$$g_{i,t}(x, a) = u_i(a; x) + \delta \sum_{z' \in Z} P(z, z')\, V^{\sigma}_{i,t+1}(R'(x, a), z')$$

can be written as

$$g_{i,t}(x, a) = u_i(a; x) + C_t(x),$$

where $C_t(x)$ is a constant with respect to the action profile a. Therefore, the continuation game at any state-date pair (t, x) has exactly the same best responses as the baseline stage game. By Theorem 3, for every state $x = (R, Price, FeeMax)$, the baseline stage game admits a unique symmetric Nash equilibrium, with aggregate security

$$s^*(x) = \min\left\{ \frac{(1-\theta)FeeMax + \sqrt{(1-\theta)^2 FeeMax^2 + 4A_N \cdot B_N \cdot Price \cdot R}}{2A_N}, \frac{FeeMax}{Sec} \right\},$$

and corresponding individual action $a^*(x) = \frac{s^*(x)}{N}$. Since here $\theta = 0$, each continuation game therefore admits the same unique symmetric pure Nash equilibrium action $a^*_t(x) = a^*(x)$ for every $t \in \{0, \dots, T\}$, $x \in \mathcal{R} \times Z$. Thus, the hypothesis of Proposition 23 is satisfied, and it follows that the dynamic validator game admits a unique symmetric pure Markov perfect equilibrium.

5.2 Continuous-State First-Order Condition

The finite-state extension above gives an existence theorem. To understand economically how forward-looking incentives modify the baseline equilibrium, it is useful to return to the original continuous model and derive the equilibrium condition that must hold whenever a pure differentiable Markov equilibrium exists.

The next result shows exactly how forward-looking incentives perturb the myopic equilibrium.

▶ **Proposition 25** (Dynamic symmetric first-order condition). *Fix a horizon $T \geq 1$ and suppose that, in the original continuous model, there exists a pure symmetric Markov perfect equilibrium with continuation value functions*

$$V_{t+1}(R, Price, FeeMax), \qquad t = 0, \dots, T-1,$$

that are continuously differentiable in R. Fix a date $t < T$ and a state $x = (R, Price, FeeMax)$. If the equilibrium at that state is interior, with aggregate security $s^{\mathrm{M}}_t(x) \in (0, FeeMax/Sec)$, then it satisfies

$$(1-\theta)FeeMax - A_N \cdot s + \frac{B_N \cdot Price \cdot R}{s} + \frac{\delta \cdot \theta}{Price}\left(FeeMax - \frac{N+1}{N} Sec \cdot s\right) M_{t+1}(x; s) = 0, \tag{25}$$

where $s = s_t^{\mathrm{M}}(x)$ *and*

$$M_{t+1}(x;s) := \mathbb{E}\left[\partial_R V_{t+1}\left((1-\rho)R + \theta\frac{(FeeMax - Sec\cdot s)s}{Price}, Price_{t+1}, FeeMax_{t+1}\right) \,\middle|\, \substack{Price_t = Price,\\ FeeMax_t = FeeMax}\right]. \tag{26}$$

▶ Remark 26. The first three terms in Equation (25) are exactly the myopic first-order condition from the baseline model. The final term is new. It is the marginal value of the effect of current participation on the next reserve stock. If $M_{t+1}(R, Price, FeeMax; s) \geq 0$, then forward-looking validators value reserve accumulation. The sign of the whole term is then governed by

$$FeeMax - \frac{N+1}{N} Sec \cdot s,$$

which is the marginal effect of one validator's additional participation on next period's reserve inflow. When aggregate participation is below the point at which extra participation sharply compresses fees, dynamic incentives push equilibrium security upward relative to the myopic benchmark. When the system is already close to maximum congestion, the intertemporal effect becomes weaker and can eventually reverse.

The next proposition shows that the runway logic survives the Markov extension.

▶ **Proposition 27** (Runway under a pure Markov equilibrium)**.** *Suppose the pure symmetric Markov perfect equilibrium from Proposition 25 exists and denote its aggregate security policy at date* t *by*

$$s_t^{\mathrm{M}}(R, Price, FeeMax).$$

Define the date-t security set

$$\mathcal{S}_t(\underline{s}) := \{(R, Price, FeeMax) \in \mathbb{R}_+ \times \mathbb{R}_{++}^2 : s_t^{\mathrm{M}}(R, Price, FeeMax) \geq \underline{s}\}.$$

Then, along any equilibrium sample path, security is maintained through date T *if and only if*

$$(R_t, Price_t, FeeMax_t) \in \mathcal{S}_t(\underline{s}) \qquad \text{for all } t = 0, \ldots, T.$$

If, in addition, $s_t^{\mathrm{M}}(R, Price, FeeMax)$ *is weakly increasing in* R *for each fixed* $(Price, FeeMax)$*, then the dynamic reserve threshold*

$$SecureR_t^{\mathrm{M}}(Price, FeeMax; \underline{s}) := \inf\{R \geq 0 : s_t^{\mathrm{M}}(R, Price, FeeMax) \geq \underline{s}\}$$

is well defined, and the runway criterion can be written as

$$R_t \geq SecureR_t^{\mathrm{M}}(Price_t, FeeMax_t; \underline{s}) \qquad \text{for all } t = 0, \ldots, T.$$

The threshold $SecureR_t^{\mathrm{M}}(\cdot)$ need no longer have a closed form, but once equilibrium participation is monotone in reserves, the safe region is still summarized by a reserve cutoff at each state and date.

6 Failure Probabilities and Expected Hand-Off Times

The main results identify the safe region state by state and path by path without committing to a particular stochastic law for price and demand. For quantitative policy analysis, however, one often wants explicit probabilistic outputs such as failure probabilities over a fixed horizon or expected time to fee-only operation. This section derives such metrics under a parametric specialization.

▶ **Assumption 28** (Lognormal state dynamics). *There exist constants $\mu_{Price}, \mu_{FeeMax} \in \mathbb{R}$ and $\sigma_{Price}, \sigma_{FeeMax} > 0$, a standard Brownian motion $(W_t)_{t\geq 0}$, and an i.i.d. sequence*

$$(\Delta W_t, \eta_t)_{t\geq 1}$$

of centered bivariate normal random vectors with unit marginal variances, where $\Delta W_t := W_t - W_{t-1}$ for $t \geq 1$, such that, for every $t \geq 0$,

$$\log Price_t = \log Price_0 + \left(\mu_{Price} - \frac{\sigma_{Price}^2}{2}\right) t + \sigma_{Price} \cdot W_t, \tag{27}$$

$$\log FeeMax_t = \log FeeMax_0 + \mu_{FeeMax} t + \sigma_{FeeMax} \sum_{k=1}^{t} \eta_k. \tag{28}$$

We write Φ for the standard normal cumulative distribution function (cdf).

▶ Remark 29 (geometric Brownian motion). Assumption 28 is imposed only in this section. It strengthens Assumption 1 in order to convert the law-free runway criteria into explicit probabilistic metrics. The price process is a geometric Brownian motion observed at integer dates, while demand remains in a discrete-time lognormal specification. Correlation between token-price shocks and demand shocks is allowed through the joint law of $(\Delta W_t, \eta_t)$; the results below use only the normal marginal distributions and a union-bound argument.

Next, we convert the law-free stress-test criterion from Theorem 15 into an explicit statement.

▶ **Theorem 30** (Finite-horizon failure probability under lognormal dynamics). *Fix a horizon $T \in \mathbb{N}$ and confidence parameters $z_{Price}, z_{FeeMax} \geq 0$. Define deterministic lower envelopes by*

$$\underline{Price}_t(z_{Price}) := Price_0 \cdot \exp\left(\left(\mu_{Price} - \frac{\sigma_{Price}^2}{2}\right) t - \sigma_{Price} \cdot \sqrt{t} \cdot z_{Price}\right), \tag{29}$$

$$\underline{FeeMax}_t(z_{FeeMax}) := FeeMax_0 \cdot \exp\left(\mu_{FeeMax} \cdot t - \sigma_{FeeMax} \cdot \sqrt{t} \cdot z_{FeeMax}\right), \tag{30}$$

for $t = 0, \ldots, T$. If

$$(1-\rho)^t R_0 \geq SecureR(\underline{Price}_t(z_{Price}), \underline{FeeMax}_t(z_{FeeMax}); \underline{s}) \qquad \text{for all } t = 0, \ldots, T, \tag{31}$$

then

$$\mathbb{P}(\tau_{\text{fail}} > T) \geq 1 - T \cdot \big(\Phi(-z_{Price}) + \Phi(-z_{FeeMax})\big). \tag{32}$$

In particular, if

$$z_{Price} = \Phi^{-1}\left(1 - \frac{\alpha_{Price}}{T}\right), \qquad z_{FeeMax} = \Phi^{-1}\left(1 - \frac{\alpha_{FeeMax}}{T}\right)$$

for some $\alpha_{Price}, \alpha_{FeeMax} \in (0,1)$, then

$$\mathbb{P}(\tau_{\text{fail}} > T) \geq 1 - \alpha_{Price} - \alpha_{FeeMax}.$$

▶ **Proposition 31** (Hand-off probabilities and expected hand-off time). *Suppose Assumption 28 holds and define the fee-only demand cutoff*

$$FeeMax^{\mathrm{FO}} := \frac{A_N \cdot \underline{s}}{1-\theta}. \tag{33}$$

Let

$$y_0 := \log FeeMax_0, \qquad y^{\mathrm{FO}} := \log FeeMax^{\mathrm{FO}}.$$

Assume $\mu_{FeeMax} > 0$ *and* $FeeMax_0 < FeeMax^{\mathrm{FO}}$. *Then:*

(i) *The hand-off time satisfies*

$$\tau_{\mathrm{hand}} = \inf\{t \geq 0 : FeeMax_t \geq FeeMax^{\mathrm{FO}}\} < \infty \qquad \textit{almost surely.}$$

(ii) *For every horizon* $T \in \mathbb{N}$,

$$\mathbb{P}(\tau_{\mathrm{hand}} \leq T) \geq 1 - \Phi\left(\frac{y^{\mathrm{FO}} - y_0 - \mu_{FeeMax} T}{\sigma_{FeeMax}\sqrt{T}}\right). \tag{34}$$

(iii) *The expected hand-off time is finite and satisfies*

$$\frac{y^{\mathrm{FO}} - y_0}{\mu_{FeeMax}} \leq \mathbb{E}[\tau_{\mathrm{hand}}] \leq 1 + \sum_{t=1}^{\infty} \Phi\left(\frac{y^{\mathrm{FO}} - y_0 - \mu_{FeeMax} t}{\sigma_{FeeMax}\sqrt{t}}\right) < \infty. \tag{35}$$

(iv) *If, in addition, the condition of Theorem 30 holds for the same horizon* T, *then*

$$\mathbb{P}(\tau_{\mathrm{hand}} < \tau_{\mathrm{fail}}) \geq 1 - T\big(\Phi(-z_{Price}) + \Phi(-z_{FeeMax})\big) - \Phi\left(\frac{y^{\mathrm{FO}} - y_0 - \mu_{FeeMax} T}{\sigma_{FeeMax}\sqrt{T}}\right). \tag{36}$$

Proposition 31 separates two distinct quantitative questions. The first is how quickly demand alone is likely to carry the protocol into the fee-only region. The second is whether that arrival is likely to happen before reserve failure. Positive demand drift makes eventual hand-off almost sure in this parametric environment, while the finite-horizon bounds show how drift and volatility translate into operational success probabilities.

The analytical results above are complemented by two numerical illustrations. Figure 2 plots three deterministic state paths built from the parametric law: one path fails before hand-off, one reaches fee-only sustainability gradually, and one hands off quickly. Figure 3 then shows how the quantitative runway picture changes with demand drift in the parametric environment. The left panel reports Monte Carlo estimates of survival, hand-off, and successful-transition probabilities at a fixed horizon, while the right panel plots the analytical lower and upper bounds on expected hand-off time from Proposition 31.

7 Conclusion and Future Work

Proof-of-stake systems with finite reserves face a reserve hand-off problem. Early validator rewards may rely on reserve subsidies, while long-run security must eventually be supported by transaction fees. This paper formalized this transition through a stochastic model with token-price risk, demand risk, and strategic validator participation. We solved the symmetric validator participation game and derived a closed-form equilibrium. Using this equilibrium, we obtained an explicit state-dependent reserve threshold. For each token price and demand

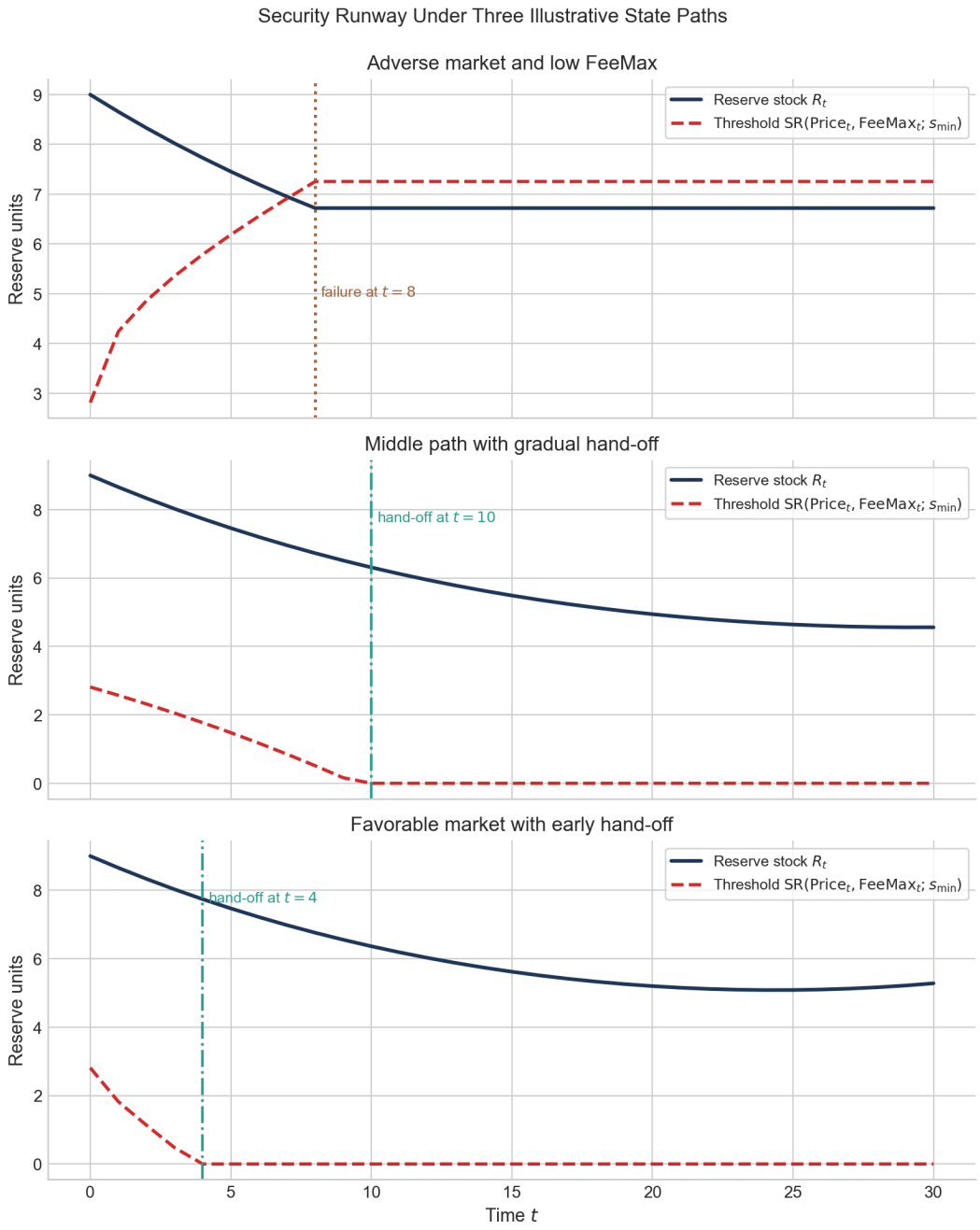


Figure 2 Illustrative runway paths under three deterministic state trajectories generated from the parametric law. In each panel, the solid blue curve is the reserve stock R_t and the dashed red curve is the state-dependent threshold $SecureR(Price_t, FeeMax_t; \underline{s})$. The vertical marker identifies the first economically relevant event along the path: hand-off when the threshold reaches zero, or failure when the reserve falls below the threshold. The adverse path crosses the failure boundary before reaching the fee-only region; the middle path reaches fee-only sustainability only after a prolonged transition; and the favorable path hands off early. The figure uses the same protocol parameters as Figure 1, with initial reserve $R_0 = 9$.

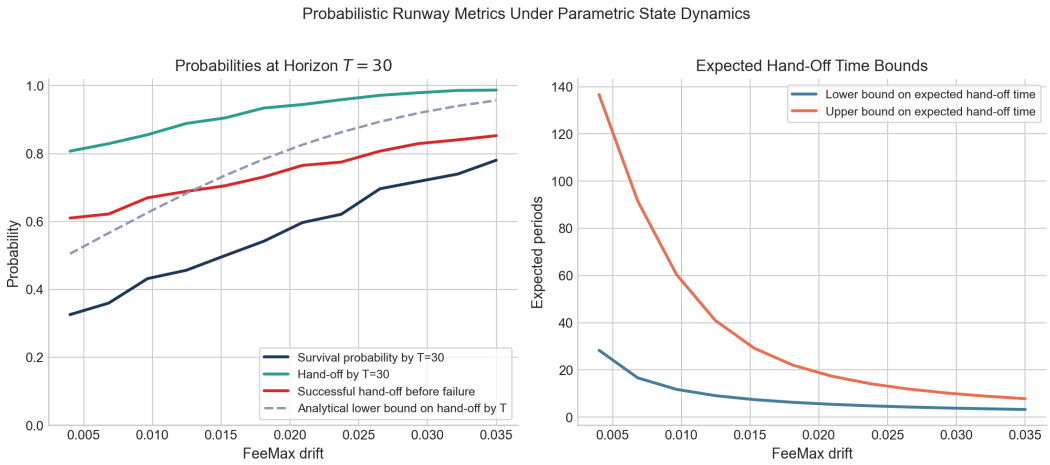


Figure 3 Probabilistic runway metrics under the parametric state dynamics. The horizontal axis in both panels is the demand drift μ_{FeeMax}. In the left panel, the solid curves are Monte Carlo estimates of survival probability by horizon $T = 30$, hand-off probability by horizon $T = 30$, and the probability of successful hand-off before failure. The dashed curve is the analytical lower bound on hand-off by time T from Proposition 31. In the right panel, the two curves are the analytical lower and upper bounds on expected hand-off time from Proposition 31. The figure uses the same protocol parameters as Figure 1, with initial reserve $R_0 = 9$, $Price_0 = 1$, $FeeMax_0 = 1$, price drift $\mu_{Price} = 0.005$, price volatility $\sigma_{Price} = 0.16$, and demand volatility $\sigma_{FeeMax} = 0.10$. Higher demand drift increases survival and hand-off probabilities and narrows the expected time to hand-off.

state, the threshold gives the minimal reserve stock necessary and sufficient to sustain a target security level. This threshold separates states in which security is infeasible, states in which security depends on the reserve, and states in which fees alone are sufficient. We then used the threshold to study the dynamic hand-off problem. Security is maintained along a path exactly while the reserve remains above the state-dependent threshold. Failure occurs when the reserve first falls below this threshold. A successful hand-off occurs when the system reaches the fee-only region before failure. This converts reserve adequacy into a state-dependent hitting-time problem. We also studied forward-looking validator incentives. In the Markov extension, current participation affects both current rewards and the next reserve stock, and therefore changes future reward opportunities. We proved finite-horizon Markov perfect equilibrium existence and derived the dynamic first-order condition that captures this continuation-value effect. Finally, under lognormal price-demand dynamics, we translated the runway analysis into finite-horizon failure-probability bounds, hand-off probabilities, and bounds on expected hand-off time.

The main conclusion is that reserves are not neutral accounting devices. Their security value depends on the current state of the system. A protocol with a large nominal reserve can still be close to failure after an adverse price or demand shock. Conversely, a protocol with a smaller reserve can be safe once demand is strong enough to support the target security level through fees alone. Reserve policy should therefore be evaluated through state-dependent runway analysis rather than through nominal depletion dates or steady-state reward ratios alone.

The model made several simplifying assumptions. Token price and demand were treated as exogenous, so the analysis did not model token valuation or user adoption. Validators were symmetric in the baseline model, so the threshold should be interpreted as a benchmark

rather than as a full description of heterogeneous operators. The finite-horizon stress test was deliberately conservative because it lower-bounded the reserve path by ignoring future fee inflows. These assumptions made it possible to isolate the reserve hand-off mechanism and obtain explicit threshold conditions.

A natural next step is adaptive reserve policy. In this paper, the policy parameters (θ, ρ) were fixed. This provided a clean benchmark for analyzing the security-runway mechanism. In principle, a protocol may want payout and fee-retention rules to respond to the current reserve, token price, and demand state. For example, the protocol could reduce reserve payouts after adverse price realizations, increase fee retention when the reserve is low, or change its policy once demand approaches the fee-only region.

Such an extension would turn the analysis from policy evaluation into policy design. Governance would choose a state-dependent rule of the form

$$(\theta_t, \rho_t) = \phi(R_t, Price_t, FeeMax_t)$$

to extend runway, reduce failure risk, or reach fee-only sustainability subject to incentive and credibility constraints. Developing such a theory would require combining the equilibrium framework in this paper with tools from dynamic mechanism design or stochastic control. We view this as an important direction for future work.

References

1 Algorand. The Algo, 2025. Accessed: 2nd June 2026. URL: `https://algorand.co/technology/the-algo`.

2 All About XRP. XRP Supply & Inflation Explained, 2025. Accessed: 27th May 2026. URL: `https://allaboutxrp.com/learn/xrp-supply-explained`.

3 Avalanche. Avax token, 2026. Accessed: 27th May 2026. URL: `https://www.avax.network/about/tokens`.

4 Cardano Docs. Cardano monetary policy. `https://docs.cardano.org/about-cardano/explore-more/monetary-policy/`. Accessed March 9, 2026.

5 Miles Carlsten, Harry Kalodner, S. Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 154–167, New York, NY, USA, 2016. Association for Computing Machinery. `doi:10.1145/2976749.2978408`.

6 Jonathan Chiu and Thorsten V. Koeppl. The economics of cryptocurrency: Bitcoin and beyond. *Canadian Journal of Economics/Revue canadienne d'économique*, 55(4):1762–1798, 2022. `doi:10.1111/caje.12625`.

7 Lin William Cong, Ye Li, and Neng Wang. Tokenomics: Dynamic adoption and valuation. *The Review of Financial Studies*, 34(3):1105–1155, 2021.

8 Lin William Cong, Ye Li, and Neng Wang. Token-based platform finance. *Journal of Financial Economics*, 144(3):972–991, 2022.

9 Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake protocol. Cryptology ePrint Archive, Paper 2017/573, 2017. URL: `https://eprint.iacr.org/2017/573`.

10 Giulia Fanti, Leonid Kogan, Sewoong Oh, Kathleen Ruan, Pramod Viswanath, and Gerui Wang. Compounding of wealth in proof-of-stake cryptocurrencies. In *Financial Cryptography and Data Security*, 2019.

11 Hedera Council. Treasury management, 2025. Accessed: 27th May 2026. URL: `https://hederacouncil.org/treasury`.

12 Nicolas Houy. The economics of Bitcoin transaction fees. Working Paper 1407, GATE Lyon Saint-Étienne, 2014. `doi:10.2139/ssrn.2400519`.

13 Gur Huberman, Jacob D. Leshno, and Ciamac C. Moallemi. Monopoly without a monopolist: An economic analysis of the bitcoin payment system. *Review of Economic Studies*, 88(6):3011–3040, 2021. doi:10.1093/restud/rdab014.

14 Samuel Häfner. Optimal decentralization and service provision on a blockchain platform with market frictions, September 2023. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3954773.

15 Felix Irresberger and Ruomei Yang. Coin concentration of proof-of-stake blockchains. *Economics Letters*, 229:111219, 2023. doi:10.1016/j.econlet.2023.111219.

16 Aggelos Kiayias, Philip Lazos, and Paolo Penna. Single-token vs two-token blockchain tokenomics. In Zeta Avarikioti and Nicolas Christin, editors, *7th Conference on Advances in Financial Technologies (AFT)*, volume 354 of *LIPIcs*, pages 22:1–22:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.AFT.2025.22.

17 Aggelos Kiayias, Philip Lazos, and Jan Christoph Schlegel. Would friedman burn your tokens? In *28th International Conference on Financial Cryptography and Data Security (FC)*, pages 231–247, Berlin, Heidelberg, 2024. Springer-Verlag. doi:10.1007/978-3-031-78676-1_13.

18 Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. Cryptology ePrint Archive, Paper 2016/889, 2016. URL: https://eprint.iacr.org/2016/889.

19 Kevin Liao and Jonathan Katz. Incentivizing blockchain forks via whale transactions. In Michael Brenner, Kurt Rohloff, Joseph Bonneau, Andrew Miller, Peter Y.A. Ryan, Vanessa Teague, Andrea Bracciali, Massimiliano Sala, Federico Pintore, and Markus Jakobsson, editors, *Financial Cryptography and Data Security*, pages 264–279, Cham, 2017. Springer International Publishing. doi:10.1007/978-3-319-70278-0_17.

20 Shashank Motepalli and Hans-Arno Jacobsen. Decentralization in PoS Blockchain Consensus: Quantification and Advancement. *arXiv preprint arXiv:2504.14351*, 2025. doi:10.48550/arXiv.2504.14351.

21 Christina Ovezik, Dimitris Karakostas, Mary Milad, Daniel W. Woods, and Aggelos Kiayias. Sok: Measuring blockchain decentralization. In Marc Fischlin and Veelasha Moonsamy, editors, *23rd International Conference on Applied Cryptography and Network Security (ACNS)*, Lecture Notes in Computer Science, pages 184–214. Springer, 2025. doi:10.1007/978-3-031-95761-1_7.

22 Paolo Penna and Manvir Schneider. Reserve depletion and security runway in proof-of-stake systems. *arXiv preprint arXiv:2606.03587*, 2026. doi:10.48550/arXiv.2606.03587.

23 Fahad Saleh. Blockchain without waste: Proof-of-stake. *Review of Financial Studies*, 34(3): 1156–1190, 2021. doi:10.1093/rfs/hhaa075.

24 Itay Tsabary and Ittay Eyal. The gap game. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 713–728, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3243734.3243737.

25 Zhilin Wang, Qin Hu, Yawei Wang, and Yinhao Xiao. Transaction pricing mechanism design and assessment for blockchain. *High-Confidence Computing*, 2(1):100044, 2022. doi:10.1016/j.hcc.2021.100044.

The Cost of Secure Restaking vs. Proof-Of-Stake

Akaki Mamageishvili
Offchain, Zurich, Switzerland

Benny Sudakov
ETH Zurich, Switzerland

Abstract

We compare the total capital efficiency of secure restaking and Proof-of-Stake (PoS) protocols. First, we consider the sufficient condition by EigenLayer project for the restaking graph to be secure, that can be checked in a polynomial time in the restaking graph size. The condition implies that it is always possible to transform such a restaking graph into separate secure PoS protocols. We derive two main results: upper and lower bounds on the required extra stakes to add to the validators of the secure restaking graph to be able to transform it into secure PoS protocols. In particular, we show that the restaking savings compared to PoS protocols can be very large and can asymptotically grow as a square root of the number of validators. We also study a complementary question of aggregating secure PoS protocols into a secure restaking graph and provide matching lower and upper bounds on the PoS savings.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Proof-of-Stake, Restaking, Efficiency

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.7

Related Version *Previous Version*: `https://arxiv.org/abs/2505.24440`

Funding *Benny Sudakov*: Research of B. Sudakov supported in part by SNSF grant 200021-228014.

Acknowledgements We want to thank anonymous reviewers for improving the paper.

1 Introduction

The cryptoeconomic security of blockchains has evolved from energy-intensive Proof-of-Work to more efficient Proof-of-Stake (PoS), and most recently to restaking protocols [1]. Restaking allows validators to reuse their staked capital across multiple services or projects, creating a bipartite graph between validators and services. This reuse promises significant capital efficiency gains over running independent PoS protocols for each service, but it also introduces complex interdependencies and potential security risks.

It can be argued that restaking gives a boost to newer projects in attracting capital for their *cryptoeconomic security*. Cryptoeconomic security refers to a security derived from staked parties being slashed their stakes in case of misbehavior. Restaking seems to be attractive for the validators of the original protocol as well. In particular, restaking allows them to reuse idle computational power and storage allocation for additional rewards from other projects. The original protocol, for which the validators are staked, may have requirements that do not exhaust all the resources of the validators. Setting low requirements is usually done for the purpose of decentralization[2]. Another potential reason for such availability might be economies of scale for establishing further validation services.

[1] Total value locked in all restaking protocols amounts to more than $\$20B$ – `https://defillama.com/protocols/restaking`, accessed on January, 2026.

[2] Check out Ethereum network plans to resolve this asymmetry between validator resources using rainbow staking `https://ethresear.ch/t/unbundling-staking-towards-rainbow-staking/18683`.



© Akaki Mamageishvili and Benny Sudakov;
licensed under Creative Commons License CC-BY 4.0



The simplest way to verify the security of the restaking uses a trivial cost-benefit analysis. In this analysis, the total amount of lost stakes of attackers is compared to the total value obtained from the attack of a set of validators [3]. The same cost-benefit analysis can be used to check the security of any PoS protocol, in which a single project/service is secured by multiple validators. In fact, the cost-benefit analysis for restaking is a direct generalization of the cost-benefit analysis for PoS protocols. This allows a comparison of restaking and multiple PoS protocols, which we develop in this paper.

Conceptually, the PoS protocol is simpler and a more natural concept than restaking. Hence, we are interested in checking the comparative benefits of restaking protocols. One particular attractive point of restaking protocols can be the lower total staking requirements than that of PoS protocols, since there are opportunity costs associated with locking up stake for security reasons. A key open question is whether restaking truly reduces total staking requirements compared to equivalent secure PoS protocols, and if so, by how much. Answering the latter question allows us to close the gap not addressed in the prior restaking literature.

Most PoS protocols pay for the opportunity cost of locking up a capital by rewarding stakers with newly issued tokens, which leads to inflation. Hence, minimizing the total stake amount while maintaining security can be a reasonable goal in both restaking or PoS protocols. Starting with a secure restaking protocol and initial stake endowments for all validators, we measure how much extra stake would be required if validators decided to form separate Proof-of-Stake protocols with their stake amounts, dividing them across the same set of projects they secure in the original restaking graph.

In the following, we describe an example that gives intuition of the secure restaking graph and its transformation to secure PoS protocols, see the Figure 2.

▶ **Example 1.** There are two services, with valuations 1 and 3 and security parameters α. The security parameter for the first service, α_1 is any number in the interval $(0, 1)$ and $\alpha_2 = \frac{1}{2}$. There are two validators, with stakes 2 and 2. The first validator secures both services. The second validator secures only the second service.

When a share of stake of a subset of validators is larger than the security parameter of the service, then such set can successfully attack this service. Let us first check that the restaking graph is secure. It is easy to see the following claims hold:

- the first validator alone can attack only the first service, since it is the only validator securing it. Such attack is not profitable;
- the second validator alone can not attack any service: it is not securing the first service, while the second service has large enough security parameter. The share of the second validator in the total stake securing the second service is $\frac{1}{2}$, which is not higher than $\alpha_2 = \frac{1}{2}$;
- finally, both validators can attack both services but they lose the same amount of stake as they gain the value.

Next, we check if the validators can distribute their stakes so that both services are secured in the corresponding PoS protocols. First, note that it is the first validator that divides its stake between two services. The first service needs at least the stake amount 1 from the first validator. After the first validator allocates stake amount 1 to the first service,

[3] [4] discusses ways to estimate the value gained from attacking a protocol and cryptoeconomic security in more detail.

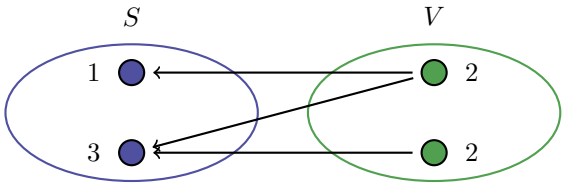


Figure 1 Example of a secure restaking graph.

the second service is not secure in the PoS protocol with stakes 1 and 2, from the first and second validator, respectively. In fact, it can be easily seen that either the first validator needs to have at least a matching stake amount 2, or the second validator needs to have at least a stake 3. In the first case, none of them can attack the service as their stake share is not larger than the service security parameter. In the second case, the second validator can attack the service, but by doing so it does not make more value than it loses. In either case, at least an extra stake 1 is required to secure both services in their respective PoS protocols.

We only allow all validator stakes to (weakly) increase. This captures a real-life consideration in which validators' stakes can not be taken away, but their stakes can be increased through financing when needed. In a sense, we look into over-collateralization of the whole network needed to allow separate secure PoS protocols. Similarly, [5] investigates the over-collateralization of individual validators (instead of the full set of validators as we do in this paper) that allows robustness of the restaking network security. Most of our results can be extended to the model with individual over-collaterization, obtaining the same bounds.

If both increasing and decreasing of validator stakes were allowed, it would be trivial to construct a restaking graph with minimum total stake. This amount is equal to the sum of all project values, and secure PoS protocols for all projects can be constructed by dividing the stakes of the validators. See Proposition 5 and discussion before it.

Our Contributions

1. We check the sufficient condition for a secure restaking graph, specified by the EigenLayer project [10]. This sufficient condition is the only currently available (generic) condition for secure restaking graph, that can be checked in polynomial time in the size of the restaking graph. Restaking graphs that satisfy the sufficient condition do not save on the total required stake. In particular, it is possible to divide the initial stakes of the validators across projects they validate, to obtain secure PoS protocols. This result motivates us to ask a question studied next. The class of restaking graphs satisfying the sufficient condition serves as a boundary point in the extended analysis.
2. We define *restaking savings* of a fixed secure restaking graph. It is equal to the extra stake required, relative to the original total available stake in the graph, to be able to obtain secure PoS protocols through the division of stake described above.
3. To upper bound restaking savings, we use simple secure PoS constructions from the original restaking graph. In particular, we upper bound the value of restaking savings with the highest degree among the projects. Another upper bound is the minimum incidence of any validator in the cover of a validator set by project neighborhoods. The third upper bound is equal to second largest multiplicative inverse of the security parameter, which is a small constant value in many practical cases. The last upper bound is equal to the square root of the number of validators. Next, we give a construction to lower bound this value. The lower bound example asymptotically matches all upper bounds obtained.

4. We study a complementary question of aggregating given secure PoS protocols into a secure restaking graph. When aggregating, each validator adds stakes across projects that they secure in separate PoS protocols, resulting in a restaking graph. Mathematically, this is an exact inverse to dividing stakes in the restaking graph into separate PoS protocols. Similarly to the restaking savings, we define *PoS savings* as the total additional stake required to add validators in the aggregate restaking graph to make it secure. We first construct an example showing that the resulting restaking graph is not secure. In fact, at least an additional stake equal to the original total stake times the number of projects/validators minus one is required to make it secure, thus showing a lower bound of number of projects/validators minus one on PoS savings. We also provide an upper bound on PoS savings, which exactly matches a lower bound.

Although mostly conceptual, our results can provide design recommendations. For an existing restaking project that has an underlying graph structure specified by stake sizes and service valuations, we could compute restaking savings. For a given restaking graph, it must be feasible to check if it is secure using heuristic algorithms, instead of checking a polynomial-time sufficient condition which severely limits the domain and may result into zero restaking savings.

Recently, EigenLayer's restaking project upgraded the system that only allows a unique slashing property: any staked asset can be slashed by a single service [4]. This restriction effectively makes it equivalent to validators partition their total stake across services, and hence results into separate secure PoS protocols. As a result, the system no longer realizes capital efficiency gains from true stake reuse, and the restaking savings collapses to zero under this constraint. Restaking savings were effectively equal to zero even before this upgrade, as long as the sufficient condition was imposed on the restaking graph.

Related Work

[8] introduced the first cryptocurrency Bitcoin and the concept of PoW, in which the random sampling of the next block proposer is done proportionally to a hashrate of a miner, by solving a puzzle that requires just computation and shortcuts for solving it can not be found. [9] provides a detailed discussion and economic comparison on PoW and PoS concepts. [4] discusses ways to estimate the value gained from attacking a protocol and cryptoeconomic security in more detail. The idea of restaking was originally proposed in the whitepaper of the Eigenlayer project [10]. A similar concept exists in traditional finance literature under the name of *rehypothecation* [6]. The interdependencies between protocols and validators introduced by restaking create new risks, which have been modeled and analyzed in the original Eigenlayer whitepaper, as well as in a follow up paper [5], in which the authors study robust restaking networks, resistant to a failure of a fraction of validating stakes and cascading effects. Thus, they extend the underlying model to a richer model in terms of risks. The authors consider over-collaterization factor of individual validators. They investigate out how large should this uniform over-collaterization factor be so that the originally secure restaking graph stays robust under these crashes. Our over-collaterization over the total stake amount can also be reduced to individual over-collaterization factors, as we argue after the Theorem 10.

[2] studies a setting in which the value of attacked set of services is not the sum of values in the set, but a concave function of them. Such model generalizes a stylized model considered in the original EigenLayer whitepaper [10], [5] and our paper. [11] studies elastic

[4] `https://blog.eigencloud.xyz/introducing-the-eigenlayer-security-model/`

restaking services, in which validators allocate stakes to services that may make up more than their initial stake. This allows for a more expressive model than the original restaking model where the whole stake of each validator is locked up for all services. The work also considers a typical distributed computing setting in which only a fraction of validators is Byzantine, while our model is assuming a simple rationality of all validators. [7] studies risks and benefits of combining different restaking services. [1] studies economic security of different protocols as a function of streaming rewards. [3] studies restaking through the perspective of Sybils.

2 Notation and Model

Our notation follows closely [5]. A restaking graph $G = (S, \pi, \alpha, V, \sigma, E)$ consists of the following:

- S denotes the set of m services $\{1, ..., m\}$ (we will refer projects as services from now on throughout the paper),
- $\pi \in \mathbb{R}^m_+$ denotes values of services,
- $\alpha \in \mathbb{R}^m_+$ denotes security parameters of services, where $\alpha_s \in [0, 1]$,
- V denotes the set of n validators $\{1, ..., n\}$,
- $\sigma \in \mathbb{R}^n_+$ denotes stakes of validators,
- The set of edges E connecting S nodes (services) to V nodes (validators).

An edge $(s, v) \in E$ indicates that a service s is secured by a validator v. Hence, a restaking graph is represented as a bipartite graph with two parts made up of services and validators.

Let $N_G(s)$ denote the neighborhood of the service $s \in S$ in the restaking graph G, i.e. $N_G(s) := \{v \in V : (s, v) \in E\}$, is the set of validators staked on, and, therefore, securing a service s. Similarly, $N_G(v)$ denotes the neighborhood of the validator $v \in V$, i.e., $N_G(v) := \{s \in S : (s, v) \in E\}$, is the set of services validator v is staked on.

A service s has a security parameter α_s if only subsets of validators staked on this server that have at least α_s fraction of the total stake of validators staked on s can attack it. More formally:

▶ **Definition 2.** *A subset of validators $W \subseteq V$ can attack the service s if and only if*

$$\frac{\sum_{k \in W \cap N_G(s)} \sigma_k}{\sum_{v \in N_G(s)} \sigma_v} > \alpha_s.$$

For each subset of validators A, we can define the maximal set of services that they can collectively attack. Let this set be denoted by $M(A)$. That is $M(A) = \{s \in S : A \text{ can attack } s\}$. The attack carried out by the validators in A is profitable when $\sum_{s \in M(A)} \pi_s > \sum_{v \in A} \sigma_v$, that is, validators lose less total stake than the total value they obtain from the attack.

▶ **Definition 3.** *A restaking graph G is secure if there exists no subset of validators A that can profitably attack its corresponding set of services $M(A)$.*

We are interested in the following question:

▶ **Question 1.** *Is it possible to divide the stakes of all validators between all services they secure in the graph G so that all services are secure in their corresponding PoS protocols?*

A division of stake σ_v of validator v across services it secures in G induces a vector $c^v \in \mathbb{R}_+^{|N_G(v)|}$, such that, $\sum_{s \in N_G(v)} c_s^v = \sigma_v$. We say that division of stake is secure if in the corresponding Proof-of-Stake protocols, that represent star graphs centered at the service, we have guaranteed security. That is, for each service s with security parameter α_s, a vector of stakes $\{k \in N_G(s) : c_s^k\}$ is secure, which means that there is no profitable attack by validators having stakes $\{k \in N_G(s) : c_s^k\}$. Let $\mathcal{W}(G)$ denote all initial stake vectors σ for which such division is possible. To simplify notation, here we implicitly assume that the stake vector is not part of a restaking graph G, and only keep its combinatorial structure together with the π and α vectors.

We consider stake vectors σ' that (weakly) dominate stake vector σ, that is, $\sigma'_v \geq \sigma_v$ for any $v \in V$. The set of all stake vectors that dominate σ is denoted by $D(\sigma)$. Let $T(\sigma) := \sum_{v \in V} \sigma_v$. Then, the total extra value to reach from the stake vector σ to the stake vector σ' is $T(\sigma') - T(\sigma)$. The rationale behind considering (weakly) dominant vectors of stakes is that we assume that initial stakes cannot be taken away from the validators, but they can be increased if needed.

The Eigenlayer project, [10], specifies a sufficient condition when a restaking graph is secure: for any $v \in V$, the following inequality holds:

$$\sigma_v \geq \sum_{s \in N_G(v)} \frac{\sigma_v}{\sum_{k \in N_G(s)} \sigma_k} \frac{\pi_s}{\alpha_s}. \tag{1}$$

This condition allows to check whether a given restaking graph is secure in polynomial time in the input graph size. In fact, the time is even linear in the number of edges $|E|$ [5].

3 Results

In this section, we first check the sufficient condition (1) and its implications. The implications motivate to define restaking and Proof-of-Stake savings. Last, we obtain lower- and upper-bound results for both types of savings.

3.1 Restaking Savings

If the sufficient condition on validator stakes (1) is satisfied, there is a division of stakes such that all services are secure in their corresponding PoS protocols. More formally, we obtain the following result:

▶ **Proposition 4.** *When the stake vector σ satisfies* (1)*, then $\sigma \in \mathcal{W}(G)$.*

Proof. Consider the following division of stakes: the validator v allocates at least

$$\frac{\sigma_v}{\sum_{k \in N_G(s)} \sigma_k} \frac{\pi_s}{\alpha_s}$$

stake to a service s it secures in G, in the PoS protocol. Such allocation is possible because of (1). Each service s is assigned a total stake amount of at least $\frac{\pi_s}{\alpha_s}$, since

$$\sum_{v \in N_G(s)} \frac{\sigma_v}{\sum_{k \in N_G(s)} \sigma_k} \frac{\pi_s}{\alpha_s} = \frac{\pi_s}{\alpha_s},$$

[5] The general decision problem whether a given restaking graph is secure is difficult, see [5].

by bringing the outer summation operator into the nominator. With $\frac{\pi_s}{\alpha_s}$ total stake amount in PoS protocol for service s, there is no profitable attack, since for the successful attack the attackers should control at least $\frac{\pi_s}{\alpha_s}\alpha_s = \pi_s$ stake, hence, it cannot be profitable. ◀

Note that the value $\frac{\pi_s}{\alpha_s}$ is the minimum required total stake in a PoS protocol for a service s with value π_s and security parameter α_s that guarantees that there is no profitable attack, independent of how this total stake is divided between validators. For any $t < \frac{\pi_s}{\alpha_s}$ and any number of validators, there exists a distribution of the total stake t among the validators such that there is a profitable attack.

On the other hand, if we consider suitable division of stakes between validators, we obtain a much weaker requirement to make secure PoS protocols. Namely, we obtain that there is a minimum total stake requirement to ensure that there is a division in secure PoS protocols.

However, such a distribution does not take into account that validators already have stakes, which cannot be taken away from them.

▶ **Proposition 5.** *The minimum stake amount to guarantee secure PoS protocols is equal to* $\sum_{s\in S}\pi_s$.

Proof. The total stake amount $\sum_{s\in S}\pi_s$ can be distributed arbitrarily among validators, with the only guarantee that each service can get its own value staked by one of the validators securing it. ◀

Another consideration is that the restaking graph with the minimum total stake in $\mathcal{W}(G)$ is not necessarily secure, while we will focus on secure restaking graphs from now on.

We are ready to define the main metric through which we measure capital efficiency of a secure restaking graph, *restaking savings*:

▶ **Definition 6.** *For a given secure restaking graph G, restaking savings, $RS(G)$, denotes a minimum extra total value to add to the stakes in σ vector, in relative terms, so that secure subdivision of the new stake vector is possible. That is,*

$$RS(G) = \min_{\sigma'\in D(\sigma)\cap\mathcal{W}(G)} \frac{T(\sigma') - T(\sigma)}{T(\sigma)}.$$

Let $d_G(u)$ denote the degree of vertex $u \in S \cup V$ in graph G. Note that $d_G(u) = |N_G(u)|$. Consider a subset of services whose neighborhoods cover the entire set of validators V:

$$\mathcal{R} := \{R : R \subseteq S \& \cup_{s\in R} N_G(s) = V\}.$$

For each of these covers, calculate the incidences of all validators in it, and take the largest incidence number. Let K be minimum such number over all elements of $\mathcal{R}$. Formally,

$$K = \min_{R\in\mathcal{R}}\{k | k = \max_{v\in V}\sum_{s\in R}\mathbb{I}(v \in N_G(s))\},$$

where indicator function $\mathbb{I}(P) = 1$ whenever the statement P is true.

We prove several upper bounds on the restaking savings, as a function of different parameters of the graph G. Note that since the original graph is secure, we have the following inequality:

$$T(\sigma) = \sum_{v\in V}\sigma_v \geq \sum_{s\in S}\pi_s = T(\pi).$$

Otherwise, there is a profitable attack that involves all validators V.

▶ **Proposition 7.** *$RS(G)$ is upper bounded by the following values:* $\max_{s\in S}(d_G(s))$ *and* K.

Proof. First, we list three types of secure PoS constructions for a service $s \in S$.

1. the vector of stakes in a corresponding PoS protocol $(c_1, ..., c_{d_G(s)})$ is equal to $(0, ..., 0, \pi_s, 0, ...0)$. This construction is secure for any α_s since the only validator that can attack the service is losing at least the value it is gaining.
2. the vector of stakes $(c_1, ..., c_{d_G(s)})$ is equal to $(0, ..., 0, \pi_s + \sigma_v, 0, ...0)$ for any v. This construction is secure for a similar reason as above.
3. the vector of stakes $(c_1, ..., c_{d_G(s)})$ is equal to $(\sigma_{v_1}, ..., \sigma_{v_{d_G(s)}})$, where $N_G(s) = \{v_1, ..., v_{d_G(s)}\}$. This construction is secure for any α_s since otherwise the original restaking graph G would not be secure: validators that secure the service s have a profitable attack on the service s.

We can add the value π_s to all stakes in $N_G(s)$, i.e., add it $d_G(s)$ times, and use subdivision for each validator v that allocates $\sigma_v + \pi_s$ once to some service s in its neighborhood $N_G(v)$ and π_k to all other corresponding services $k \in N_G(v) \setminus \{s\}$. This way, all services are secure. Also all original stake sizes are used, so that the resulting stake vector is in $D(\sigma)$. At the same time, extra value added is upper bounded by

$$\sum_{s\in S} d_G(s)\pi_s \le \max_{s\in S}(d_G(s)) \sum_{s\in S} \pi_s \le \max_{s\in S}(d_G(s)) \sum_{v\in V} \sigma_v.$$

Next, we show $RS(G) \le K$. Consider a cover of vertices in V by neighborhoods of services $s \in R \subseteq S$ that has the minimum maximum incidence number K. For each neighborhood $N_G(s)$, where $s \in R$, use construction 3: original $(\sigma_{v_1}, ..., \sigma_{v_{d_G(s)}})$ stakes from the restaking graph to secure service s in the PoS protocol. To be able to do that, we need to add at most $K - 1$ times all σ_v stakes, for any $v \in V$. In this way, we utilize all σ_v stakes and secure services in the cover set R. For service s in $S \setminus R$, we add π_s stake once to any of its neighbors in $N_G(s)$ and use construction $(0, ..., 0, \pi_s, 0, ..., 0)$ to secure service s in a corresponding PoS protocol. This makes sure we have secured all services and utilized all stakes of all validators. Then, $RS(G) \le (K - 1) + 1 = K$. ◀

Note that $RS(G) \le K$ in particular implies that $RS(G) \le \max_{v\in V}(d_G(v))$, as K is upper bounded by $\max_{v\in V}(d_G(v))$. There are examples where $K = \max_{s\in S}(d_G(s))$, implying that the upper bound can be as high as n. However, next, we show an upper bound as a function of the number of validators, which is asymptotically much lower than n.

▶ **Theorem 8.** *For any secure restaking graph G, the following inequality holds $RS(G) \le 2\sqrt{n} - 1$.*

Proof. The proof relies on a greedy covering of high-degree services. The proof combines constructions from the proof of Proposition 7. Initialize the set $S_L := \emptyset$ and $V_L := \emptyset$. The following procedure is repeated until it is no longer possible: At each step, find a service s in $S \setminus S_L$ such that its degree in the set $V \setminus V_L$ is at least $\sqrt{n}$. Update the set $S_L := S_L \cup s$ and $V_L := V_L \cup N_G(s)$. Note that after at most $\sqrt{n}$ steps, it is not possible to find a new service s and, hence, the procedure stops. That is, for the resulting set of services S_L, we have $|S_L| \le \sqrt{n}$. This, in particular, implies that the maximum outdegree of a validator v in V_L to S_L, denoted by $d_{S_L}(v)$, is upper bounded by $\sqrt{n}$.

Add to each validator $v \in V_L$ a stake amount $(d_{S_L}(v) - 1)\sigma_v$. This allows to secure any service $s \in S_L$ by using a construction $(\sigma_{v_1}, ..., \sigma_{v_{d_G(s)}})$ from the proof of Proposition 7. Together with securing all services in S_L, we also utilize validator stakes in V_L, and this is done by adding at most $(\sqrt{n} - 1)T(\sigma)$ extra stake.

We can secure any service $s \in S \setminus S_L$ by adding stakes π_s to all its neighbors in $V \setminus V_L$, that is, by adding $d_{V \setminus V_L}(s)\pi_s$ extra stakes. By the construction of the set S_L we know that $d_{V \setminus V_L}(s)$ is upper bounded by $\sqrt{n}$ for any $s \in S \setminus S_L$. We can, at the same time, utilize the remaining stakes in $V \setminus V_L$, by using a construction of type $(0, .., 0, \pi_s + \sigma_v, ..0)$ for $v \in V \setminus V_L$. The last step is possible, because for each validator in $V \setminus V_L$, there must be at least one service in $S \setminus S_L$ it secures. The total extra stake is upper bounded by $\sqrt{n}T(\pi_{S \setminus S_L}) \leq \sqrt{n}T(\sigma)$, where $T(\pi_{S \setminus S_L})$ denotes the sum $\sum_{s \in S \setminus S_L} \pi_s$. For a service $s \in S \setminus S_L$ it is possible that s has no neighbor in $V \setminus V_L$. That is, all of its neighboring validators were assigned to V_L. Let $S_0 \subseteq S \setminus S_L$ be the set of such services with 0 out-degree in $V \setminus V_L$. By the current construction, such a service receives zero stake. In this case, we let s add stake π_s to one of its previous neighbors, who are now in V_L, and require that validator to stake π_s to s in a PoS. This adds a stake amount of $T(\pi_{S_0})$, which contributes towards the stake values added for securing services in $S \setminus S_L$, as $S_0 \subseteq S \setminus S_L$. By summing up two values of extra stakes used to secure S_L and $S \setminus S_L$ sets, and also utilizing all stakes in V, we get

$$RS(G) \leq \frac{(\sqrt{n}-1)T(\sigma) + \sqrt{n}T(\sigma)}{T(\sigma)} = 2\sqrt{n} - 1,$$

a required bound of the Theorem claim. ◀

The number of unique addressed validators is high for the large-scale ecosystems such as Ethereum chain, approaching one million, while the real number of validators can still be on the order of tens of thousands. All validators are staked for Ethereum, that is, the restaking protocol based on Ethereum has the highest degree $\max_s d_G(s) = n$. The number of projects that require staking services is typically low, in the order of hundreds, and it is highly unlikely that a validator would provide services for most of them, therefore, it would be expected that $\max_v(d_G(v))$ remains low.

Note that in the proofs of Proposition 7 and Theorem 8, we did not use any property of α values. Both upper bounds use combinatorial features of the underlying restaking graph G and hold for any values of security parameters α. Next, we show an upper bound that is a function of the security parameters α_s. We show the following upper bound on $RS(G)$:

▶ **Proposition 9.** *Assume that services are labeled in the way that the security parameters are sorted in increasing order $\alpha_1 \leq \alpha_2 \leq ... \leq \alpha_m$. Then, $RS(G) \leq \frac{1}{\alpha_2}$.*

Proof. For the service $s = 1$, with the security parameter α_1, allocate the original stakes of the restaking graph in the PoS protocol: $(c_1, ..., c_{d_G(1)}) = (\sigma_{v_1}, ..., \sigma_{v_{d_G(1)}})$, where $N_G(1) = \{v_1, ..., v_{d_G(1)}\}$ is the set of validators staked on service $s = 1$. This PoS protocol is secure by the assumption that the original restaking graph is secure. Repeat the following procedure for all services $s > 1$:

1. add stake $t_s = \frac{\pi_s}{\alpha_s}$ to any validator $v \in N_G(s)$,
2. use construction $(\sigma_{v_1}, ..., \sigma_{v_{d_G(s)}})$ to secure service s in PoS protocol, where $N_G(s) = \{v_1, ..., v_{d_G(s)}\}$ is the set of validators staked on service s,
3. set all stake values in the vector $(\sigma_{v_1}, ..., \sigma_{v_{d_G(s)}})$ to 0.

Each PoS derived at step 2. is secure, as the total stake allocated for a service $s > 1$ is at least $\frac{\pi_s}{\alpha_s}$. This is an observation made in the proof of Proposition 4.

In this way, we add $\sum_{s \in S, s>1} \frac{\pi_s}{\alpha_s}$ total extra stake to the original stakes of validators. This implies the following chain of inequalities:

$$RS(G) \leq \frac{\sum_{s \in S, s>1} \frac{\pi_s}{\alpha_s}}{T(\sigma)} \leq \frac{1}{\alpha_2} \frac{\sum_{s \in S, s>1} \pi_s}{T(\pi)} \leq \frac{1}{\alpha_2},$$

finishing the proof of the proposition claim. ◀

The upper bound obtained in Proposition 9 is particularly relevant in practical cases, since the security parameters of most protocols are usually from the set of fractions $\{\frac{1}{4}, \frac{1}{3}, \frac{1}{2}\}$. These values come from Byzantine fault-tolerant mechanisms, used to achieve consensus on the state of the chain. In such cases, the bound simplifies to at most 2 to 4 times the original total stake, providing a tight estimate of the capital overhead required to achieve equivalent PoS security.

Next, we construct an example of a secure restaking graph deriving a lower bound on the restaking savings that is linear in the number of services and asymptotically matches to all upper bounds derived so far.

▶ **Theorem 10.** *For any $m \in \mathbb{N}$, there are instances of a secure restaking graph G in which the restaking savings $RS(G) \in \Omega(m)$.*

Proof. Assume the number of validators is $n = m^2 + 1$ for some $m \in \mathbb{N}$. The edge set E consists of m edges $(s, m^2 + 1)$, for any $1 \leq s \leq m$ and m^2 edges of type $(s, (s-1)m + j)$, for any $1 \leq s \leq m$, $1 \leq j \leq m$.

The value of service s is defined as $\pi_s = 2$ for any $1 \leq s \leq m$. Any validator with index less than $v < m^2 + 1$ has stake $\sigma_v = \frac{1}{m}$. The last validator has stake $\sigma_{m^2+1} = 2m$. Security parameter of service s is defined as $\alpha_s = \frac{1}{2m+1}$.

Intuitively, there is one validator with large stake and many validators with equal small stakes. Each service is easy to attack, but the large staked validator does not find it profitable to attack. On the other hand, each service is hard enough to attack by all small staked validators that secure it. Each service is secured by m small staked validators. See the Figure 1 for $m = 3$.

First, we show that G is a secure restaking graph. Note that the large validator, indexed with $m^2 + 1$ and with stake $\sigma_{m^2+1} = 2m$ does not want to participate in any attack, as it is losing at least the total value available across all services. No service $s \in \{1, 2, ..., m\}$ can be attacked by all validators securing it other than the large validator, since the service security parameter is $\alpha_s = \frac{1}{2m+1}$, while all small validators indexed $((s-1)m+1, (s-1)k+2, ..., sm)$ make up stake 1 in total. The total available stakes securing a service s is $2m + 1$.

To show a lower bound on $RS(G)$, first we show that when any service with value 2 and security parameter $\frac{1}{2m+1}$ gets initial allocation of stakes in PoS protocol equal to a vector $(\frac{1}{m}, ..., \frac{1}{m}, 0)$, there needs to be at least an extra m stake to make the PoS protocol secure. Suppose we add stakes such that new vector of stakes becomes $(a_1, ..., a_m, a_{m+1})$. From now on we distinguish three cases:

1. If there is a subset of validators, whose stakes sum up to a number between 1 and 2, then it must be that the total stake, $\sum_{i=1}^{m+1} a_i$, is at least $2m + 1$. Otherwise, this subset would be able to attack the service and the attack would be profitable. This implies that the extra stake size is at least $2m = 2m + 1 - m\frac{1}{m}$.
2. If there is a subset of validators, whose stakes sum up to a number between 0.5 and 1, then it must be that the total stake, $\sum_{i=1}^{m+1} a_i$, is at least $m + 1$. Otherwise, this subset would be able to attack the service and the attack would be profitable. This implies that the extra stake size is at least m.
3. If there is no such subset for either of the cases 1. and 2., then it must be that more than half of the numbers among $(a_1, ..., a_m, a_{m+1})$ are at least 2, which implies that at least $\frac{m+1}{2}(2 - \frac{1}{m}) \geq m$ extra stake was added to the initial stake distribution of $(\frac{1}{m}, ..., \frac{1}{m}, 0)$.

Initially, $2m$ stake of validator indexed $m^2 + 1$ is available, therefore, we need to add at least extra $m \cdot m - 2m$ stake:

- there are m services to secure,
- for each service we need to add extra stake at least m.

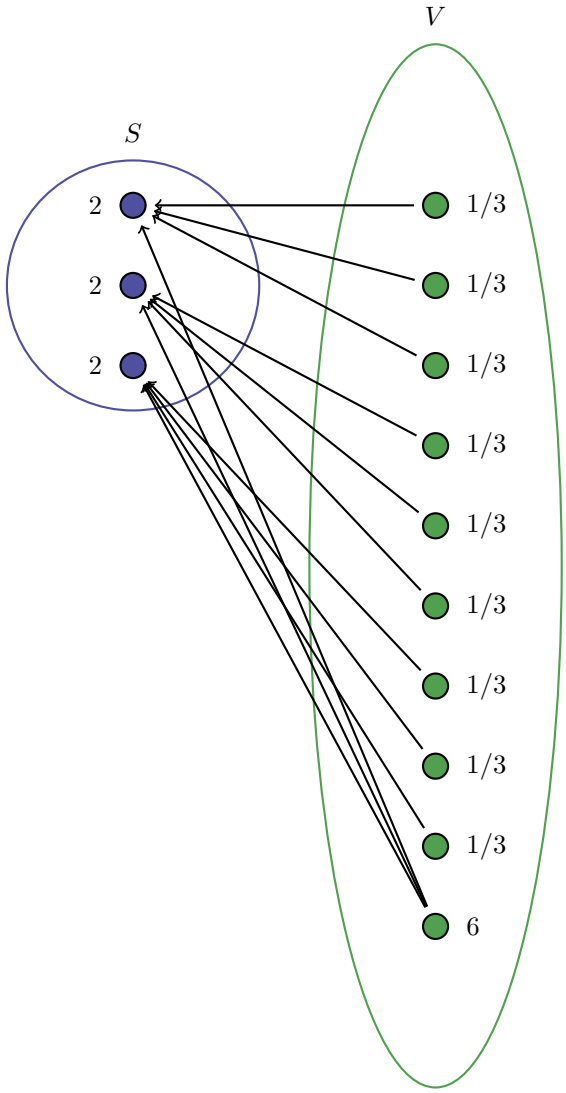


Figure 2 Example construction for $m = 3$.

On the other hand, original total stake amount is $T(\sigma) = 3m$, therefore, restaking savings satisfy

$$RS(G) \geq \frac{m \cdot m - 2m}{3m} \in \Omega(m).$$

This shows a lower bound on the required stake.

It is easy to show that $O(m^2)$ total stake is enough to add to validators in V, so that the resulting stake vector $\sigma \in \mathcal{W}(G)$. It can, for example, be done by adding the stake $2m^2 - m$ to the validator m^2+1 so that its total stake becomes $(2m+1)m$. Then, assume the validator splits this total stake into m equal stakes of $2m + 1$, for each m PoS protocols. The small stakers in these protocols can not attack any of the services even if they all coordinate, while the large staker does not find any attack profitable. This proves the claim of the theorem, $RS(G) \in \Theta(m)$. ◀

The sets S_L and V_L from the proof of Theorem 8 end up being the sets S and V, respectively. The highest degree in V_L is equal to $m = \lfloor \sqrt{n} \rfloor$. This provides an asymptotically matching lower bound to the upper bound result derived in Theorem 8.

▶ **Corollary 11.** *There are instances of secure restaking graphs G in which $RS(G) \in \Theta(\sqrt{n})$.*

Note that in the lower bound example construction from Theorem 10, inverse of the security parameter $\frac{1}{\alpha_s} = 2m+1$ for any $s \in S$, hence, this construction gives an asymptotically matching lower bound to the $\frac{1}{\alpha_s} - 1$ upper bound as well. The upper bounds obtained in Proposition 7, $\max_{s \in S} d_G(s)$ and K, are also of the order $\Theta(m)$, hence asymptotically matching the lower bound from Theorem 10.

The total stake in the example of Theorem 10 is equal to $3m$, while the lower bound requirement on the total stake derived from condition (1) is equal to $\sum_{s \in S} \frac{\pi_s}{\alpha_s} = m(2m+1)$, which shows that in some cases the condition (1) requires a substantially higher total stake than would be enough to secure the restaking graph. Moreover, the condition for any validator v except the last validator $v = m^2 + 1$ translates into

$$\frac{1}{m} \geq \frac{1/m}{2m+1} \cdot \frac{2}{\frac{1}{2m+1}} = \frac{2}{m},$$

which is violated by a factor two. The condition for the validator $v = m^2 + 1$ is violated by a very large multiplicative factor, as it is equivalent to:

$$2m \geq \sum_{s=1}^{m} \frac{2m}{2m+1} \cdot \frac{2}{\frac{1}{2m+1}} = 4m^2.$$

From the proof of Theorem 10 we can also observe that at least one of the individual validator stakes should be increased by a factor of $\Theta(m)$. Therefore, if restaking savings were to be measured like over-collaterization factor of individual validators, as in [5], the asymptotics would be the same. This adds to the strength of the lower bound example.

▶ **Definition 12.** *Let $RS := \sup_G RS(G)$ denote the restaking savings in the extremal case on all secure restaking graphs.*

Then, the direct corollary of Theorem 10 is the following:

▶ **Corollary 13.** *Restaking savings are not bounded from above, i.e. , $RS = \infty$.*

That is, restaking savings are not bounded by a constant, and they may grow unboundedly as the number of services and validators grows. Note that we need both the number of services and the number of validators to grow unboundedly. Otherwise, the upper bounds obtained in Proposition 7 ensure that $RS(G)$ is constant.

3.2 PoS Savings

Suppose a PoS protocol centered at any service $s \in S$, having security parameter α_s and allocated σ_v^s stakes for any $v \in N_G(s)$ is safe, by the same definition as above: no attacking set of validators finds it profitable to attack service s. Consider the total stake for any validator $v \in V$ defined as $\sigma_v = \sum_{s \in N_G(v)} \sigma_v^s$. That is, we aggregate all stakes that the validator has staked across all PoS protocols in which it participates. We ask a similar question to the previous section:

▶ **Question 2.** *How much stake do we need to add to the aggregate validator stakes, σ_v, so that the resulting restaking graph is secure?*

The security of the restaking graph is considered by the original definition 3. Let $U(G)$ denote the space of stake vectors for which G restaking graph is secure. Then, for each restaking graph, we define:

▶ **Definition 14.** *For a given aggregate restaking graph G, PoS savings, $PoSS(G)$ denotes a minimum additional total value to stakes in σ vector in relative terms to $T(\sigma)$ so that the resulting restaking graph G is secure:*

$$PoSS(G) = \min_{\sigma' \in D(\sigma) \cap U(G)} \frac{T(\sigma') - T(\sigma)}{T(\sigma)}$$

$PoSS(G)$ thus captures the inefficiency of aggregation, where interdependencies may necessitate over-collateralization to restore security. First, we note the following inequality that relates the total value to capture from all services and the total staked amount of validators. It will be useful to obtain an upper bound on the PoS savings.

▶ **Lemma 15.** *The total stake amount of all validators is more or equal than the total value derived from all projects: $T(\sigma) \geq T(\pi)$.*

Proof. A PoS protocol centered at $s \in S$ is secure implies that

$$\sum_{v \in N_G(s)} \sigma_v^s \geq \pi_s, \tag{2}$$

as otherwise all validators in $N_G(s)$ would be able to profitably attack the service s. Summing up left hand side of (2) for all $s \in S$ and the definition of σ_v imply the claim of the lemma:

$$T(\sigma) = \sum_{v \in V} \sigma_v = \sum_{s \in S} \sum_{v \in N_G(s)} \sigma_v^s \geq \sum_{s \in S} \pi_s = T(\pi). \qquad \blacktriangleleft$$

Similarly to restaking savings RS, we can define a measure $PoSS$, which is maximized on all instances of the underlying graph G and initial secure PoS protocols.

▶ **Definition 16.** *Let $PoSS := \sup_{G, \sigma_v^S} PoSS(G)$ denote the extreme value of PoS savings, where the stake vector $\{\sigma_v^s : v \in N_G(s)\}$ constitutes the secure PoS protocol for any $s \in S$.*

We obtain the following lower bound on the $PoSS$.

▶ **Proposition 17.** *There are instances of G so that $PoSS(G) \geq m - 1$.*

Proof. Suppose there are m services, each with a value of $\pi_s = 1$. Suppose there are m validators, and assume each validator secures all m services. For each service s, there is its corresponding dedicated validator s with a stake of $\sigma_s^s = 1$, and the other $m - 1$ validators a stake of ϵ each. Set the security parameter for each service to $\alpha_s = \frac{(m-1)\epsilon}{1+(m-1)\epsilon}$. First, we check the security of these PoS protocols. An attacking subset containing the dedicated validator would lose at least 1, which results in no profit. A subset excluding the dedicated validator has a maximum combined stake of $(m-1)\epsilon$, which gives a stake fraction of exactly α_s. Because an attack requires a stake fraction strictly greater than α_s, this subset cannot attack. Therefore, the independent PoS protocols are secure. The initial total stake is $T(\sigma) = m(1 + (m-1)\epsilon)$.

When aggregated into a restaking graph, each validator's total stake becomes $1+(m-1)\epsilon$. Because all validators secure all services, the total stake on any given service is $m(1+(m-1)\epsilon)$. Consequently, the fraction of the total stake controlled by a single validator on any service is exactly $\frac{1}{m}$. If we choose ϵ to be sufficiently small (specifically $\epsilon < \frac{1}{(m-1)^2}$), then $\frac{1}{m}$

becomes strictly greater than α_s. This means any single validator has enough stake fraction to independently attack any service. Since this applies to all services, every validator can profitably attack all m services, gaining a value of m while only losing a stake of roughly 1.

To secure this aggregated graph, we must eliminate this profitable attack for all m individual validators. For any given validator, we can remove the attack in one of two ways:

- Increase the total stake on enough services to force the validator's stake fraction to drop below α_s.
- Increase the validator's total stake to $\geq m$, so that attacking m services is no longer profitable.

Because ϵ can be chosen to be arbitrarily close to 0, blocking the attack via the α_s threshold requires an arbitrarily large amount of extra stake (inversely proportional to ϵ) to inflate the denominator of the attacking fraction. Therefore, the strictly cheaper option is to increase every validator's stake to m. The extra stake required for each validator is $m - (1 + (m-1)\epsilon)$. Across all m validators, the total extra stake E is $m(m - 1 - (m-1)\epsilon)$.

Then, PoSS for this instance is the ratio of the extra stake to the initial total stake:

$$PoSS(G) = \frac{E}{T(\sigma)} \geq \frac{m(m-1-(m-1)\epsilon)}{m(1+(m-1)\epsilon)}.$$

Taking the limit as ϵ approaches 0, this ratio simplifies exactly to $m - 1$. ◀

The lower bound can also be interpreted as $n - 1$, since by construction the number of validators is equal to the number of services. This stands in contrast to the upper bound obtained for restaking savings, which was of order $\Theta(\sqrt{n})$. Similarly to Corollary 13, we obtain:

▶ **Corollary 18.** *PoS savings are not bounded from above, i.e., $PoSS = \infty$.*

Next, we provide an upper bound on $PoSS(G)$, similar to the upper bound obtained for restaking savings in Proposition 7.

▶ **Proposition 19.** *For any underlying graph G, we have the following inequality: $PoSS(G) \leq \max_{s \in S} d_G(s) - 1$.*

Proof. First, we show that $PoSS(G) \leq \max_{s \in S} d_G(s)$, for the intuition. For each service $s \in S$, add to all its validators $v \in N_G(s)$ a stake equal to π_s. The resulting restaking graph is secure. For any subset of validators, $U \subseteq V$, they lose more cumulative stake than the value they derive from the maximal set they attack, $M(U)$, resulting into a not profitable attack. The reason for this inequality is that by our construction, for any $s \in M(U)$, at least one of the validators in U received an additional stake of π_s. We used an extra stake of

$$\sum_{s \in S} d_G(s)\pi_s \leq \max_{s \in S} d_G(s)T(\pi) \leq \max_{s \in S} d_G(s)T(\sigma),$$

where the last inequality is derived in Lemma 15. This completes the proof of the claim. Note that in this proof, we did not use original stakes validators are endowed to. Now suppose that to each validator $v \in V$ we add a stake equal to $\max(0, \sum_{s \in N(v)} \pi_s - \sigma_v)$. It is clear from the proof of the first claim that the resulting graph is secure.

In the following, we will try to upper bound the value

$$E := \sum_{v \in V} (\max(0, \sum_{s \in N_G(v)} \pi_s - \sigma_v)).$$

Let $\Delta_v := \sum_{s \in N_G(v)} \pi_s - \sigma_v$. The set of validators can be divided into two disjoint subsets V^+ and V^-, so that $V^+ := \{v \in V : \Delta_v \geq 0\}$ and $V^- := \{v \in V : \Delta_v < 0\}$. Let S^+ denote the set $\{s \in S : N_G(s) \subseteq V^+\}$ of the services whose validators are all in V^+, and $S' = S \setminus S^+$ be the set of remaining services, each having at least one validator in V^-. For $s \in S'$, $|N_G(s) \cap V^+| \leq d_G(s) - 1$. Then,

$$E = \sum_{v \in V^+} \Delta_v = \sum_{v \in V^+} (\sum_{s \in N_G(v)} \pi_s - \sigma_v) = \sum_{s \in S} \pi_s |N_G(s) \cap V^+| - \sum_{v \in V^+} \sigma_v. \tag{3}$$

Next, the following chain holds:

$$\sum_{s \in S} \pi_s |N_G(s) \cap V^+| = \sum_{s \in S^+} \pi_s |N_G(s) \cap V^+| + \sum_{s \in S'} \pi_s |N_G(s) \cap V^+| \leq$$
$$\sum_{s \in S+} \pi_s d_G(s) + \sum_{s \in S'} \pi_s (d_G(s) - 1).$$

Therefore, the extra added value to stakes can be upper bounded as

$$E \leq \sum_{s \in S+} \pi_s d_G(s) + \sum_{s \in S'} \pi_s (d_G(s) - 1) - \sum_{v \in V+} \sigma_v.$$

Since each PoS protocol is secure, we have $\sum_{v \in N_G(s)} \sigma_v^s \geq \pi_s$ for all $s \in S$, which implies that:

$$\sum_{s \in S^+} \sum_{v \in N_G(s)} \sigma_v^s \geq \sum_{s \in S^+} \pi_s.$$

For $s \in S^+$, $N_G(s) \subseteq V^+$, so the left hand side of the above is:

$$\sum_{v \in V^+} \sum_{s \in S^+ \cap N_G(v)} \sigma_v^s \leq \sum_{v \in V^+} \sum_{s \in N_G(v)} \sigma_v^s = \sum_{v \in V^+} \sigma_v. \tag{4}$$

Therefore, $\sum_{v \in V^+} \sigma_v \geq \sum_{s \in S^+} \pi_s$. Substituting in the upper bound on E, we get:

$$E \leq \sum_{s \in S^+} \pi_s d_G(s) + \sum_{s \in S'} \pi_s (d_G(s) - 1) - \sum_{s \in S^+} \pi_s = \sum_{s \in S} \pi_s (d_G(s) - 1).$$

Finally,

$$\sum_{s \in S} \pi_s (d_G(s) - 1) \leq (\max_{s \in S} d_G(s) - 1) T(\pi) \leq (\max_{s \in S} d_G(s) - 1) T(\sigma). \quad \blacktriangleleft$$

Note that the upper bound in Proposition 19 exactly matches the lower bound obtained in Proposition 17, unlike the case of restaking savings, in which case the lower and upper bounds are only asymptotically matching.

4 Conclusion

We introduce a formal framework for comparing the capital efficiency of secure restaking protocols and equivalent independent Proof-of-Stake (PoS) protocols, measured by the total stake. Our main results establish tight asymptotic bounds on the restaking savings (the relative extra capital needed to redistribute stakes from a secure restaking graph into separate secure PoS protocols): we prove matching upper and lower bounds of $\Theta(\sqrt{n})$, where n is the number of validators. These bounds are achieved through greedy covering technique and explicit construction, showing that restaking can deliver meaningful capital efficiency gains in large-scale systems.

In the complementary direction, we demonstrate that aggregating secure PoS protocols into a secure restaking graph can incur a significant overhead. We provide matching lower and upper bounds on PoS savings (the extra stake needed post-aggregation to restore security), equal to maximum service degree in the restaking graph minus one. Taken together, our results indicate that neither restaking nor independent PoS protocols dominate each other in terms of capital requirements. Each concept has structural advantages depending on the underlying graph.

References

1 Eric Budish. Trust at scale: The economic limits of cryptocurrencies and blockchains. *The Quarterly Journal of Economics*, 140:1–62, 2025.

2 Tarun Chitra and Mallesh Pai. How much should you pay for restaking security? In *Web and Internet Economics, WINE*, 2024.

3 Tarun Chitra, Paolo Penna, and Manvir Schneider. On sybil-proofness in restaking networks. *CoRR*, abs/2509.18338, 2025. doi:10.48550/arXiv.2509.18338.

4 Soubhik Deb, Robert Raynor, and Sreeram Kannan. Stakesure: Proof of stake mechanisms with strong cryptoeconomic safety. *CoRR*, abs/2401.05797, 2024. doi:10.48550/arXiv.2401.05797.

5 Naveen Durvasula and Tim Roughgarden. Robust restaking networks. In Raghu Meka, editor, *16th Innovations in Theoretical Computer Science Conference, ITCS 2025, January 7-10, 2025, Columbia University, New York, NY, USA*, volume 325 of *LIPIcs*, pages 48:1–48:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ITCS.2025.48.

6 Cyril Monnet. Rehypothecation. *Business Review, Federal Reserve Bank of Philadelphia*, pages 18–25, 2011.

7 Abhimanyu Nag, Dhruv Bodani, and Abhishek Kumar. Economic security of multiple shared security protocols. In *International Conference on Mathematical Research for Blockchain Economy (MARBLE)*, 2025.

8 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

9 Fahad Saleh. Blockchain without waste: Proof-of-stake. *The Review of Financial Studies*, 34:1156–1190, 2021.

10 EigenLayer Team. Eigenlayer: The restaking collective, 2023. URL: https://docs.eigenlayer.xyz/eigenlayer/overview/whitepaper.

11 Roi Bar Zur and Ittay Eyal. Elastic restaking networks: United we fall, (partially) divided we stand. In Chun-Ying Huang, Jyh-Cheng Chen, Shiuh-Pyng Shieh, David Lie, and Véronique Cortier, editors, *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, CCS 2025, Taipei, Taiwan, October 13-17, 2025*, pages 3476–3489. ACM, 2025. doi:10.1145/3719027.3765077.

StakeNote: A Proof-Of-Stake Protocol for CryptoNote Payments

Bernardo David
IT University of Copenhagen (ITU), Denmark
Common Prefix, Copenhagen, Denmark

Dimitris Karakostas
Common Prefix, Athens, Greece

Abstract

This work proposes StakeNote, a distributed ledger protocol that combines Proof-of-Stake (PoS) with privacy and anonymity preserving payments. The protocol combines Ouroboros Praos, a provably secure PoS protocol, with CryptoNote, a privacy-preserving payment system based on ring signatures which has been widely used in practice. We prove that StakeNote inherits the security guarantees of Ouroboros Praos and the privacy guarantees of CryptoNote. We also demonstrate its practicality via a proof of concept implementation of its cryptographic components, showing that block signing requires less than 25 ms and eligibility proofs are approx. 3 KB for anonymity sets of size 16. Finally, we discuss heuristic enhancements that potentially increase privacy and enable dynamic participation.

2012 ACM Subject Classification Security and privacy

Keywords and phrases Proof-of-Stake, anonymity, CryptoNote, ring signatures

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.8

Related Version *Full Version*: `https://eprint.iacr.org/2026/1205.pdf` [13]

Supplementary Material
Software (Source Code): `https://github.com/commonprefix/stakenote` [12]
archived at `swh:1:dir:2ee7704e94b9fa3526b672fabc2b7cc381af6830`

Funding The project was partially funded by Zano and Common Prefix.

Acknowledgements We thank Andrey Sabelnikov and Sowle for their helpful discussions, questions, and feedback.

1 Introduction

Bitcoin [33] introduced a new paradigm for distributed computing and financial applications. On the technical side, Proof-of-Work (PoW) sybil resilience enabled an open and permissionless database, where participants join and leave the protocol arbitrarily. On the application side, the Bitcoin cryptocurrency established Decentralized Finance (DeFi) [40], which was later enhanced with programmability capabilities via smart contracts. Still, Bitcoin left a lot to be desired.

The first concern is Bitcoin's sustainability. Bitcoin mining consumes such amounts of energy [18, 14] that call for regulatory or policy interventions [5]. This concern drove research to more sustainable approaches, the most promising being Proof-of-Stake (PoS) [26, 11]. PoS replaces physical assets, namely energy in PoW, with digital assets, namely the assets ("stake") traded on the ledger itself.

A second concern stemmed from Bitcoin's public nature. By default all information on Bitcoin, often of sensitive financial nature, is public and readable by everyone. To avoid this, various protocols aim to guarantee user anonymity and privacy, with the most popular being Zerocash [3] and CryptoNote [38], used by the Zcash and Monero platforms respectively.


© Bernardo David and Dimitris Karakostas;
licensed under Creative Commons License CC-BY 4.0

Naturally, these concerns have led to the exploration of combined solutions, i.e., privacy-preserving PoS protocols. Unfortunately, although the literature does offer such protocols [25, 20], none has been deployed in practice. As academic research has mostly focused on Zerocash, little progress has been made on practical privacy-preserving PoS protocols using CryptoNote. This lack of research can be attributed to multiple reasons. First, Zerocash is suitable for academic research, since it is provably secure and offers well defined strong privacy guarantees. On the contrary, CryptoNote has not been formally defined nor proven secure yet. In fact, a there is a line of works on tracing CryptoNote transactions [28, 32, 43, 24, 41, 42] under different conditions and different degrees of success, showing that the exact anonymity guarantees are unclear. While there exist provably secure cryptocurrency schemes [30, 16] based on the CryptoNote approach, those schemes assume an underlying distributed ledger and are not directly compatible with existing PoS consensus protocols. As a result, none of the systems that use Cryptonote is pure PoS: Monero[1] uses PoW, Zano[2] uses an ad hoc hybrid of PoW and PoS, whereas Sentz[3] uses a custom quorum-based mechanism inspired by Stellar [29].

Our work fills this gap by proposing a *ring-friendly, secure, and practical* PoS distributed ledger protocol for CryptoNote payments. Our protocol is: (i) secure, meaning it guarantees safety and liveness; (ii) as private as CryptoNote; (iii) practical, i.e., does not rely on heavy and inefficient cryptographic primitives and is as efficient as deployed PoS systems. Essentially, our protocol is a realistic option for privacy preserving blockchain systems, in order to move away from PoW towards a more environmentally sustainable alternative.

Related Work

Two design philosophies dominate the literature on private payments. The first, introduced by Zerocash [3], uses succinct non-interactive zero-knowledge proofs to hide each transaction within the entire set of unspent outputs. The second, introduced by CryptoNote [38], uses ring signatures to hide each transaction within an explicit anonymity set of decoys, chosen at signing time. The two approaches expose a fundamental tradeoff between the strength of the anonymity guarantee and the strength of the setup assumption. Zerocash-style protocols offer anonymity within the full UTxO set, but rely on a structured common reference string (CRS) whose generation requires a trusted setup ceremony and, in classical instantiations, must be repeated for each relation.[4] CryptoNote-style protocols require no setup beyond the random oracle, but their anonymity is bounded by the anonymity set used in each transaction and is sensitive to decoy selection.

The same tradeoff carries over to privacy-preserving PoS protocols. Ouroboros Crypsinous [25] embeds a Zerocash-style payment scheme within a PoS leader election, while the privacy-aware blockchain of [20] assumes a list of commitments to each party's stake (which RingCT [35] already provides) and builds the eligibility proofs from generic non-interactive zero-knowledge proofs. Both inherit the structured CRS of the underlying proof system. A modular formalization of anonymous PoS leader election [2], instantiated with Algorand's selection function, similarly relies on non-interactive zero-knowledge proofs when parties hold arbitrary stake amounts. To our knowledge, no prior PoS protocol explores the CryptoNote side of the tradeoff. StakeNote fills this gap by offering the same (locally bounded) anonymity as CryptoNote and requiring no structured setup.

[1] `https://www.getmonero.org`
[2] `https://zano.org`
[3] `https://www.sentz.com`
[4] Recent transparent or universal-updatable proof systems reduce or remove this dependence, but to our knowledge no privacy-preserving PoS protocol has been built atop them.

We note that an orthogonal line of work studies the inherent privacy limitations of anonymous PoS, both information-theoretically [27] and under differential privacy [39]. These results apply to any anonymous PoS protocol, including ours, and characterize the anonymity that any such construction can hope to achieve.

2 Model and Preliminaries

We now outline our security model and assumptions, along with the cryptographic primitives and protocols on which our protocol relies.

2.1 Security Model and Assumptions

Delayed Adaptive Adversaries. Instead of security fully adaptive adversaries as achieved in Ouroboros Praos, we consider a weaker form of delayed adaptive adversary. Such adversaries may request to corrupt any party at any point in time but the corruption takes places only after a preset delay. As in the original Ouroboros protocol, we set this delay to be the duration of an epoch, such that any corruption request made in a certain epoch only gives control over the corrupted party to the adversary in the next epoch. This ensures that an adaptive adversary cannot create arbitrary conflicting versions of a block by immediately corrupting a slot leader after it broadcast a block.

Network Model. As in Ouroboros Praos, we consider a semi-synchronous network, where the adversary $\mathcal{A}$ can selectively delay any messages sent by honest parties for up to $\Delta \in \mathsf{poly}(\lambda)$ slots where λ is the security parameter. This is formalized in the diffuse functionality discussed in Section 2.2.

GKL security properties. As in Ouroboros Praos, we adopt the GKL model [21] to capture the security properties, namely, common prefix, chain quality and chain growth. We briefly recall these properties informally. Common prefix states that for the blockchain in the view of any honest party, removing a suffix of k blocks results in a prefix that is in the blockchain in the view of all other honest parties. Chain quality states that at least one block generated by an honest party appears in a portion of s blocks of the blockchain in an honest party's view. Chain growth states that, every s slots, at least $\tau \cdot s > 0$ blocks are produced.

Execution Model. As In Ouroboros Praos, we adopt the GKL model [21] to capture the execution of our protocol. In this model, an *environment* orchestrates the execution by instructing parties to create new transactions, diffusing transactions among parties (i.e., emulating the inclusion of transactions in a mempool via a peer-to-peer network) and instructing the adversary to corrupt parties. The execution is augmented by a *functionality*, which models among other things semi-synchronous network communication among parties and setup assumptions (i.e., the initial stake distribution and key generation for parties present in the beginning of the execution). For any environment and adversary, the GKL security properties must hold for the blockchains in the view of all honest parties, except with negligible probability. In order to achieve these properties, we must restrict the environment to activate all honest parties often enough and not to allow the adversary to corrupt parties that jointly control 50% or more of the stake.

ℓ-DDH Hardness Assumption. It was shown in [34] that if the standard DDH-problem is hard, then the ℓ-DDH problem is hard for $\ell(\lambda) = poly(\lambda)$ and the distinguishing advantage is the same. Note that the 1-DDH hardness assumption is the usual DDH hardness assumption.

▶ **Definition 1** (ℓ-DDH hardness assumption)**.** *Let $\ell := \ell(\lambda)$ be an integer depending on the security parameter λ. Let D_0, D_1 be the probability distributions outputting tuples $\mathbf{x} := (G, \alpha \cdot G, \{\beta_i \cdot G\}_{i=0}^{\ell-1}, \{\gamma_i \cdot G\}_{i=0}^{\ell-1})$ where G is a uniformly random generator of a group $\mathbb{G}$ of order q, and $\alpha, \{\beta_i\}_{i=0}^{\ell-1}$ are independently sampled uniformly random values in $\mathbb{F}_q$. In the case of D_0, $\gamma_i = \alpha \cdot \beta_i$ for all $i \in [0, \ell-1]$. In the case of D_1, $\{\gamma_i\}_{i=0}^{\ell-1}$ are sampled uniformly at random from $\mathbb{F}_q$ and independently of each other and of everything else. Given a PPT adversary $\mathcal{A}$, we define its distinguishing advantage as the probability $Adv(\mathcal{A}) = \left| Pr[b = b' \mid b \xleftarrow{\$} \{0,1\}, \mathbf{x} \xleftarrow{\$} D_b, b' \xleftarrow{\$} \mathcal{A}(\mathbf{x})] - 1/2 \right|$. We say that the ℓ-DDH problem is hard if for every PPT adversary $\mathcal{A}$, $Adv(\mathcal{A})$ is negligible in λ.*

Random Oracle Model. Throughout this paper we model hash functions as random oracles with the appropriate output space. This is crucial for the VRF construction and block signature (i.e., modified CLSAG) we adopt.

2.2 The Ouroboros Family of PoS Protocols

Ouroboros Praos is a Proof-of-Stake (PoS) distributed ledger protocol which is secure against adaptive corruptions under a semi-synchronous network [11].

In Ouroboros Praos, time is divided into discrete units called *slots*. In the context of a blockchain-based distributed ledger, each time slot is associated with at most one *block*. Parties are equipped with synchronized clocks that indicate the current slot, such that clock drift is subsumed in the slot's length. In general, each slot sl_r is indexed by an integer $r \in \{1, 2, \ldots\}$, where the real time window that corresponds to each slot has two properties. The network is semi-synchronous, i.e., the adversary $\mathcal{A}$ can selectively delay any messages sent by honest parties for up to $\Delta \in \mathsf{poly}(\lambda)$ slots where λ is the security parameter.

In Ouroboros Praos, each party U_i is identified by: (i) an amount of stake $v_i \in \mathbb{N}$; (ii) a Key-Evolving Signature (KES) public key $\mathsf{vk}_i^{\text{kes}}$; (iii) a Verifiable Random Function (VRF) public key $\mathsf{vk}_i^{\text{vrf}}$; (iv) a regular EUF-CMA secure digital signature public key $\mathsf{vk}_i^{\text{dsig}}$.

The party's stake v_i defines the party's power (weight) within the consensus protocol. Intuitively, the probability of a party becoming eligible for participation is a (almost) linear function of its stake. For each slot, every party may be elected as slot leader, which allows them to create a block that corresponds to the slot.

The leader election is based on local sortition and implemented via a Verifiable Random Function (VRF) [31]. Specifically, each party computes locally the output of the VRF, using their VRF private key and the slot index as input. If the VRF output is below a threshold, which is a (slightly concave) function of their stake, then they are an eligible slot leader.

After being elected as slot leader, a party constructs a block by collecting unpublished transactions and relevant metadata (e.g., timestamps). The party also includes the VRF eligibility proof, which can be verified by anyone using the party's public VRF key. Finally, the party signs the block using their private KES key, evolve it, and immediately delete the used key. In this manner, even if the adversary corrupts the party immediately afterwards, they cannot retroactively create blocks for past slots for which the party was elected as leader, since the corresponding block signing key has been evolved and deleted.

Finally, the party may create a transaction that spends their stake using their regular key $\mathsf{vk}_i^{\text{dsig}}$. To support dynamic stake distributions, where parties can spend their assets and stake distribution shifts, Ouroboros Praos is organized in *epochs*. Before each epoch,

the stake distribution is captured in a snapshot and protocol participation within the epoch follows the snapshot. Additionally, a randomness beacon, which is simulatable using a hash function, defines a pseudorandom nonce per epoch, used to randomize the leader election process.

Leader election. A core requirement of our protocol is *sybil resilience and collusion proofness*. A stakeholder should be independently selected, with probability p_i which depends only on its relative stake. In Ouroboros Praos, the probability is determined by the stakeholder's relative stake and a parameter f, called the *active slots coefficient*.[5] Specifically, $p_i = \phi_f(\alpha_i) \triangleq 1 - (1-f)^{\alpha_i}$, where α_i is the relative stake held by stakeholder U_i ($\alpha_i = \frac{v_i}{\sum_j v_j}$).

This function has the desirable property of "aggregation independence", meaning that a party controlling relative stake $\frac{v}{v_{\mathsf{total}}}$ is elected with probability $\phi_f(\frac{v}{v_{\mathsf{total}}})$ no matter how they spread their stake among many UTxOs (or a few).

Note that, since the events "U_i is a leader for sl_j" are independent, this mechanism may generate multiple or no leaders for a given slot.

Diffuse functionality. The diffuse functionality $\mathsf{DDiffuse}_\Delta$ models the communication network. It is parameterized by $\Delta \in \mathsf{poly}(\lambda)$ and interacts with (i) the environment $\mathcal{Z}$, (ii) parties (stakeholders) $U_1, \ldots, U_n$, (iii) and the adversary $\mathcal{A}_{\mathsf{net}}$. On each round, every activated party can fetch incoming messages, which have been diffused via $\mathsf{DDiffuse}_\Delta$ by some other party, and diffuse their own messages (once per round). Once activated, $\mathcal{A}_{\mathsf{net}}$ can read all inboxes and diffuse requests and deliver messages to parties in any order it prefers. Additionally, $\mathcal{A}_{\mathsf{net}}$ can delay the deliver of any message to any party up to Δ slots (at which point the functionality delivers the delayed message to the respective parties' inboxes).

Ouroboros Genesis Fork Choice Rule. For the fork choice rule, we turn to Ouroboros Genesis [1], the followup to Ouroboros Praos. One drawback of Ouroboros Praos was the usage of "moving checkpoints" for long-range attack prevention. Briefly, if a new chain forks the party's chain by more than k blocks, then the new chain is discarded. Due to the checkpoints, newly-joining parties needed to acquire a relatively recent finalized block from a trusted source in order to properly "bootstrap". Ouroboros Genesis avoids this requirement via a fork choice rule that enables a newly-joining party to correctly bootstrap while having access only to the protocol's genesis block. Specifically, under Genesis's fork choice rule, given two chains that fork by more than k blocks, the party chooses the one that grows at a faster rate immediately after the forking point. In the context of our work, since our protocol is oblivious to the chain selection rule, we opt for the Ouroboros Genesis rule which gives the best guarantees.

2.3 CryptoNote and Confidential Transactions

CryptoNote is a distributed ledger protocol that aims to offer transaction untraceability [38]. The original CryptoNote whitepaper proposed various advancements on Bitcoin's blockchain-based payment system, such as an ASIC-resistant Proof-of-Work (PoW) function, as well as an alternative coin generation function and scripting language. Nonetheless, its main contribution was a transaction mechanism, which is based on traceable ring signatures [19].

[5] A party controlling all stake has probability f to generate a block: $\phi_f(1) = f$.

At a high level, the CryptoNote protocol inherits the UTxO model of Bitcoin. Specifically, each unspent output holds an amount of assets, which (in the original version of CryptoNote) is public. In order to spend a UTxO the party picks a subset of all outputs, whose amount is equal to the UTxO. Following, the UTxO's owner creates a ring signature, which proves that they own (the private key of) one of the outputs, among the chosen subset plus the correct UTxO.

In order to avoid double-spending, the signature is *one-time* via the usage of *key images*. Specifically, each UTxO is controlled by a one-time private key. To consume the UTxO, the party creates a public key using the (UTxO's) private key, called a "key image". The key image is not linked with the public key which is used within the ring signature of the transaction, but is deterministically produced using the same private key. Therefore, when receiving a transaction, participants cross-reference its key images with the set of all past key images; if the key image has been observed before, i.e., if the private key has been used in a past transaction, then the new transaction is rejected.

Finally, CryptoNote enables a party to publish a single address (i.e., public payment key) and receive arbitrary unlinkable payments. To do so, the sender of a transaction creates a one-time public key using a random mask. Following, the recipient observes all newly-created outputs. If a transaction is aimed at them, the recipient can both identify it and recover the corresponding private key.

Confidential Transactions. Confidential transactions (CT) is a method for hiding a UTxO's value [37]. The main idea is using homomorphic commitments, such as Pedersen commitments [36]. Specifically, each UTxO is linked with a commitment of its asset amount. Following, a transaction includes a *range proof*, which proves that (i) the sum of all (committed) input values is greater than the sum of all (committed) output values; (ii) each output value is positive. In other words, the range proof shows that the transaction does not generate new assets.

Ring Confidential Transactions. A notable building block of our work is Ring Confidential Transactions (RingCT) [35]. RingCT combines the ring signatures of CryptoNote with the CT scheme, resulting in a payment protocol with both transaction unlinkability and amount hiding.

Concise Linkable Spontaneous Anonymous Group (CLSAG) proofs. Finally, our construction relies heavily on CLSAG proofs and signatures [23]. Briefly, CLSAGs are ring signatures inspired by Monero's Multilayered Linkable Spontaneous Anonymous Group (MLSAG) Signatures [35], but with performance enhancements that are particularly useful in our case.

2.4 Building Blocks

Verifiable Random Functions (VRFs). Our protocol uses Verifiable Random Functions (VRF) [31] with unpredictability under malicious key generation [11, 22]. VRFs with unpredictability under malicious key generation consist of three algorithms: $\mathsf{VRF.KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$: on input a security parameter λ outputs a secret key sk and a public key pk; $\mathsf{VRF.Eval}(\mathsf{sk}, x) \to (y, \pi_y)$: on input a secret key sk and a value x, outputs a pseudorandom value $y \in \{0,1\}^{l_{\mathsf{VRF}}}$ and a proof π_r; $\mathsf{VRF.Verify}(\mathsf{pk}, x, y, \pi_y) \to 0/1$: on input a public key pk, a value x, a value y, and a proof π_y, outputs 1 if and only if the proof verifies.

Range proofs. In order to prove eligibility, StakeNote also uses a range proof for a specially crafted commitment. Various protocols can be used to implement range proofs, with varying performance and trust assumptions. One such protocol was described in the "confidential assets" paper [37] and does not require a trusted setup. Another zero-knowledge proof protocol, which has been widely used in a similar context of confidential payments, is Bulletproofs [4]. For a systematic review of the space of zero-knowledge range proofs we refer to [8].

3 StakeNote Protocol

At a high level, the protocol follows the same steps as Ouroboros Praos [11], with some changes in the public setup and leader eligibility proofs. In our setting, each output consists of a public key and a commitment to the output's stake value. To compute each epoch's snapshot, parties should know when and if an output has been spent; for this we require the block generator to prove that they had an unspent output at the time of the snapshot when they create a block.

On each slot, every party takes the following steps to create a block:[6]

1. $\mathcal{P}$ creates a VRF output, using the output's VRF key, in the same manner as Praos, i.e., passing as input the epoch's nonce and slot number.
2. If the VRF output is below some threshold, which is a function of the output's stake value, then $\mathcal{P}$ is the slot leader and proceeds as follows:
   a. $\mathcal{P}$ collects a subset of the snapshot's UTxOs including the chosen output.
   b. $\mathcal{P}$ creates a block with the following proofs:
      i. A proof that $\mathcal{P}$ knows the spending key of an output in the chosen subset and that it was unspent at the time of the snapshot.
      ii. A proof that the private VRF key, which created the VRF output from Step 1, is deterministically derived from the private payment key of the same output from Step 2(b)i.
      iii. A proof that the stake value of the same output from Step 2(b)i was used to compute the threshold of Step 2.
   c. $\mathcal{P}$ signs the block by turning the proofs from Step 2b to a signature of knowledge, proving that the private signing key is the same as the spending key of the output from Step 2(b)i.

3.1 Construction

At its core, our construction relies on a proof which shows that the block is produced by some eligible UTxO, among a subset of UTxOs in the snapshot, without revealing which specific UTxO is being used.

The eligibility criterion is the same as Ouroboros Praos, i.e., the output of a VRF evaluated on a random nonce under the VRF secret key corresponding to the VRF public key in the UTxO should be below a certain threshold determined by the leader election probability function. Each UTxO models a single party, so it is associated with a VRF key. Notably, a UTxO's private VRF key is chosen to be the same scalar as the UTxO's private payment key, i.e., the key that is used to consume the output.

[6] If $\mathcal{P}$ controls multiple UTxOs, then the steps are executed for every output.

Briefly, $\mathcal{P}$ generates the VRF key pair in a way that the secret key is the same as the payment key, computes the VRF output and checks whether he is elected as slot leader, in the same manner as Ouroboros Praos. If so, $\mathcal{P}$ creates the block, proves that the VRF secret key is the same as the UTxO's payment key and that the UTxO's (committed) stake value is above a certain value such that the VRF output is below the threshold determined by the leader election probability function evaluated on that value, and finally creates a ring signature on the block with the UTxO's payment key and a set of decoys. Crucially, all of the above is done in a way that *does not reveal* the exact UTxO that is being used, but only reveals a set of UTxOs which includes the correct one.

Snapshot

As in Ouroboros Praos, the stake distribution used to elect slot leaders at a certain epoch must be "frozen" before the random nonce for that epoch is generated. This means that for leader election in a given epoch we must take into consideration only UTxOs that appear in blocks far enough in the past of the chain, *before* the protocol for generating the random nonce is initiated. Moreover, we must ensure that all honest transactions in this snapshot are seen by all honest parties. In practice, this means that a snapshot used for leader election in a given epoch contains all of the UTxOs that appear in blocks generated 2 epochs ago, as the random nonce generation protocol is executed in the previous epoch.

Differently from Ouroboros Praos, as we consider CryptoNote transactions, our snapshot specifically contains both the set $\mathbb{O}_e$ of UTxOs that appeared in the blocks generated up to 2 epochs before epoch e (*until epoch* $e-2$) and the set $\mathbb{I}_e$ of key images that have appeared in the blocks generated up epoch $e-2$. Notice that, by proving that it controls an UTxO in $\mathbb{O}_e$ whose key image is *not* in $\mathbb{I}_e$, a party proves that this UTxO was unspent at the time of the snapshot, thus that it is eligible for the leader election procedure.

Looking forward, another particularity of the use of snapshots in our case is that, when using a UTxO to generate a modified CLSAG as part of a leader election proof, a party must only *choose decoy UTxOs that are also in the snapshot*. In particular, this means that once a UTxO has been used in such a modified CLSAG, it must always be used with the same set of decoys. Otherwise, the anonymity set protecting that UTxO diminishes, as it will be known that the UTxO used in all modified CLSAGs is the same (because the key image is revealed in the newly generated blocks) and that it must be in the intersection of all the decoy sets used in the newly generated blocks.

Objects and notation

- Global protocol parameters:
  - G_1, G_2, G_{PAY}, G_{VRF}: group generators of finite cyclic groups over a finite field F_p for some prime p.
  - n: ring size of CryptoNote payment transactions.
  - random oracle H_{clsag}.
  - δ: snapshot height delta, i.e., the slot distance between the snapshot's taking and the beginning of the epoch.
  - v_{total}: the total amount of stake in the system.
- For every UTxO (output) o, we have the following:
  - private elements: (i) sk_o: private payment key; (ii) v_o: stake value; (iii) r_o: randomness of value's Pedersen commitment.
  - public elements: (i) $\mathsf{vk}_o = \mathsf{sk}_o \cdot G_{\mathsf{PAY}}$: public payment key; (ii) $C_o = v_o \cdot G_1 + r_o \cdot G_2$: Pedersen commitment to output's stake value.

- For every spent UTxO (output) o, we have the output's key image $\Im_o = \mathsf{sk}_o \cdot H(\mathsf{vk}_o)$ that is *not linkable* to o.
- For every UTxO (output) o used to generate a block, we have the following public elements that are *not linkable* to o: (i) the output's key image $\Im_o = \mathsf{sk}_o \cdot H(\mathsf{vk}_o)$; (ii) the VRF public key $\mathsf{vk}_{\mathrm{VRF},o} = \mathsf{sk}_o \cdot G_{\mathsf{VRF}}$.
- Output sets: (i) $\mathbb{O}$: all outputs that have ever appeared; (ii) $\mathbb{I}$: all key images that have ever appeared.
- Epoch e: (i) sl_e: epoch start (slot); (ii) η_e: epoch random nonce.
- Snapshot sets for epoch e: (i) $\mathbb{O}_e \subset \mathbb{O}$: outputs in the snapshot (UTxOs that appeared in blocks produced up to epoch $e-2$); (ii) $\mathbb{I}_e \subset \mathbb{I}$: key images in the snapshot (UTxOs that appeared in blocks produced up to epoch $e-2$).

Block creation

We present the protocol for block creation in Figure 1. The protocol is run in the context of an epoch e for which a random nonce η_e has been generated in the previous epoch (i.e., in epoch $e-1$) following the same procedure as in Ouroboros Praos. Moreover, the epoch defines a snapshot consisting of a set $\mathbb{O}_e$ of outputs and a set $\mathbb{I}_e$ of key images that appear in blocks created before the previous epoch (i.e., block created up to epoch $e-2$).

In our protocol, each UTxO o is associated with private values sk_o, v_o, r_o that implicitly define a VRF key pair $(\mathsf{sk}_o, \mathsf{vk}_{\mathrm{VRF},o})$ where $\mathsf{vk}_{\mathrm{VRF},o} = \mathsf{sk}_o \cdot G_{\mathsf{VRF}}$. A party $\mathcal{P}$ is elected as a slot leader (i.e., allowed to create a block) if, for any of their UTxOs in $\mathbb{O}_e$, they can obtain a small enough VRF output $y_o < 2^{l_{\mathsf{VRF}}} \phi_f(\frac{v_o}{v_{\mathsf{total}}})$ by evaluating the VRF on the epoch's nonce η_e using the UTxO's sk_o as secret key. Intuitively, this allows the party to prove they were elected by publishing $(\mathsf{vk}_{\mathrm{VRF},o}, y_o, \pi_{y_o})$ in their block, but it leaves the challenge of proving that $\mathsf{vk}_{\mathrm{VRF},o}$ and y_o are valid without revealing the output o or the amount v_o.

In order to preserve privacy for v_o, we instead reveal a value $T < v_o$, such that $y_o < 2^{l_{\mathsf{VRF}}} \phi_f(\frac{T}{v_{\mathsf{total}}})$. However, we need to prove that $T < v_o$ without revealing y_o, which we do by revealing a commitment $\mathcal{C}$ to the UTxO value minus T and using a range proof π_{RANGE} to show it contains a positive value. In order to preserve privacy for o and sk_o, we follow the CryptoNote approach by revealing a set of UTxOs $\mathbb{O}_{\mathsf{block}} \subset \mathbb{O}_e$, such that $o \in \mathbb{O}_{\mathsf{block}}$ (instead of revealing o itself), while proving that in $\mathbb{O}_{\mathsf{block}}$ there is a UTxO with sk_o, v_o, r_o, such that $\mathsf{vk}_{\mathrm{VRF},o} = \mathsf{sk}_o \cdot G_{\mathsf{VRF}}$ and that $\mathcal{C}$ is correctly computed from v_o, r_o, T.

Putting the previous ideas together, we want to create a block by generating a signature on its payload B and proving correctness of revealed values $(\mathsf{vk}_{\mathrm{VRF},o}, y_o, \pi_{y_o}), \mathcal{C}, T, \pi_{\mathsf{RANGE}}$, which allow us to check that the slot leader $\mathcal{P}$ indeed was elected to create a new block. Moreover, we need to show that the UTxO $o \in \mathbb{O}_e$ was unspent when the snapshot was taken. When creating a block, $\mathcal{P}$ publishes a random set of UTxOs $\mathbb{O}_{\mathsf{block}} \subset \mathbb{O}_e$ containing o (plus decoys) together with a proof of knowledge of private values sk_o, v_o, r_o for o such that the following statements hold *without revealing the private values* sk_o, v_o, r_o *or* o:

1. $o \in \mathbb{O}_{\mathsf{block}}$.
2. o can be spent using sk_o.
3. The VRF public key is $\mathsf{vk}_{\mathrm{VRF},o} = \mathsf{sk}_o \cdot G_{\mathsf{VRF}}$.
4. The block is signed using sk_o.
5. The VRF output y_o, π_{y_o} is valid with respect to public key $\mathsf{vk}_{\mathrm{VRF},o}$ and input $\eta_e || \mathsf{sl}_r$ (i.e., $\mathsf{VRF.Verify}(\mathsf{vk}_{\mathrm{VRF},o}, \eta_e || \mathsf{sl}_r, y_o, \pi_{y_o}) \to 1$).
6. The stake value of o is $v_o \geq T$, such that $y_o < 2^{l_{\mathsf{VRF}}} \phi_f(\frac{T}{v_{\mathsf{total}}})$.
7. o was unspent in the snapshot (or rather that the key image for o is not in $\mathbb{I}_e$).

Protocol StakeNote

This protocol is run by every party $\mathcal{P}$ on every epoch e with snapshot sets $\mathbb{O}_e, \mathbb{I}_e$ (UTxOs and key images in blocks up to epoch $e-2$, respectively) and random nonce η_e (assumed to be generated as in Ouroboros Praos). A party controls an output $o \in \mathbb{O}_e$ if it knows the secret key $\mathsf{sk} = (\mathsf{sk}_o, v_o, r_o)$ connected to the output.

- **Setup:** Parties have access to an initialization functionality $\mathcal{F}_{\mathsf{INIT}}$, which holds system parameters p for our modified CLSAG, the (confidential) initial stake distribution and the genesis block containing CLSAG-authenticated RingCT transactions following this stake distribution. In the beginning of the execution, every party $\mathcal{P}$ calls $\mathcal{F}_{\mathsf{INIT}}$ with $(\mathsf{init}, \mathcal{P})$, receiving as response parameters p, the genesis block B_0 and the secret key $\mathsf{sk} = (\mathsf{sk}_{\mathcal{P}}, v_{\mathcal{P}}, r_{\mathcal{P}})$ corresponding to a UTxO $(\mathsf{sk}_{\mathcal{P}} \cdot G_{\mathsf{PAY}}, v_{\mathcal{P}} \cdot G_1 + r_{\mathcal{P}} \cdot G_2)$ contained in B_0.
- **Block Creation:** On a slot sl_r, for every unspent output o in the epoch's snapshot $\mathbb{O}_e$ that party $\mathcal{P}$ controls, it proceeds as follows:
  1. Compute $\mathsf{VRF.Eval}(\mathsf{sk}_o, \eta_e || \mathsf{sl}_r) \to (y_o, \pi_{y_o})$.
  2. If $y_o < 2^{l_{\mathsf{VRF}}} \phi_f(\frac{v_o}{v_{\mathsf{total}}})$, o is eligible for creating a block on slot sl_r, so $\mathcal{P}$ proceeds. Otherwise, $\mathcal{P}$ ignores the next steps, advancing to the next $o \in \mathbb{O}_e$.
  3. Set the block payload B (transactions and metadata, e.g., timestamps, etc).
  4. Determine the minimum T such that $v_o \geq T$ and $y_o < 2^{l_{\mathsf{VRF}}} \phi_f(\frac{T}{v_{\mathsf{total}}})$.
  5. Sample $r' \xleftarrow{\$} \mathbb{Z}_q$ and compute commitment $\mathcal{C} = (v_o - T) \cdot G_1 + r' \cdot G_2$.
  6. Generate a range proof[a] π_{RANGE} that commitment $\mathcal{C}$ contains $(v_o - T) \geq 0$.
  7. Set block contents $\mathcal{M} = \langle B, e, \mathsf{sl}_r, (\mathsf{vk}_{\mathrm{VRF},o}, y_o, \pi_{y_o}), T, \mathcal{C}, \pi_{\mathsf{RANGE}}, \Im_o \rangle$, where $\Im_o$ is the key image $\Im_o = \mathsf{sk}_o \cdot H(\mathsf{vk}_o)$ for o and $\mathsf{vk}_{\mathrm{VRF},o} = \mathsf{sk}_o \cdot G_{\mathsf{VRF}}$.
  8. Sample a random subset of outputs $\mathbb{O}_{\mathsf{block}} \subseteq \mathbb{O}_e$ such that $o \in \mathbb{O}_{\mathsf{block}}$ and generate a modified CLSAG [23] (Section 3.2) $\mathsf{Sign}(L_{\mathsf{SN}}, (\mathsf{sk}_o, v_o, r_o, r'), (T, \mathcal{C}, \mathsf{vk}_{\mathrm{VRF},o}, \Im_o, \mathbb{O}_{\mathsf{block}}), \mathcal{M}) \to \sigma_{\mathsf{CLSAG}}$.
  9. Diffuse the final block $\langle \mathcal{M}, \mathbb{O}_{\mathsf{block}}, \sigma_{\mathsf{CLSAG}} \rangle$ in the network.
- **Block Verification:** For every block $\langle \mathcal{M}, \mathbb{O}_{\mathsf{block}}, \sigma_{\mathsf{CLSAG}} \rangle$ received in epoch e, a full node proceeds as follows:
  1. Parse $\mathcal{M} = \langle B, e, \mathsf{sl}_r, (\mathsf{vk}_{\mathrm{VRF},o}, y_o, \pi_{y_o}), T, \mathcal{C}, \pi_{\mathsf{RANGE}}, \Im_o \rangle$ and check that it extends the local chain according to the Ouroboros Genesis chain rule.
  2. Check that $\mathbb{O}_{\mathsf{block}} \subseteq \mathbb{O}_e$ and that $\Im_o \notin \mathbb{I}_e$ (i.e., $\mathbb{O}_{\mathsf{block}}$ is in snapshot $\mathbb{O}_e$ and the UTxO used to create the block was unspent according to $\Im_o$ and $\mathbb{I}_e$).
  3. Verify π_{RANGE} using the range proof verifier (i.e., check that $T < v_o$).
  4. Check that $\mathsf{VRF.Verify}(\mathsf{vk}_{\mathrm{VRF},o}, \eta_e || \mathsf{sl}_r, y_o, \pi_{y_o}) \to 1$ and that $y_o < 2^{l_{\mathsf{VRF}}} \phi_f(T)$ (i.e., v_o is high enough for the leader election to be valid).
  5. Check that $\mathsf{Verify}(L_{\mathsf{SN}}, (T, \mathcal{C}, \mathsf{vk}_{\mathrm{VRF},o}, \Im_o, \mathbb{O}_{\mathsf{block}}), \mathcal{M}, \sigma_{\mathsf{CLSAG}}) \to 1$ (i.e., $\mathsf{vk}_{\mathrm{VRF},o}$, $\mathcal{C}$ and $\Im_o$ are valid w.r.t. a witness $(\mathsf{sk}_o, v_o, r_o, r')$ corresponding to an UTxO $o \in \mathbb{O}_{\mathsf{block}}$ and T, and B is signed under this witness).
  6. Accept the block into the local chain if all checks pass.

[a] For example, this can be done using a Bulletproof [4]

Figure 1 Protocol StakeNote.

Revealing only $\mathbb{O}_{\mathsf{block}}$ but not o maintains the same anonymity guarantees as the original CryptoNote protocol. Proving the statements above ensures that the distribution of slot leaders (parties who create blocks) remains the same as in Ouroboros Praos, which allows us in principle to inherit the consensus security guarantees from that PoS protocol.

Statements 1, 2, 3 and 4 are proven via our modified CLSAG [23] (Section 3.2). Informally, σ_{CLSAG} contains a new commitment $\mathcal{C}$ and proves knowledge of private values sk_o, v_o, r_o for an output $o \in \mathbb{O}_{\mathsf{block}}$ such that the following statements hold without revealing o or sk_o, v_o, r_o:

1. the VRF public key is $\mathsf{vk}_{\mathrm{VRF},o} = \mathsf{sk}_o \cdot G_{\mathsf{VRF}}$;
2. $\mathcal{C}$ is a commitment to the output's stake v_o minus the value T;
3. the output's key image is $\Im_o = \mathsf{sk}_o \cdot H(\mathsf{vk}_o)$.

Statement 5 is trivially proven by checking that $\mathsf{Verify}(\mathsf{vk}_{\mathrm{VRF},o}, x, y_o, \pi_{y_o}) \to 1$ given that $\mathsf{vk}_{\mathrm{VRF},o}$ is proven to be valid by σ_{CLSAG}, that y_o, π_{y_o} are revealed in the block and that $\eta_e || \mathsf{sl}_r$ is publicly known.

Statement 6 is proven leveraging the commitment $\mathcal{C}$ in σ_{CLSAG} and the range proof π_{RANGE}, which proves that $\mathcal{C}$ is a commitment to a positive value $v_o - T > 0$. This helps us establish that the value T revealed with the block is not larger than the actual value of o, meaning that $y_o < 2^{l_{\mathsf{VRF}}} \phi_f(\frac{T}{v_{\mathsf{total}}})$ can be trivially checked in public given that y_o is valid as established before.

Finally, statement 7 is proven by revealing the key image $\Im_o$ for o, allowing for publicly checking that $\Im_o \notin \mathbb{I}_e$. A caveat of this approach is that it is possible to link different blocks generated using the same o.

Block verification

Block verification follows easily by checking all of the statements proven as part of block creation. This requires ensuring that $\mathbb{O}_{\mathsf{block}} \subseteq \mathbb{O}_e$ and that $\Im_o \notin \mathbb{I}_e$, verifying the modified CLSAG σ_{CLSAG}, verifying the range proof π_{RANGE}, verifying the VRF output y_o, π_{y_o} and ensuring that $y_o < 2^{l_{\mathsf{VRF}}} \phi_f(T)$.

3.2 Block's CLSAG signature

We now describe our construction's main signature scheme, which is built using the ideas of CLSAG [23]. This primitive allows us to prove that a slot leader has been elected due to controlling a UTxO o (in the snapshot), with value v_o, without revealing the exact UTxO. Essentially, the leader proves that the UTxO is unspent and that the value is "high enough", while keeping it private.

We will present this primitive as a *signature of knowledge* [6], an approach commonly used to construct ring signatures. Instead of proving that a message was authenticated by the owner of a given secret key as in a regular digital signature, a signature of knowledge allows for proving that a message is authenticated by a party who *knows* a witness w for a statement $w \in L$ for *any NP language* L. A signature of knowledge is composed of the following algorithms:

- $\mathsf{Setup}(1^\lambda)$, which outputs parameters p implicitly taken as input by the other algorithms;
- $\mathsf{Sign}(L, w, x, m)$, which outputs a signature σ that is valid if w is a valid witness for a statement $x \in L$ for a NP language L.
- $\mathsf{Verify}(L, x, m, \sigma) \to \{0, 1\}$, which outputs 1 if and only if σ was generated using a valid witness w showing that $x \in L$.

Intuitively, the statement we are trying to prove can be imagined as a matrix of n rows and 4 columns, comprising an OR statement across rows and an AND statement across columns. Each row corresponds to one of the outputs in $\mathbb{O}_{\mathsf{block}}$, and each column (also often called a "layer") corresponds to a value of interest of which the party proves knowledge: (i) column 1 proves knowledge of sk_o corresponding to the payment key sk_o; (ii) column 2 proves that

the VRF public key $\mathsf{vk}_{\mathrm{VRF},o}$ was computed from the same sk_o; (iii) column 3 proves that the key image $\Im_o$ was computed from the same sk_o; (iv) column 4 proves knowledge of the difference between the output's value and the revealed threshold and of the randomness used to compute a commitment $\mathcal{C}$ to this difference.

Formally, we are interested in statements in the language L_{SN} defined below.

▶ **Definition 2** (Language L_{SN}). *Let* $(w, x) \in L_{\mathsf{SN}}$. *It holds that* $w = (\mathsf{sk}_o, v_o, r_o, r')$ *and* $x = (T, \mathcal{C}, \mathsf{vk}_{VRF,o}, \Im_o, \mathbb{O}_{\mathsf{block}})$, *where* $\mathbb{O}_{\mathsf{block}} = \langle(\mathsf{vk}_0, C_0), \ldots, (\mathsf{vk}_{n-1}, C_{n-1})\rangle$, *such that:*

$$\bigvee_{i=0,\ldots,n-1} \Big(\mathsf{vk}_i = \mathsf{sk}_o \cdot G_{\mathsf{PAY}} \wedge \mathsf{vk}_{VRF,o} = \mathsf{sk}_o \cdot G_{\mathsf{VRF}} \wedge \Im_o = \mathsf{sk}_o \cdot H(\mathsf{vk}_i) \wedge (r_o - r') \cdot G_2 = C_i - T \cdot G_1 - \mathcal{C}\Big).$$

Modified CLSAG Scheme

Our signature scheme uses a random oracle H_{clsag} with codomain F_q and consists of the tuple $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{p}$: sample a cyclic group $\mathbb{G}$ of order q, generators $G_1, G_2, G_{\mathsf{PAY}}, G_{\mathsf{VRF}} \xleftarrow{\$} \mathbb{G}$, outputting $\mathsf{p} = (q, G_1, G_2, G_{\mathsf{PAY}}, G_{\mathsf{VRF}})$, which is implicitly taken as input by all algorithms.
- $\mathsf{Sign}(L_{\mathsf{SN}}, w, x, \mathcal{M}) \to \sigma$:
  1. Parse w as $(\mathsf{sk}_o, v_o, r_o, r')$, x as $(T, \mathcal{C}, \mathsf{vk}_{\mathrm{VRF},o}, \Im_o, \mathbb{O}_{\mathsf{block}})$ and $\mathbb{O}_{\mathsf{block}}$ as $\langle(\mathsf{vk}_0, C_0), \ldots, (\mathsf{vk}_{n-1}, C_{n-1})\rangle$.
  2. Find the index k, such that $\mathsf{vk}_k = (\mathsf{sk}_o \cdot G_{\mathsf{PAY}}, v_o \cdot G_1 + r_o \cdot G_2)$.
  3. Sample $(\alpha_x, \alpha_r) \xleftarrow{\$} (\mathsf{F}_p)^2$ and compute:
     - $L_k = \alpha_x \cdot G_{\mathsf{PAY}}$, $V_k = \alpha_x \cdot G_{\mathsf{VRF}}$, $I_k = \alpha_x \cdot H(\mathsf{vk}_o)$
     - $A_k = \alpha_r \cdot G_2$
     - $c_{(k+1) \mod n} = H_{\mathsf{clsag}}(\mathcal{M}||k||L_k||V_k||I_k||A_k)$
  4. Set $i = (k+1) \mod n$ and proceed as follows until $i = k$:
     a. Sample $(s_i^{(x)}, s_i^{(r)}) \xleftarrow{\$} (\mathsf{F}_p)^2$ and compute:
        - $L_i = s_i^{(x)} \cdot G_{\mathsf{PAY}} + c_i \cdot \mathsf{vk}_i$, $V_i = s_i^{(x)} \cdot G_{\mathsf{VRF}} + c_i \cdot \mathsf{vk}_{\mathrm{VRF},o}$, $I_i = s_i^{(x)} \cdot H(\mathsf{vk}_i) + c_i \cdot \Im_o$
        - $A_i = s_i^{(r)} \cdot G_2 + c_i \cdot D_i$, where $D_i = C_i - T \cdot G_1 - \mathcal{C}$.[a]
        - $c_{i+1 \mod n} = H_{\mathsf{clsag}}(\mathcal{M}||i||L_i||V_i||I_i||A_i)$
     b. Set $i = (i+1) \mod n$.
  5. Compute $s_k^{(x)} = \alpha_x - c_k \cdot \mathsf{sk}_o$ and $s_k^{(r)} = \alpha_r - c_k \cdot (r_o - r')$.
  6. Output $\sigma_{\mathsf{CLSAG}} = \langle c_0, (s_0^{(x)}, s_0^{(r)}), \ldots, (s_{n-1}^{(x)}, s_{n-1}^{(r)})\rangle$.
- $\mathsf{Verify}(L_{\mathsf{SN}}, x, \mathcal{M}, \sigma) \to \{0, 1\}$:
  1. Parse x as $(T, \mathcal{C}, \mathsf{vk}_{\mathrm{VRF},o}, \Im_o, \mathbb{O}_{\mathsf{block}})$, $\mathbb{O}_{\mathsf{block}}$ as $\langle(\mathsf{vk}_0, C_0), \ldots, (\mathsf{vk}_{n-1}, C_{n-1})\rangle$ and σ_{CLSAG} as $\langle c_0, (s_0^{(x)}, s_0^{(r)}), \ldots, (s_{n-1}^{(x)}, s_{n-1}^{(r)})\rangle$.
  2. Set $c'_0 = c_0$ and, for every $i \in [0, \ldots, n-1]$, compute:
     - $L_i = s_i^{(x)} \cdot G_{\mathsf{PAY}} + c'_i \cdot \mathsf{vk}_i$, $V_i = s_i^{(x)} \cdot G_{\mathsf{VRF}} + c'_i \cdot \mathsf{vk}_{\mathrm{VRF},o}$, $I_i = s_i^{(x)} \cdot H(\mathsf{vk}_i) + c'_i \cdot \Im_o$
     - $A_i = s_i^{(r)} \cdot G_2 + c'_i \cdot D_i$, where $D_i = C_i - T \cdot G_1 - \mathcal{C}$
     - $c'_{(i+1)} = H_{\mathsf{clsag}}(\mathcal{M}||i||L_i||V_i||I_i||A_i)$
  3. If $c_0 = c'_n$, output 1. Otherwise, output 0.

[a] Observe that: $D_k = C_o - T \cdot G_1 - \mathcal{C} = (v_o - T - v_o + T) \cdot G_1 + (r_o - r') \cdot G_2 = (r_o - r') \cdot G_2$.

Figure 2 Modified CLSAG Scheme.

Notice that a signature of knowledge of a witness for a statement in L_{SN} is a ring signature for the ring $\langle(\mathsf{vk}_0, C_0), \ldots, (\mathsf{vk}_{n-1}, C_{n-1})\rangle$, as it does not reveal to which (vk_i, C_i) the witness $(\mathsf{sk}_o, v_o, r_o, r')$ corresponds.

Our modified CLSAG scheme is presented in Figure 2. It is constructed using standard Schnorr-style sigma protocols for equality and knowledge of discrete logarithms. The proof is turned into a signature of knowledge via the Fiat-Shamir transformation [17] along with the standard OR-composition technique [9] for proving disjunctions among several statements.

3.3 Security Analysis

We now analyze the security of StakeNote, arguing that it achieves the same security guarantees as Ouroboros Praos, although against a weaker delayed adaptive adversary. First, Lemma 3 establishes that the security of the VRF and modified CLSAG schemes is maintained, even though we reuse the same secret key sk_o to compute the public payment key $\mathsf{vk}_o = \mathsf{sk}_o \cdot G_{\mathsf{PAY}}$, the VRF public key $\mathsf{vk}_{\mathrm{VRF},o} = \mathsf{sk}_o \cdot G_{\mathsf{VRF}}$, and the key image $\Im_o = \mathsf{sk}_o \cdot H(\mathsf{vk}_o)$.

▶ **Lemma 3.** *Let $G_{\mathsf{PAY}}, G_{\mathsf{VRF}}, H(\mathsf{vk}_o)$ be generators of a cyclic group $\mathbb{G}$ of order q. The tuples $\mathbf{T_0} = (G_{\mathsf{PAY}}, G_{\mathsf{VRF}}, H(\mathsf{vk}_o), \mathsf{vk}_o = \mathsf{sk}_o \cdot G_{\mathsf{PAY}}, \mathsf{vk}_{VRF,o} = \mathsf{sk}_o \cdot G_{\mathsf{VRF}}, \Im_o = \mathsf{sk}_o \cdot H(\mathsf{vk}_o))$ where $\mathsf{sk}_o \xleftarrow{\$} \mathbb{Z}_q$ and $\mathbf{T_1} = (G_{\mathsf{PAY}}, G_{\mathsf{VRF}}, H(\mathsf{vk}_o), \mathsf{vk}_o = \mathsf{sk}_o \cdot G_{\mathsf{PAY}}, \mathsf{vk}_{VRF,o} = \mathsf{sk}_{\mathsf{VRF}} \cdot G_{\mathsf{VRF}}, \Im_o = \mathsf{sk}_{\Im} \cdot H(\mathsf{vk}_o))$ where $\mathsf{sk}_o, \mathsf{sk}_{\mathsf{VRF}}, \mathsf{sk}_{\Im} \xleftarrow{\$} \mathbb{Z}_q$ are computationally indistinguishable.*

Proof. The proof directly follows from the ℓ-DDH Hardness Assumption 1. Given black-box access to an adversary $\mathcal{A}$ that distinguishes $\mathbf{T_0}, \mathbf{T_1}$, we can construct an adversary $\mathcal{A}_{\ell-DDH}$ that solves the ℓ-DDH problem with the same probability.

First, notice that $\mathbf{T_0}$ is distributed as a tuple from D_0 in the ℓ-DDH problem:

$$\mathbf{x_0} := (G_{\mathsf{PAY}}, \mathsf{sk}_o \cdot G_{\mathsf{PAY}}, G_{\mathsf{VRF}} = \beta_0 \cdot G_{\mathsf{PAY}}, H(\mathsf{vk}_o) = \beta_1 \cdot G_{\mathsf{PAY}}, (\mathsf{sk}_o \cdot \beta_0) \cdot G_{\mathsf{PAY}}, (\mathsf{sk}_o \cdot \beta_1) \cdot G_{\mathsf{PAY}})$$

Also, $\mathbf{T_1}$ is distributed as a tuple from D_1 in the ℓ-DDH problem:

$$\mathbf{x_1} := (G_{\mathsf{PAY}}, \mathsf{sk}_o \cdot G_{\mathsf{PAY}}, G_{\mathsf{VRF}} = \beta_0 \cdot G_{\mathsf{PAY}}, H(\mathsf{vk}_o) = \beta_1 \cdot G_{\mathsf{PAY}}, \mathsf{sk}_{\mathsf{VRF}} \cdot G_{\mathsf{PAY}}, \mathsf{sk}_{\Im} \cdot G_{\mathsf{PAY}})$$

$\mathcal{A}_{\ell-DDH}$, uses this correspondence to ℓ-DDH tuples to construct $\mathbf{T_0}, \mathbf{T_1}$ tuples that it inputs to $\mathcal{A}$. Upon receiving a challenge $\mathbf{x} := (G, \alpha \cdot G, \beta_0 \cdot G, \beta_1 \cdot G, \gamma_0 \cdot G, \gamma_1 \cdot G)$, $\mathcal{A}_{\ell-DDH}$ constructs $\mathcal{A}$'s input as follows:

$$(G_{\mathsf{PAY}} = G, G_{\mathsf{VRF}} = \beta_0 \cdot G, H(\mathsf{vk}_o) = \beta_1 \cdot G, \mathsf{vk}_o = \alpha \cdot G, \mathsf{vk}_{\mathrm{VRF},o} = \gamma_0 \cdot G, \Im_o = \gamma_1 \cdot G)$$

$\mathcal{A}_{\ell-DDH}$ outputs whatever $\mathcal{A}$ outputs. Since the input given to $\mathcal{A}$ is distributed exactly as $\mathbf{T_0}$ when the challenge comes from D_0 and exactly as $\mathbf{T_1}$ when the challenge comes from D_1, $\mathcal{A}_{\ell-DDH}$ has the same advantage as $\mathcal{A}$. ◀

We now show Lemma 4, establishing that parties are elected to generate a block in StakeNote with the same probability as in Ouroboros Praos. This is crucial when later arguing security of StakeNote.

▶ **Lemma 4.** *In Protocol StakeNote, for every slot sl_r of an epoch e, a party that controls stake v distributed over multiple UTxOs $o_1, \ldots, o_\ell$ with respective amounts $v_{o_1}, \ldots, v_{o_n}$ in the snapshot $\mathbb{O}_e$ for epoch e such that $v = v_{o_1} + \cdots + v_{o_n}$ is considered eligible to generate a block (i.e., passes the check in Step 2 of StakeNote's block generation phase) with the same probability as in Ouroboros Praos.*

Proof. First, notice that the epoch random nonce η_e is generated exactly as in Ouroboros Praos and that the eligibility test checking that $y_o < 2^{l_{\mathsf{VRF}}} \phi_f(\frac{v_o}{v_{\mathsf{total}}})$ for $\mathsf{VRF.Eval}(\mathsf{sk}_o, \eta_e || \mathsf{sl}_r) \to (y_o, \pi_{y_o})$ is exactly the same eligibility test as in Ouroboros Praos. Moreover, notice that

the epoch snapshot $\mathbb{O}_e$ for epoch e considers the same set of UTxOs as Ouroboros Praos for epoch e. The only difference in the setup for eligibility check, since the VRF public key $\mathsf{vk}_{\mathrm{VRF},o} = \mathsf{sk}_o \cdot G_{\mathsf{VRF}}$ is such that it depends on $\mathsf{vk}_o = \mathsf{sk}_o \cdot G_{\mathsf{PAY}}$ and $\Im_o = \mathsf{sk}_o \cdot H(\mathsf{vk}_o)$. However, as established in Lemma 3, this does not introduce any difference in relation to the Ouroboros Praos setup where $\mathsf{vk}_{\mathrm{VRF},o} = \mathsf{sk}_{\mathsf{VRF}} \cdot G_{\mathsf{VRF}}$ is generated from an independently sampled uniformly random secret key $\mathsf{sk}_{\mathsf{VRF}}$.

The crucial detail establishing that the probability of being eligible for generating a block in slot sl_r of epoch e in StakeNote is the same as in Ouroboros Praos is the *aggregation independence* property of function $\phi_f(\frac{v_o}{v_{\mathsf{total}}})$. This property is central in the original security proof of Ouroboros Praos, and guarantees that a party controlling total stake v according to the snapshot $\mathbb{O}_e$ is elected to generate a block in every slot sl_r with the same probability regardless of how v is distributed over multiple UTxOs $o_1, \ldots, o_\ell$ with respective amounts $v_{o_1}, \ldots, v_{o_n}$ such that $v = v_{o_1} + \cdots + v_{o_n}$. Hence, since the eligibility check is performed exactly as in Ouroboros Praos considering epoch random nonces generated exactly in the same way and exactly the same set of UTxOs in the snapshot, the party is considered eligible to generate a block for every slot with exactly the same probability regardless of how this party's stake is distributed over multiple UTxOs. ◀

Next, Lemma 5 analyzes the security of the modified CLSAG construction (Figure 2). Although our construction is similar to the CLSAG of [23], we state it as a signature of knowledge, since the SimExt-Security [6] guarantees of such schemes will make it easier to analyze our protocol's security in the GKL model [21]. However, our construction also retains the privacy guarantees offered by the original CLSAG. In particular, the simulation soundness implied by SimExt-Security implies that no information about the witness is leaked, hiding the exact UTxO used to generate the signature within the ring.

▶ **Lemma 5.** *The modified CLSAG construction of Figure 2 is a signature of knowledge for language L_{SN} (Definition 2) with SimExt-Security [6] under the ℓ-DDH assumption in the random oracle model.*

Proof (Sketch). In order to show that the construction Figure 2 is a signature of knowledge with SimExt-Security as defined in [6], we must show that it is correct, construct a simulator that generates valid signatures without knowing a witness and construct an extractor that obtains a witness from a valid signature. Correctness follows by inspection, so we will focus on establishing that it is possible to construct a simulator and an extractor for our construction.

We observe that our construction is essentially an OR-composition following the techniques of [9] of the Chaum-Pedersen sigma protocol for discrete logarithm equality [7] (used to prove that sk_o is used to correctly compute the VRF public key and key image) and of the Schnorr proof of knowledge of discrete logarithms (used to prove knowledge of $r_o - r'$). It is clear that both of these sigma protocols have the completeness, honest verifier zero knowledge and special soundness properties [10], meaning that they can be used to obtain a zero knowledge proof system in the random oracle model via the Fiat-Shamir transform. At the core of this construction is a Fiat-Shamir transform where the message to be signed is given as input to a random oracle when computing the challenge for the underlying sigma protocols.

In order to construct a simulator and an extractor for our scheme, we can leverage the techniques from [15]. As shown in [15], zero knowledge proof systems obtained via the Fiat-Shamir transform from sigma protocols are both simulation sound and weakly simulation extractable. The generic construction of a simulator in the simulation soundness proof for the Fiat-Shamir transform presented in [15] can be directly applied to our construction. The caveat in their proof of the simulation extractability property is that the extractor must

rewind a black-box copy of the adversary, which is not allowed in the strong version of simulation extrability. However, in this case, we can rewind the adversary when arguing security of our scheme, so we can use the generic extractor construction of [15] in order to obtain an extractor for our modified CLSAG. ◀

To analyze the security of StakeNote (Figure 1), we show that the leader election process determining the parties who generate blocks is computationally indistinguishable from Ouroboros Praos's. Hence, the block distribution generated by corrupted or honest parties in our protocol is essentially the same as Ouroboros Praos. Establishing this fact will allow us to inherit the security properties of Ouroboros Praos under the same assumptions (Theorem 6).

▶ **Theorem 6.** *Let* $\langle \mathsf{VRF.KeyGen}, \mathsf{VRF.Eval}, \mathsf{VRF.Verify} \rangle$ *be the Verifiable Random Function with unpredictability under malicious key generation from Ouroboros Praos [11]. Let* $(\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ *be a signature of knowledge for the language* L_{SN}*. StakeNote (cf. Figure 1) satisfies the common prefix, chain growth and chain quality properties (resp. liveness and persistence guarantees) with the same parameters as established in [11, Theorems 7, 8, 9] (resp. [11, Theorem 11]) under the* ℓ*-DDH assumption in the random oracle model with security against a delayed adaptive adversary with corruption delay equal to epoch length over a semi-synchronous network.*

Proof (Sketch). The core argument is that StakeNote elects leaders to generate blocks with the same distribution as in Ouroboros Praos. In the terminology of Ouroboros Praos, we are interested in the characteristic strings denoting empty slots, slots with blocks generated by honest parties and slots with blocks generated by corrupted parties. Each execution is characterized by such a characteristic string according to the blocks that were produced, and the security of Ouroboros Praos is established by proving statements about characteristic strings. Hence, we want to show that the characteristic strings arising from executions of StakeNote are distributed exactly as characteristic strings arising from executions of Ouroboros Praos, except with negligible probability.

First, we observe that the randomness for leader election in StakeNote is generated exactly as in Ouroboros Praos, i.e., by having slot leaders of an epoch execute the VRF-based leaky resettable beacon protocol of Ouroboros Praos in order to generate the randomnes for the next epoch. Moreover, the oblivious leader election scheme adopted by StakeNote is very similar to the one adopted by Ouroboros Praos. A party determines if it is a slot leader by evaluating the VRF on the epoch's randomness concatened with the slot number, then checking if this output is smaller than a function ϕ of the relative stake represented by the amount of each of their UTxOs, where ϕ is defined exactly as in Ouroboros Praos. There are two main differences in StakeNote: 1. a party's total stake (and thus their total relative stake) is unknown and distributed over many different UTxOs, rather than publicly known as in Ouroboros Praos; 2. a slot leader does not reveal the exact UTxO used in leader election (nor its exact value) when proving that they were elected a leader, instead publishing a signature of knowledge showing that a VRF public key (to be used in verifying the proof of leader election) has been correctly computed from a UTxO within a set of UTxOs. We must show that these differences do not affect the distribution of characteristic strings.

As formally established in Lemma 4, the first difference does not affect the distribution of characteristic strings due to the "independent aggregation" property of the ϕ function from Ouroboros Praos. This property ensures that a party gets elected with the same probability regardless of whether their stake is aggregated into a single UTxO or arbitrarily distributed across multiple UTxOs. Hence, even though the leader election check is executed considering the individual amount of each UTxO controlled by a party, that party is guaranteed to be elected with the same probability as if its entire stake was aggregated into a single UTxO.

The second difference between StakeNote and Ouroboros Praos is trickier to analyse, since parties cannot simply verify that a slot leader has been correctly elected as in Praos, as that would require learning the relative stake controlled by the slot leader as well as their VRF public key. Instead, in the case of StakeNote, parties learn a VRF public key attached to the block and a stake *threshold* T that must be guaranteed to respect $T \leq v_o$, where v_o is the stake of the UTxO used to execute the leader election check. Notice that this verification will work as expected (and produce the same results as Ouroboros Praos) as long as the VRF public key and threshold T contained in the block are correctly generated from the secret ket sk_o connected to the UTxO while respecting the condition that $T \leq v_o$. The correctness of the VRF public key is guaranteed by the simulation extractability property of the modified CLSAG construction established in Lemma 5. The fact that $T \leq v_o$ is also guaranteed by the simulation extractability of the modified CLSAG construction, which ensures that the commitment $\mathcal{C}$ is indeed a commitment to $v_o - T$, and by the soundness of the range proof, which guarantees that the value of $\mathcal{C}$ is above zero (i.e., $v_o - T > 0$ implies that $T \leq v_o$).

The last difference between StakeNote and Ouroboros Praos left to analyze is the lack of forward security in the modified CLSAG signature, whereas Praos utilizes a key evolving signature scheme. The lack of forward security means that StakeNote does not achieve security against fully adaptive adversaries. However, if the adversary is only delayed adaptive with corruption delay equal to epoch duration (as we consider), an adversary corrupting a party in one epoch only gains control in the next epoch. Hence, if the adversary requests to corrupt a slot leader who just generated a block, it only gains control of that party in the next epoch, and does not succeed in generating a conflicting version of that block.

We have thus established that, under the assumptions stated in the theorem, the characteristic strings arising from executions of StakeNote are distributed exactly as characteristic strings of executions of Ouroboros Praos, except with negligible probability. Hence, StakeNote achieves the same properties as Ouroboros Praos with the same parameters. ◀

3.4 Discussion

CLSAG. Our CLSAG proof σ_{CLSAG} satisfies two different purposes. First, as is the original purpose of CLSAG, the block producer proves that they control one of the outputs in the group. Note that, in our case, linkability is not needed since we allow the same private key to produce multiple messages, that is multiple blocks per epoch. Second, it offers a discrete logarithm equality proof, between vk_o and $\mathsf{vk}_{\mathrm{VRF},o}$. This is achieved by CLSAG's elegant idea of reusing the same response $s_i^{(x)}$ for the two columns, that correspond to vk_o and $\mathsf{vk}_{\mathrm{VRF},o}$, effectively showing that the secret of both columns is the same.

Key Image Reuse. With our mechanism, a UTxO's key image is published multiple times: (i) every time a block is created using the corresponding UTxO; (ii) when the UTxO is spent. No action should reveal the exact UTxO which corresponds to the revealed key image. Therefore, for each of these actions, the party should use an anonymity set of decoys. It is easy to see that, if these sets are different, then the correct UTxO is in their intersection. Therefore, to retain the highest possible level of anonymity, the user should use the same anonymity set at all times. Essentially, the user picks the anonymity set of decoys once, the first time that it publishes the key image, and reuses it for every future action. Note that this does not reduce the anonymity guarantees of the original CryptoNote protocol, since the set of outputs, from which the decoys are picked, still consists of all historical outputs.

Block Linkability. In our construction, each block is associated with the key image of the output used for its creation. Since this key image is deterministic for each output, all blocks that are created using the same output are publicly linkable. In other words, everyone can deduce that the same output was used for the creation of all blocks that share the same key image, albeit the precise output is hidden within the anonymity set.

4 Implementation

We evaluate StakeNote with a proof of concept implementation in Rust.[7] Our implementation uses the ristretto255 group on the Curve25519 elliptic curve. Specifically, the VRF implementation uses the `vrf_r255` library,[8] which implements the ECVRF-RISTRETTO255-SHA512 ciphersuite.[9] The CLSAG implementation uses the ristretto255 implementation of the `curve25519_dalek` library.[10] The range proof is constructed with the `bulletproofs` library,[11] which implements Bulletproofs [4] using ristretto255.

Our evaluation focuses around two main questions:

1. What is the time and size performance of StakeNote?
2. What is the output's amount leakage, due to revealing the threshold T?

4.1 Performance

We evaluated the performance of our mechanism by running 10,000 block creation instances, each simulating the leader election process. First, we created an output with amount of stake between $5 - 45\%$ of the total stake, as well as a set of outputs to be used as decoys. Second, we created a VRF key pair, which shares the same secret key as the chosen output. Third, we picked a random value as the epoch's nonce η_e and iterated over a counter i, which simulated the slot number until the VRF output is below the eligibility threshold.

Table 1 Timing evaluation of StakeNote.

Component	Creation			Verification		
	Min (ms)	Max (ms)	Avg (ms)	Min (ms)	Max (ms)	Avg (ms)
VRF output	0.369	2.224	0.428	0.22	1.64	0.27
T estimation	0.001	0.066	0.004	-	-	-
Range proof	14.40	61.68	16.53	3.26	19.69	3.95
CLSAG proof	7.04	36.32	8.10	7.42	35.28	8.48
Total	21.44	98.07	24.63	10.68	54.97	12.43

Time Performance. All timing measurements are shown in Table 1.

During block creation, we first generate the VRF output and proof, which requires between $0.369 - 2.224$ ms, at an average time of 0.428 ms. Second, we perform a binary search to find the smallest threshold T (with 180 precision decimal points); this is essentially instantaneous (less than 0.1 ms). Third, we compute the range proof; this is the slowest

[7] `https://github.com/commonprefix/stakenote`
[8] `https://docs.rs/vrf-r255/latest/vrf_r255`
[9] `https://github.com/C2SP/C2SP/blob/main/vrf-r255.md`
[10] `https://docs.rs/curve25519-dalek/latest/curve25519_dalek/`
[11] `https://docs.rs/bulletproofs/latest/bulletproofs`

computation, between $14.40 - 61.68$ ms, with average proof creation time 16.53 ms. Fourth, we compute the CLSAG proof, which lasts between $7.04 - 36.32$ ms with an average time of 8.10 ms. In total, block creation lasts between $21.44 - 98.07$ ms (average 24.63 ms).

Block verification is significantly faster. VRF verification lasts between $0.22 - 1.64$ ms, with the average being 0.27 ms. Verification of the range proof lasts between $3.26 - 19.69$ ms, at an average time of 3.95 ms. CLSAG verification is slower, requiring between $7.42 - 35.28$ ms with an average time being 8.48 ms.

As the average values suggest, block creation typically lasts less than 25 ms and verification less than 15 ms. The evaluations took place on home equipment, so timings occasionally increase e.g., during increased load to the CPU, although never exceeding 100 ms for creation and 55 ms for verification. We also note that our code is not optimized beyond using Rust's standard "release" compiling profile.

Table 2 Proof size evaluation of StakeNote with ring size n.

Component		Size (bytes)
VRF	Public key	32
	Output	128
	Proof	80
Range proof	Threshold value T	8
	Bulletproof	672
CLSAG	Key Image	32
	$(v_o - T)$ commitment	32
	c_0	32
	$s_i^{(x)}$ *(per output in the ring)*	32
	$s_i^{(r)}$ *(per output in the ring)*	32
	public key *(per output in the ring)*	32
	amount commitment *(per output in the ring)*	32
Total		$1{,}064 + n \cdot 128$

Size Performance. The size measurements are shown in Table 2.

Assuming the CLSAG ring size is n, the block consists of the following: (i) $2 + 2 \cdot n$ elliptic curve points; (ii) $2 + 2 \cdot n$ scalars; (iii) one 256-bit number; (iv) VRF output and proof; (v) range proof. Naturally, these depend on the primitives' implementation, so the following are reference values.

Regarding the VRF elements, the public key is 32 bytes, the output is 128 bytes, and the proof is 80 bytes. The range proof is again the heaviest element, requiring 672 bytes. The value T is 8 bytes, the key image is 32 bytes, and the commitment (to the difference between the output's amount and T) is 32 bytes. Regarding the CSLAG proof, the value c_0 is 32 bytes, whereas each pair $(s_i^{(x)}, s_i^{(r)})$ is $(32, 32)$ bytes. Additionally, each output amounts to 32 bytes for the public key plus 32 bytes for the amount's commitment. In total, the StakeNote proofs require 1,064 bytes plus 128 bytes for each output in the ring.

4.2 Output Amount Leakage

Our construction reveals a lower bound T of the used output's stake amount. Specifically, the construction reveals that *one of* the outputs in the anonymity set has *some amount* which is higher than T. Therefore, a relevant question is how close T is to the real stake amount of the correct output, i.e., how much does T reveal regarding the correct output's amount.

Since the VRF output is chosen from a uniform distribution, the value T should also be uniform, albeit with the extra restriction of being smaller than the output's real amount. We experimentally verify this intuition as follows. We ran 10,000 block creation instances, each simulating the leader election process, for various stake amounts (between $5 - 45\%$ of the total stake). In each instance, we used a binary search to find the smallest (180 decimal point precision) threshold, for which the leader eligibility inequality holds. Next, we measured this threshold as a percentage of the output's real amount. These percentages form the distribution of observed values for the threshold T.

To evaluate how close this distribution is to the uniform distribution, we performed a series of tests: (i) χ^2 test: $1{,}052 - 1{,}141$; (ii) $p-$value: $0.00115 - 0.0973$; (iii) total variation: $0.128 - 0.134$; (iv) max deviation: $0.0011 - 0.0013$. Under all tests, the observed distribution of T values is indeed very close to uniform.

5 Practical Enhancements

Although the protocol of Section 3 is sound, it presents two potential drawbacks. First, blocks created with the same output are linkable via the key image. Second, security hinges on half of *all* stake participating honestly, thus not allowing honest regular users to abstain from consensus. In this section we discuss practical enhancements that address these concerns.

5.1 Block Unlinkability

In Ouroboros Praos the stake distribution snapshot is taken before the epoch starts, so only the VRF keys of the snapshot are eligible for leader election. In CryptoNote this is not possible, since outputs are not verifiably spent. A CryptoNote transaction defines a set of "decoy" outputs (the anonymity set) and the user proves that the output o can be spent, while hiding o within the anonymity set, so no party can identify whether o is unspent.

This creates a problem: when using a UTxO o to create a block, how can one prove that o is in the snapshot? In StakeNote (Section 3) we utilize *key images*. However, key images are deterministic and link all blocks created by the same output. To avoid block linkability, either the output's key image should be used once or not used at all in the block's proof.

The first practical option is to minimize the times when a key image is used. Essentially, this is achievable by using each UTxO only once for block production and consuming it immediately after the first use. In that case the UTxO's key image is used exactly twice: once for producing a block and once for spending the output. In practice, a privacy-oriented participant can split their staking amount across many outputs such that, on expectation, each output produces at most one block per two epochs.

An alternative approach is to explicitly specify which outputs are eligible for consensus participation, ensuring that these outputs are identifiably spent. This is achievable by adding an extra bit to every output, which signifies whether it is a regular payment output or a "staking" output. To consume a staking output, the party does not use any decoys, so the exact point when each staking output is spent is publicly identifiable. Additionally, staking outputs cannot be used as decoys for regular payment transactions, while also payment outputs cannot be used as decoys for block creation. Essentially, the system's stake is temporarily split between payment and staking outputs, so each epoch's snapshot consists of unspent staking outputs.

5.2 Dynamic Participation via Estimated Stake

Enabling dynamic participation is more challenging, because it requires changes on the protocol level. Here we sketch a mechanism which allows to infer participating stake by observing the block production rate of every epoch.

The protocol of Figure 1 makes use of the global variable v_{total}, i.e., the total stake in the system. Now, we replace v_{total} with a per-epoch total-stake value $v_{(e)}$, assuming an initial value $v_{(0)}$ for the first two epochs. Each epoch lasts R slots. At the beginning of epoch e, the protocol counts the number of blocks $B_{(e-2)}$ that were added to the canonical chain during epoch $e-2$ and estimates the stake used by the leader election in e as:

$$v_{(e)} = v_{(e-2)} \cdot \frac{B_{(e-2)}}{f \cdot R}$$

where $f \cdot R$ is the expected number of blocks per epoch when both (i) $v_{(e)}$ matches the true participating stake and (ii) all participating stake is honest. Therefore, in the per-output eligibility threshold of Figure 1, v_{total} is replaced with $v_{(e)}$.

The base protocol of Section 3 is secure under the standard Praos assumption that more than half of the total stake is honest. Given the above stake estimation mechanism, this assumption changes such that, in every epoch, the honest parties control more than half of the *participating* stake.

Intuitively, the new variant's security holds as follows. For ease of readability and without loss of generality, we assume that the honest and adversarial stake amounts remain fixed throughout the epochs discussed next.

Assume that the protocol's execution is secure up to epoch e. On e, the inferred participating stake is $v_{(e)}$. Let v_H and $v_{\mathcal{A}}$ denote the honest and adversarial participating stake respectively on epoch e, with $v_H > v_{\mathcal{A}}$ by assumption.

If all participating stake $v_H + v_{\mathcal{A}}$ was honest, then block production would be ideal and the estimated stake for epoch $e+2$ would be: $v_{(e+2)} = v_{(e)} \cdot \frac{B_{(e-2)}}{f \cdot R} = v_{(e)}$.

However, the adversary $\mathcal{A}$ can choose to withhold blocks arbitrarily, in order to artificially change the stake estimation. Remember that each output is elected per slot with probability proportional to its stake divided by $v_{(e)}$ and, due to aggregation independence, the election probability of each party depends only on its stake amount, not on how it is spread across multiple UTxOs. Therefore, the honest parties create on expectation $f \cdot R \cdot \frac{v_H}{v_{(e)}}$ blocks during epoch e, whereas $\mathcal{A}$ creates $f \cdot R \cdot \frac{v_{\mathcal{A}}}{v_{(e)}}$ blocks.

By withholding blocks arbitrarily, $\mathcal{A}$ can bias the estimated stake participation for epoch $e+2$. Specifically, $\mathcal{A}$ can choose for $B_{(e)}$ any value in the range: $[f \cdot R \cdot \frac{v_H}{v_{(e)}}, f \cdot R]$. Consequently, the estimated stake for epoch $e+2$ can take any value in the range: $[v_H, v_{(e)}]$. By assumption it holds that $v_H > v_{\mathcal{A}}$, so the final range of estimated stake for epoch $e+2$ is: $[\frac{v_{(e)}}{2}, v_{(e)}]$.

For any value v in this interval, the per-output threshold rescales honest and adversarial outputs identically. Therefore, the ratio of expected honest to adversarial block production remains $\frac{v_H}{v_{\mathcal{A}}}$. Since the security argument of Ouroboros Praos depends on this ratio rather than on absolute block rates, its security should intuitively carry over, with one caveat: by lowering the threshold for block eligibility, more blocks are created on expectation and, consequently, the probability of forks increases. Therefore, the frequency of slots with multiple honest leaders also increases, hence chain growth is slightly hurt. Nonetheless, the active slots coefficient f can be set small enough such that, even under the artificially lowered stake estimation, the probability of multiple honest leaders per slot (which result in honest forks) does not affect security. A full formal analysis of this mechanism, including the interaction with network delay and chain-quality losses due to honest forks, is left for future work.

References

1 Christian Badertscher, Peter Gazi, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 913–930. ACM, 2018. doi:10.1145/3243734.3243848.

2 Foteini Baldimtsi, Varun Madathil, Alessandra Scafuro, and Linfeng Zhou. Anonymous lottery in the proof-of-stake setting. In *33rd IEEE Computer Security Foundations Symposium, CSF 2020, Boston, MA, USA, June 22-26, 2020*, pages 318–333. IEEE, 2020. doi:10.1109/CSF49147.2020.00030.

3 Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 459–474. IEEE Computer Society, 2014. doi:10.1109/SP.2014.36.

4 Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 315–334. IEEE Computer Society, 2018. doi:10.1109/SP.2018.00020.

5 Sanaz Chamanara, S Arman Ghaffarizadeh, and Kaveh Madani. The environmental footprint of bitcoin mining across the globe: Call for urgent action. *Earth's Future*, 11(10):e2023EF003871, 2023.

6 Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 78–96. Springer, 2006. doi:10.1007/11818175_5.

7 David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer, 1992. doi:10.1007/3-540-48071-4_7.

8 Miranda Christ, Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Deepak Maram, Arnab Roy, and Joy Wang. Sok: Zero-knowledge range proofs. In Rainer Böhme and Lucianna Kiffer, editors, *6th Conference on Advances in Financial Technologies, AFT 2024, September 23-25, 2024, Vienna, Austria*, volume 316 of *LIPIcs*, pages 14:1–14:25. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.AFT.2024.14.

9 Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Annual International Cryptology Conference*, pages 174–187. Springer, 1994. doi:10.1007/3-540-48658-5_19.

10 Ivan Damgård. On σ-protocols. *Lecture Notes, University of Aarhus, Department for Computer Science*, 84, 2002.

11 Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 66–98. Springer, 2018. doi:10.1007/978-3-319-78375-8_3.

12 Bernardo David and Dimitris Karakostas. StakeNote. Software, swhId: swh:1:dir:2ee7704e94b9fa3526b672fabc2b7cc381af6830 (visited on 2026-09-17). URL: https://github.com/commonprefix/stakenote, doi:10.4230/artifacts.27543.

13 Bernardo David and Dimitris Karakostas. StakeNote: A proof-of-stake protocol for CryptoNote payments. Cryptology ePrint Archive, Paper 2026/1205, 2026. URL: https://eprint.iacr.org/2026/1205.

14 Digiconomist. Bitcoin energy consumption index, 2025. URL: https://digiconomist.net/bitcoin-energy-consumption.

15 Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the fiat-shamir transform. In Steven D. Galbraith and Mridul Nandi, editors, *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*, volume 7668 of *Lecture Notes in Computer Science*, pages 60–79. Springer, 2012. doi:10.1007/978-3-642-34931-7_5.

16 Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio Orlandi. Quisquis: A new design for anonymous cryptocurrencies. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 649–678. Springer, 2019. doi:10.1007/978-3-030-34578-5_23.

17 Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986. doi:10.1007/3-540-47721-7_12.

18 Cambridge Centre for Alternative Finance. Cambridge bitcoin electricity consumption index, 2025. URL: https://ccaf.io/cbnsi/cbeci.

19 Eiichiro Fujisaki and Koutarou Suzuki. Traceable ring signature. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography - PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16-20, 2007, Proceedings*, volume 4450 of *Lecture Notes in Computer Science*, pages 181–200. Springer, 2007. doi:10.1007/978-3-540-71677-8_13.

20 Chaya Ganesh, Claudio Orlandi, and Daniel Tschudi. Proof-of-stake protocols for privacy-aware blockchains. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 690–719. Springer, 2019. doi:10.1007/978-3-030-17653-2_23.

21 Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 281–310. Springer, 2015. doi:10.1007/978-3-662-46803-6_10.

22 Emanuele Giunta and Alistair Stewart. Unbiasable verifiable random functions. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part IV*, volume 14654 of *Lecture Notes in Computer Science*, pages 142–167. Springer, 2024. doi:10.1007/978-3-031-58737-5_6.

23 Brandon Goodell, Sarang Noether, and Arthur Blue. Concise linkable ring signatures and forgery against adversarial keys. Cryptology ePrint Archive, Paper 2019/654, 2019. URL: https://eprint.iacr.org/2019/654.

24 Abraham Hinteregger and Bernhard Haslhofer. Short paper: An empirical analysis of monero cross-chain traceability. In Ian Goldberg and Tyler Moore, editors, *Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers*, volume 11598 of *Lecture Notes in Computer Science*, pages 150–157. Springer, 2019. doi:10.1007/978-3-030-32101-7_10.

25 Thomas Kerber, Aggelos Kiayias, Markulf Kohlweiss, and Vassilis Zikas. Ouroboros crypsinous: Privacy-preserving proof-of-stake. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 157–174. IEEE, 2019. doi:10.1109/SP.2019.00063.

26 Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 357–388. Springer, 2017. doi:10.1007/978-3-319-63688-7_12.

27 Markulf Kohlweiss, Varun Madathil, Kartik Nayak, and Alessandra Scafuro. On the anonymity guarantees of anonymous proof-of-stake protocols. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 1818–1833. IEEE, 2021. doi:10.1109/SP40001.2021.00107.

28 Amrit Kumar, Clément Fischer, Shruti Tople, and Prateek Saxena. A traceability analysis of monero's blockchain. In Simon N. Foley, Dieter Gollmann, and Einar Snekkenes, editors, *Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II*, volume 10493 of *Lecture Notes in Computer Science*, pages 153–173. Springer, 2017. doi:10.1007/978-3-319-66399-9_9.

29 David Mazieres. The stellar consensus protocol: A federated model for internet-level consensus. *Stellar Development Foundation*, 32(4):1–45, 2015.

30 Sarah Meiklejohn and Rebekah Mercer. Möbius: Trustless tumbling for transaction privacy. *Proc. Priv. Enhancing Technol.*, 2018(2):105–121, 2018. doi:10.1515/POPETS-2018-0015.

31 Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *40th Annual Symposium on Foundations of Computer Science, FOCS 1999, New York, NY, USA, October 17-18, 1999*, pages 120–130. IEEE Computer Society, 1999. doi:10.1109/SFFCS.1999.814584.

32 Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, and Nicolas Christin. An empirical analysis of traceability in the monero blockchain. *Proc. Priv. Enhancing Technol.*, 2018(3):143–163, 2018. doi:10.1515/POPETS-2018-0025.

33 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

34 Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, 2004. doi:10.1145/972639.972643.

35 Shen Noether, Adam Mackenzie, et al. Ring confidential transactions. *ledger*, 1:1–18, 2016. URL: https://ledgerjournal.org/ojs/index.php/ledger/article/view/34.

36 Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991. doi:10.1007/3-540-46766-1_9.

37 Andrew Poelstra, Adam Back, Mark Friedenbach, Gregory Maxwell, and Pieter Wuille. Confidential assets. In Aviv Zohar, Ittay Eyal, Vanessa Teague, Jeremy Clark, Andrea Bracciali, Federico Pintore, and Massimiliano Sala, editors, *Financial Cryptography and Data Security - FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers*, volume 10958 of *Lecture Notes in Computer Science*, pages 43–63. Springer, 2018. doi:10.1007/978-3-662-58820-8_4.

38 Nicolas Van Saberhagen. Cryptonote v 2.0, 2013.

39 Chenghong Wang, David Pujol, Kartik Nayak, and Ashwin Machanavajjhala. Private proof-of-stake blockchains using differentially-private stake distortion. In Joseph A. Calandrino and Carmela Troncoso, editors, *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pages 1577–1594. USENIX Association, 2023. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/wang-chenghong.

40 Sam M. Werner, Daniel Perez, Lewis Gudgeon, Ariah Klages-Mundt, Dominik Harz, and William J. Knottenbelt. Sok: Decentralized finance (defi), 2022. arXiv:2101.08778.

41 Dimaz Ankaa Wijaya, Joseph K. Liu, Ron Steinfeld, Dongxi Liu, and Jiangshan Yu. On the unforkability of monero. In Steven D. Galbraith, Giovanni Russello, Willy Susilo, Dieter Gollmann, Engin Kirda, and Zhenkai Liang, editors, *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, AsiaCCS 2019, Auckland, New Zealand, July 09-12, 2019*, pages 621–632. ACM, 2019. doi:10.1145/3321705.3329823.

42 Jiangshan Yu, Man Ho Allen Au, and Paulo Jorge Esteves Veríssimo. Re-thinking untraceability in the cryptonote-style blockchain. In *32nd IEEE Computer Security Foundations Symposium, CSF 2019, Hoboken, NJ, USA, June 25-28, 2019*, pages 94–107. IEEE, 2019. doi:10.1109/CSF.2019.00014.

43 Zuoxia Yu, Man Ho Au, Jiangshan Yu, Rupeng Yang, Qiuliang Xu, and Wang Fat Lau. New empirical traceability analysis of cryptonote-style blockchains. In Ian Goldberg and Tyler Moore, editors, *Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers*, volume 11598 of *Lecture Notes in Computer Science*, pages 133–149. Springer, 2019. doi:10.1007/978-3-030-32101-7_9.

Approximation and Hardness Results for the Parallel-Block Construction Problem

Arivarasan Karmegam
IMDEA Networks Institute, Madrid, Spain
Universidad Carlos III de Madrid, Spain

Alexandru Popa
University of Bucharest, Romania

Lucianna Kiffer
IMDEA Networks Institute, Madrid, Spain

Antonio Fernández Anta
IMDEA Software Institute, Madrid, Spain
IMDEA Networks Institute, Madrid, Spain

Abstract

Several high-throughput blockchains, including Solana, Sui, and Aptos, execute transactions in parallel across multiple cores to improve throughput and reduce latency. However, transaction conflicts induced by shared state access fundamentally couple block construction with parallel scheduling. We formalize and study the *Parallel-Block Construction* problem: given transactions with execution times, rewards, and conflict relations, select and schedule a subset of transactions on p parallel cores within a runtime (gas) budget to maximize total reward.

We provide a comprehensive analysis of the complexity and approximation landscape for this problem. When the number of cores p is part of the input, we prove strong inapproximability via a reduction from Maximum Clique: unless NP = ZPP, no polynomial-time algorithm achieves an $n^{1-\epsilon}$-approximation for any $\epsilon > 0$. For constant p and uniform processing times, we give a greedy algorithm achieving a tight $(1 - 1/e)$-approximation ratio. We further establish NP-completeness even for fixed $p = 4$ and unit-length transactions, and show that for $p = 2$ the problem admits an exact polynomial-time algorithm via a reduction to maximum-weight matching (with a cardinality constraint). In contrast, allowing heterogeneous processing times restores hardness: the problem becomes NP-complete for $p = 2$ with processing times in $\{1, 3\}$ (even with unit rewards). For $p = 2$ and processing times in $\{1, 2\}$, we design a polynomial-time $(2/3 - \delta)$-approximation algorithm (for any $\delta > 0$) via a structural decomposition and a reduction to a budgeted matching problem.

We validate our theoretical results with experiments on Ethereum mainnet execution traces. In the homogeneous setting, our $(1 - 1/e)$-approximation algorithm almost always achieves rewards above 99% of the MILP-based optimal baseline, far exceeding the guaranteed factor of $1 - 1/e = 0.63$. In the heterogeneous setting ($p = 2$, processing times in $\{1, 2\}$), the non-overlap variant achieves above 99% of the optimal across all tested configurations, confirming that the theoretical 2/3 bound is a pessimistic worst-case guarantee that does not reflect typical performance on real workloads.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Theory of computation → Problems, reductions and completeness; Theory of computation → Parallel algorithms

Keywords and phrases Blockchain, Parallel Execution, NP-Completeness, Approximation Algorithms

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.9

Funding This work is funded in part by MICIU/AEI /10.13039/501100011033/, ERDF, and the ESF+ under predoctoral training grant PREP2022-000373 and grant DRONAC (PID2022-140560OB-I00) and by MICIU/AEI/10.13039/501100011033 under grant CEX2024-001471-M.

Acknowledgements We would like to thank Lioba Heimbach and the DISCO group at ETH Zurich for providing the transaction traces used in our experiments section.



© Arivarasan Karmegam, Alexandru Popa, Lucianna Kiffer, and Antonio Fernández Anta;
licensed under Creative Commons License CC-BY 4.0



1 Introduction

Block builders run on multi-core CPUs, yet many platforms still execute transactions largely sequentially to preserve deterministic, order-equivalent semantics. This conservative execution model leaves substantial performance on the table: a large fraction of transactions touch disjoint parts of the state and could be processed concurrently, but any pair that overlaps on a state key with at least one write must be serialized. As a result, parallelism is constrained by *conflict structure* rather than only by available hardware. Chains that execute concurrently, such as Solana's Sealevel runtime [19, 22], Block-STM-based systems [9, 1], and object-centric designs such as Sui [5], confront exactly this constraint.

This tension is especially acute during *block construction*. A block builder assembling a block from a mempool is not only deciding how to execute transactions fast but also making an economic choice about which transactions to include under a hard runtime (or gas) budget. When execution is done in parallel, these decisions become tightly coupled: selecting a transaction can reduce feasible parallelism by introducing conflicts, while a more parallel-friendly subset can increase total reward by fitting more transactions into the same budget. A recent systems work [13] has formalized and empirically evaluated this viewpoint via the *Parallel-Block Construction* (PBC) problem, proposing MILP baselines and fast deterministic heuristics and showing that MILPs quickly become intractable while heuristics remain practical at scale.

1.1 Problem Overview

In this paper, we complement the systems perspective of Karmegam et al. [13] with a *complexity and approximation-theoretic* study of the underlying optimization problem. Understanding the formal computational boundaries of this problem is practically important; it tells block builders which regimes admit efficient, optimal, or near-optimal scheduling, and which regimes are provably intractable regardless of the algorithm used.

We assume a mempool in which each transaction τ is characterized by its execution time $\tau.t$, reward $\tau.reward$, and read/write sets $(\tau.R, \tau.W)$, which induce conflicts as defined in Section 2. The PBC problem is defined as follows: given a set *Mpool* of transactions, p identical cores, conflict constraints induced by read/write sets, and a runtime budget B, select a subset of *Mpool* and schedule it non-preemptively so that (i) conflicting transactions never overlap, (ii) the makespan is at most B, and (iii) the total reward is maximized. PBC captures the algorithmic core of parallel-block construction and exposes sharp trade-offs between reward and parallelizability. This coupling of selection and scheduling connects PBC to classical packing and scheduling-with-conflicts problems, and it exposes clique- and matching-like structure in the induced compatibility graph that we exploit in several regimes.

While our model is deliberately simplified, real mempools contain transactions with highly varied execution times, complex smart-contract interactions, and richer conflict structures than we capture here. Our goal is not to model Ethereum in full generality, but to establish the complexity-theoretic foundations of parallel-block construction by analyzing the cleanest non-trivial cases and progressively relaxing assumptions. Even in these simplified regimes, the picture is already rich: sharp phase transitions emerge between tractable and intractable cases as p or the allowed processing-time set changes by the smallest possible amount. Understanding where these boundaries lie is a necessary step toward a complete theory of conflict-aware block construction.

1.2 Related Work

Our work sits at the intersection of (i) deterministic parallel execution in blockchains, and (ii) classical scheduling and packing under conflict constraints. On the systems side, several designs demonstrate that substantial speedups are possible when parallelism is exploited; on the theory side, conflict constraints naturally lead to hard combinatorial optimization problems, motivating a careful complexity and approximation analysis.

Parallel execution in blockchain systems. A large line of systems work aims to exploit multi-core hardware while preserving deterministic semantics. Speculative execution engines such as Block-STM execute transactions optimistically in parallel and resolve conflicts via deterministic validation and re-execution against a fixed transaction order [9]. In contrast, Solana's Sealevel runtime relies on *declared access lists* (read/write accounts) to schedule non-conflicting transactions concurrently, while serializing conflicts [19, 22]. Object-centric designs adopt related ideas by using explicit access metadata to expose parallelism (e.g., at the level of objects) while preserving deterministic outcomes [5]. These works focus primarily on accelerating execution of an already determined block (typically with a fixed order), whereas our focus is on the *block construction* side: selecting transactions from the mempool and scheduling them under a hard runtime budget to maximize reward.

Validator-side optimization for block construction. Closest to our setting is recent validator-side work that formalizes and empirically studies (i) ordered-block scheduling and (ii) parallel-block construction under conflict constraints, proposing MILP formulations and fast heuristics and evaluating them on Ethereum traces [13]. Their results show that exact MILP optimization becomes quickly intractable as instance size, heterogeneity, or core count grows, while lightweight heuristics remain practical and achieve near-optimal solutions on many workloads [13]. In contrast to this empirical/systems emphasis, our contribution is to establish a detailed *complexity- and approximation-theoretic* landscape for the PBC problem, including strong inapproximability for general p, exact solvability in special regimes, and tight approximation guarantees in others.

List scheduling for precedence constraints. Deterministic execution under a fixed total order naturally induces precedence constraints, relating ordered-block execution to precedence-constrained makespan scheduling on identical machines ($P|\text{prec}|C_{\max}$). This classical problem is NP-hard in general [20], and it is a standard motivation for approximation via list scheduling, for which the celebrated $(2 - 1/p)$ performance guarantee is due to Graham [10, 11]. While our paper centers on block *construction* (where selection is also a decision variable), these scheduling foundations help explain why even executing a fixed ordered block optimally is computationally challenging, and why approximation and structural restrictions are essential.

Scheduling with conflict graphs. Our model relates to scheduling under incompatibilities represented by a *conflict graph* $G = (V, E)$. In the classical parallel-machine setting with conflicts (often written $P|G|C_{\max}$), each edge indicates a pair of jobs that cannot be assigned to the same machine, and feasibility is equivalent to m-colorability of G [14]. For unit-time jobs in slotted time, the same conflict-graph abstraction yields a direct correspondence to graph coloring: colors represent time slots, so feasible schedules correspond one-to-one with proper colorings and the chromatic number equals the minimum number of slots [15]. With heterogeneous processing times, this clean equivalence breaks, and one naturally encounters richer coloring generalizations (e.g., multicoloring variants for jobs requiring

Table 1 Summary of complexity and approximation results for the Parallel-Block Construction (PBC) Problem.

Block length	p	Tx Lengths	Result	Reference
One transaction	Arbitrary	Homogeneous	$(n^{1-\epsilon})$-inapproximable	Corollary 8
Arbitrary	Constant	Homogeneous	$(1-1/e)$-approximation	Theorem 9
Constant	Constant	Homogeneous	Exact polynomial time	Remark 11
Arbitrary	4	Homogeneous	NP-complete	Theorem 14
Arbitrary	2	Homogeneous	Exact polynomial time	Theorem 15
Arbitrary	2	$\{1,3\}$	NP-complete	Theorem 17
Arbitrary	2	$\{1,2\}$	$\left(\frac{2}{3}-\delta\right)$-approximation	Theorem 19

multiple slots) [15]. In blockchain execution, this perspective has been used to contrast *block-order-preserving* schedulers with *coloring-based* orderings derived from (approximate) colorings of the transaction conflict graph; the latter can offer substantial latency improvements depending on the conflict ratio and workload [17].

Packing/knapsack under conflicts. On the selection side, PBC generalizes several packing problems. Multiple knapsack (MKP) is a canonical model for selecting profitable items subject to multiple capacity constraints and admits a PTAS in important regimes [6]. Adding conflicts between items yields the knapsack problem with conflict graphs, which is strongly NP-hard in general and has been studied via exact methods and pseudo-polynomial/FPTAS results on restricted graph classes [16]. Our setting further couples such conflict-aware selection with parallel-machine scheduling under a makespan budget, which goes beyond classical knapsack variants and helps explain the need for new complexity classifications and approximation techniques.

Clique hardness and matching-based optimization. Our hardness results leverage the well-known inapproximability of Maximum Clique established by Håstad, under the standard assumption NP $\neq$ ZPP [12]. On the algorithmic side, our polynomial-time solvability results for restricted regimes exploit matching structure; maximum-weight matching is solvable in polynomial time via Edmonds' algorithm, and parametric/budgeted variants are a classical tool for enforcing cardinality-type constraints [7, 18]. These techniques become relevant in our $p = 2$ regime, where schedules decompose into pairings and singletons, enabling exact optimization.

1.3 Our Contributions

We provide a comprehensive complexity and approximation landscape for the Parallel-Block Construction problem.

Homogeneous Transactions (Uniform Processing Times)

Strong Inapproximability for Arbitrary p. When the number of cores p is part of the input, we prove that the problem is extremely hard to approximate. Via a reduction from Maximum Clique, we show that, unless NP = ZPP, no polynomial-time algorithm achieves an $n^{1-\varepsilon}$-approximation for any $\varepsilon > 0$.

$(1-1/e)$-Approximation for Constant p. When p is constant, and all transactions have identical processing time, we design a greedy round-based algorithm that achieves a tight $(1-1/e)$-approximation ratio. The algorithm repeatedly selects a maximum-reward clique of size at most p in the compatibility graph.

NP-Completeness for $p = 4$. Even when $p = 4$ and all transactions have unit processing time, we prove that the decision version of the PBC problem is NP-complete via a reduction from 3D-Matching. Thus, fixing the number of cores does not eliminate computational hardness.

Exact Polynomial-Time Algorithm for $p = 2$. For $p = 2$ and uniform processing times, we show that the problem admits an exact polynomial-time solution. The key idea is a reduction to maximum-weight matching with a cardinality constraint, solvable using parametric matching techniques.

Heterogeneous Transactions

Hardness with Heterogeneous Processing Times. Allowing heterogeneous processing times immediately restores hardness: for $p = 2$ and processing times in $\{1, 3\}$ (with unit rewards), the problem becomes NP-complete.

$(2/3 - \delta)$-Approximation for $p = 2$, Lengths $\{1, 2\}$. For $p = 2$ and processing times in $\{1, 2\}$, we design a polynomial-time $(2/3 - \delta)$-approximation algorithm for any $\delta > 0$. The algorithm reduces a structured non-overlap variant to a budgeted maximum-profit matching problem and uses a structural decomposition of optimal schedules to bound the loss.

Table 1 summarizes the complexity results described. Taken together, our results reveal sharp phase transitions: small changes in the number of cores or in the transaction lengths dramatically alter computational complexity.

1.4 Organization of the Paper

Section 2 introduces notation and formal definitions. Section 3 describes the proofs and approximations for cases which handle only homogeneous transactions, i.e., transactions with the same execution time. Section 3.1 proves strong inapproximability when p is part of the input. Section 3.2 presents the $(1 - 1/e)$-approximation algorithm for constant p. Section 3.3 establishes NP-completeness for $p = 4$. Section 3.4 gives the exact polynomial-time algorithm for $p = 2$ in the uniform case. Section 4 describes the proofs and approximations for heterogeneous transactions. Section 4.1 proves NP-completeness for $p = 2$ with lengths in $\{1, 3\}$. Section 4.2 presents the $(2/3 - \delta)$-approximation algorithm for $p = 2$ with lengths in $\{1, 2\}$. In Section 5, Subsection 5.1 discusses implementation aspects and practical considerations, and Subsection 5.2 discusses the results. Section 6 concludes the paper.

2 Preliminaries

Transactions. We assume that a block builder has an execution environment that can process up to p transactions in parallel, where p is the number of cores of the block builder. Each transaction τ has a *Priority Fee*, denoted $\tau.tip$. Computation is priced in units of *gas*, an abstract measure of computational effort. When executed, τ consumes a certain amount of gas $\tau.gasU$, and yields a reward, $\tau.reward = \tau.gasU \cdot \tau.tip$, which goes to the block builder. The $\tau.tip$ could be 0, so $\tau.reward \geq 0$. We also assume that the time to execute a transaction τ is proportional to its gas consumption. Let us denote the execution time of a transaction τ as $\tau.t$. Transactions are non-preemptive. We denote by $\tau.R$ and $\tau.W$ the subset of state keys a transaction τ reads and writes, respectively (Here "keys" refers to accounts or contracts by address).

Conflicts. We say that two transactions τ_i and τ_j conflict if and only if they access a common state key and at least one of them writes to it. Formally, $\text{conflict}(\tau_i, \tau_j) \equiv \big((\tau_i.W \cap \tau_j.W) \cup (\tau_i.W \cap \tau_j.R) \cup (\tau_i.R \cap \tau_j.W) \neq \varnothing\big)$.

Schedules and the PBC problem. Then, we can define:

▶ **Definition 1** (Schedule). *A schedule S of a set of transactions T on p cores assigns to each transaction in T a core and a starting time to run. Given a schedule S of a set of transactions T on p cores, its* makespan*, denoted $S.mspan$, is the time to complete the execution (without preemption) of all transactions in T as defined by S.*

We now formalize the problem we study.

▶ **Problem 2** (Parallel-Block Construction (PBC), optimization version).
Input: *a mempool $Mpool = \{\tau_1, \ldots, \tau_n\}$, where each transaction τ_i has an execution time $\tau_i.t \in \mathbb{Z} > 0$, a reward $\tau_i.reward \in \mathbb{Q} \geq 0$ and read/write sets $(\tau_i.R, \tau_i.W)$ over a universe of state keys; a number of cores $p \in \mathbb{Z} > 0$; and a runtime limit $B \in \mathbb{Z} > 0$. All numbers are encoded in binary.*
Output: *a subset $T \subseteq Mpool$ and a schedule S of T on p cores (Definition 1) such that (i) no two transactions assigned to the same core overlap in time, (ii) no two conflicting transactions overlap in time, and (iii) $S.mspan \leq B$.*
Objective: *maximize $T.reward = \sum_{\tau \in T} \tau.reward$.*

▶ **Problem 3** (PBC-D, decision version).
Input: *an instance of Problem 2 together with a target reward $K \in \mathbb{Q} \geq 0$.*
Question: *does there exist a pair (T, S) feasible for Problem 2 with $T.reward \geq K$?*

3 Homogeneous Transactions

3.1 Hardness of Approximation for Arbitrary p

In this section, we show that, unless $NP = ZPP$, Problem 2 (PBC) is hard to approximate to a factor of $n^{1-\epsilon}, \forall \epsilon > 0$. The assumption $NP \neq ZPP$ is standard in hardness-of-approximation results. Recall that ZPP is the class of problems solvable by zero-error randomized algorithms with expected polynomial running time [2]. Hence, $NP = ZPP$ would imply that every NP-complete problem admits a zero-error randomized algorithm with expected polynomial running time, which is widely believed to be unlikely.

The proof is done via an approximation-preserving reduction from the maximum clique problem that we define next.

▶ **Definition 4** (Clique). *Let $G = (V, E)$ be an undirected graph. A subset $C \subseteq V$ is a* clique *if every pair of distinct vertices in C is adjacent in G, i.e., $\forall\, u, v \in C,\ u \neq v \ \Rightarrow\ (u, v) \in E$. (Equivalently, the subgraph of G induced by C is complete.)*

▶ **Problem 5** (Maximum Clique Problem). *Let $G = (V, E)$ be an undirected graph. Given G, finding a clique $C \subseteq V$ of maximum cardinality, i.e., $C \in \arg\max\{\, |C'| : C' \subseteq V,\ C' \text{ is a clique in } G \,\}$, is called the* Maximum Clique *problem.*

▶ **Theorem 6** (Håstad [12]). *For any $\epsilon > 0$, unless $NP = ZPP$, there is no polynomial-time algorithm that approximates* Max-Clique *within a factor of $n^{1-\epsilon}$.*

We show next the reduction from Maximum Clique to Problem 2.

Algorithm 1 Greedy $(1 - 1/e)$-approximation for Problem 2 (constant p, homogeneous transactions).

1. **Input:** Set of transactions T, runtime budget B, compatibility graph $G = (T, E)$.
2. **Constant:** Number of cores p.
3. Set $R \leftarrow \lfloor B/\tau.t \rfloor$.
4. Set $T_0 \leftarrow T$.
5. Initialize the schedule $S \leftarrow \emptyset$.
6. For each round $i = 1, 2, \ldots, R$ do:
   a. In the induced subgraph $G[T_{i-1}]$, choose a clique $C_i \in \arg\max\{C.reward : (C \text{ is a clique in } G[T_{i-1}]) \wedge (|C| \leq p)\}$.
   b. If $C_i = \emptyset$, then **break** the loop.
   c. Schedule all transactions in C_i in parallel, in round i, i.e., add (i, τ) to S for every $\tau \in C_i$.
   d. Update the remaining transactions set: $T_i \leftarrow T_{i-1} \setminus C_i$.
7. **Output:** schedule S.

▶ **Theorem 7.** *If there exists a polynomial time c-approximation algorithm for Problem 2, then the Maximum Clique problem admits a polynomial time c-approximation algorithm.*

Proof. Let $G = (V, E)$ be an instance of Maximum Clique, where $V = \{v_1, \ldots, v_n\}$. We construct an instance $\mathcal{I}(G)$ of Problem 2 as follows.

Let $T = \{\tau_1, \ldots, \tau_n\}$ be the set of transactions corresponding to V. For each vertex $v_i \in V$, create a corresponding transaction τ_i with reward $\tau_i.reward = 1$ and processing time $\tau_i.t = 1$. For every non-edge, i.e., $(v_i, v_j) \notin E$, introduce a conflict between τ_i and τ_j as follows. Define a different item c_{ij} for each pair of tasks τ_i and τ_j, and include c_{ij} in both $\tau_i.W$ and $\tau_j.W$. We emphasize that c_{ij} is unique to the pair (i, j) and is not included in the read/write sets of any other transaction. Thus, two transactions are non-conflicting if and only if their corresponding vertices are adjacent in G. We set the number of cores to $p = n$ and the runtime limit to $B = 1$, so that any feasible schedule must execute all the transactions in a single round.

We now prove that for any integer k, G contains a clique of size k *if and only if* the constructed instance admits a feasible solution of total reward k.

($\Rightarrow$) If direction. Let $C \subseteq V$ be a clique of size k in G. By definition of a clique, for any distinct $v_i, v_j \in C$ we have $(v_i, v_j) \in E$, hence τ_i and τ_j do not conflict. Therefore, all transactions $\{\tau_i : v_i \in C\}$ can be scheduled in parallel in a single round, yielding a feasible schedule with a reward k.

($\Leftarrow$) Only-if direction. Let $T' \subseteq T$ be any feasible solution of reward k for Problem 2. Since every transaction has reward 1, $|T'| = k$. We define the corresponding vertex set $C = \{v_i : \tau_i \in T'\}$. Feasibility implies that no two transactions in T' conflict, and by construction, this holds if and only if for all distinct $\tau_i, \tau_j \in T'$ we have $(v_i, v_j) \in E$. Since every pair of distinct vertices in C is adjacent in G, the set C is a clique of size k in G. ◀

Given the result in Theorem 6, we have the following corollary.

▶ **Corollary 8.** *For any $\epsilon > 0$, unless $NP = ZPP$, there is no polynomial-time algorithm that approximates Problem 2 within a factor of $n^{1-\epsilon}$.*

3.2 $(1 - 1/e)$-Approximation for Constant p

We show in Section 3.1 that Problem 2 is hard to approximate if $p = n$, so p is part of the input. In this section, we show a polynomial time approximation algorithm that achieves a factor of $(1 - 1/e)$, if p is a constant and all the transactions are homogeneous, i.e., they take the same amount of time $\tau.t$.

We divide time into rounds of $\tau.t$ time each. The block builder can execute at most p transactions in parallel per round. Let $R = \lfloor B/\tau.t \rfloor$ be the maximum number of rounds possible in the block, where B is the runtime limit of the block. Let T denote the set of all transactions in the mempool, and we define the *compatibility graph* $G = (T, E)$ as follows. The vertex set is the set of transactions T, and an edge $\{\tau, \tau'\} \in E$ indicates that τ and τ' do not conflict and therefore can be executed in parallel. For a subset $T' \subseteq T$, we write $T'.reward = \sum_{\tau \in T'} \tau.reward$.

The algorithm follows a greedy strategy, has R rounds, and at each round it selects a maximum-reward clique C_i of size at most p in the current compatibility graph and schedules all transactions in that clique in parallel. (For convenience, we use C_i to denote the clique and its vertices. Its meaning is clear from the context.)

▶ **Theorem 9.** *Algorithm 1 is a polynomial time* $(1 - 1/e)$*-approximation and runs in time* $O(n^p R)$.

Proof. We first bound the running time and then prove the approximation ratio.

Running time. For each round $i \in \{1, \ldots, R\}$, recall that Algorithm 1 maintains the remaining transaction set $T_{i-1} \subseteq T$ and chooses a clique $C_i \subseteq T_{i-1}$ by maximizing the total reward over all cliques $C \subseteq T_{i-1}$ of size at most p in the compatibility graph $G[T_{i-1}]$. Since p is a constant, the number of candidate subsets is $\sum_{k=0}^{p} \binom{|T_{i-1}|}{k} = O(n^p)$, where $n = |T|$. For each candidate C, we can check in constant time (depending only on p) whether C is a clique in G and compute $C.reward$. Thus, one iteration of this loop takes $O(n^p)$ time. As there are at most R rounds, the total running time is $O(n^p R)$.

Approximation ratio. Let OPT denote the maximum total reward of any feasible schedule that uses at most R rounds. Let $(S_1, \ldots, S_R)$ be the schedule produced by Algorithm 1. Recall that C_i is the maximum-reward clique of size at most p chosen by Algorithm 1 in round i, so $S_i = C_i$. For each $i \in \{0, \ldots, R\}$, let $A_i = \bigcup_{j=1}^{i} S_j$ be the set of transactions scheduled by the algorithm in the first i rounds, with the convention $A_0 = \emptyset$. We also write $A_i.reward = \sum_{\tau \in A_i} \tau.reward$. Consider a fixed optimal schedule with rounds $O_1, \ldots, O_R$, where each O_ℓ is a clique of size at most p in the compatibility graph and $\sum_{\ell=1}^{R} O_\ell.reward = OPT$.

We first prove a key lemma that bounds the additional reward obtained by each greedy round of Algorithm 1 in terms of the remaining optimal reward.

▶ **Lemma 10.** *For every* $i = 1, \ldots, R$, $A_i.reward - A_{i-1}.reward \geq \frac{OPT - A_{i-1}.reward}{R}$.

Proof. Since rewards are nonnegative and $T_{i-1} = T \setminus A_{i-1}$, the reward of the transactions of the optimal schedule that remain available at round i is at least the optimal reward minus the reward already collected by the greedy schedule.

Formally, let $O = \bigcup_{\ell=1}^{R} O_\ell$ be the set of transactions scheduled by the fixed optimal solution. Then $O.reward = \sum_{\ell=1}^{R} O_\ell.reward = OPT$, and

$$\begin{aligned}(O \cap T_{i-1}).reward &= (O \setminus A_{i-1}).reward = O.reward - O \cap A_{i-1}.reward \\ &\geq O.reward - A_{i-1}.reward = OPT - A_{i-1}.reward.\end{aligned}$$

Moreover, $\bigcup_{\ell=1}^{R} (O_\ell \cap T_{i-1}) = O \cap T_{i-1}$, so

$$\Big(\bigcup_{\ell=1}^{R} (O_\ell \cap T_{i-1})\Big).reward \geq OPT - A_{i-1}.reward.$$

For each $\ell \in \{1, \ldots, R\}$ let $O'_\ell = O_\ell \cap T_{i-1}$. Then,

$$\sum_{\ell=1}^{R} O'_\ell.reward \;\geq\; OPT - A_{i-1}.reward.$$

Hence there exists an index $\ell^* \in \{1, \ldots, R\}$ such that

$$O'_{\ell^*}.reward \;\geq\; \frac{OPT - A_{i-1}.reward}{R}.$$

By construction, $O'_{\ell^*} \subseteq T_{i-1}$ and O'_{ℓ^*} is a clique of size at most p in G, so O'_{ℓ^*} is a feasible choice in round i. Since Algorithm 1 selects a clique C_i maximizing $C.reward$ among all cliques $C \subseteq T_{i-1}$ with $|C| \leq p$, we have

$$C_i.reward \;\geq\; O'_{\ell^*}.reward \;\geq\; \frac{OPT - A_{i-1}.reward}{R}.$$

Moreover, by construction $C_i \subseteq T_{i-1} = T \setminus A_{i-1}$, so C_i is disjoint from A_{i-1} and therefore $A_i.reward - A_{i-1}.reward = C_i.reward$. Combining the two inequalities yields the claim. ◀

We now unroll the inequality of Lemma 10. Define $\Delta_i = OPT - A_i.reward$ for $i = 0, \ldots, R$. Lemma 10 is equivalent to $A_i.reward - A_{i-1}.reward \;\geq\; \frac{\Delta_{i-1}}{R}$, which we can rewrite as

$$\Delta_i = OPT - A_i.reward \leq OPT - \left(A_{i-1}.reward + \frac{\Delta_{i-1}}{R}\right) = \Delta_{i-1}\left(1 - \frac{1}{R}\right)$$

Thus, by induction on i,

$$\Delta_i \;\leq\; \left(1 - \frac{1}{R}\right)^i OPT \qquad \text{for all } i = 0, \ldots, R.$$

In particular, for $i = R$ we obtain

$$OPT - A_R.reward = \Delta_R \;\leq\; \left(1 - \frac{1}{R}\right)^R OPT,$$

and therefore

$$A_R.reward \;\geq\; \left(1 - \left(1 - \frac{1}{R}\right)^R\right) OPT.$$

Using the standard inequality $(1 - 1/R)^R \leq 1/e$, we finally obtain $A_R.reward \;\geq (1 - 1/e)\, OPT$. Hence, Algorithm 1 achieves an approximation ratio of $(1 - 1/e)$. Combining the running-time analysis with the approximation guarantee completes the proof. ◀

▶ Remark 11. When both p and $R = \lfloor B/\tau.t \rfloor$ are constants, Problem 2 is solvable in $O(n^{pR} \cdot Rp^2)$ time by exhaustive enumeration over all ordered sequences of R disjoint cliques of size at most p. There are $O(n^p)$ candidate cliques per round, disjointness across rounds reduces the effective pool by at most pR transactions, and the feasibility of each candidate clique can be checked in $O(p^2)$ time. The NP-hardness results in this work both require R to grow with the input and do not apply when R is fixed.

3.3 NP-Completeness for $p = 4$

In this section, we prove that Problem 3 (PBC-D) is NP-complete even when the number of cores of the block builder is fixed at $p = 4$, and all transactions are homogeneous (i.e., all have identical execution times). It is immediate that PBC-D is in NP: a certificate is a schedule S, and we can in polynomial time check that the rounds in S are conflict-free, compute its total reward and its makespan, and compare them to K and B, respectively. We prove our result using a polynomial reduction from the 3D-Matching problem, which we define below.

▶ **Definition 12** (3D-Matching). *Let V_1, V_2, V_3 be three pairwise disjoint sets of equal size n, and let $E \subseteq V_1 \times V_2 \times V_3$ be a set of hyperedges, where each hyperedge $e = (v_1, v_2, v_3)$ contains exactly one vertex from each of the three sets V_1, V_2, V_3. Given an integer k, the* 3D-Matching *problem asks whether there exists a subset $E' \subseteq E$ such that $|E'| = k$ and such that every two distinct hyperedges in E' are vertex-disjoint: $\forall e, e' \in E', e \neq e' \implies e \cap e' = \emptyset$.*

It is well known that 3D-matching is NP-complete [8].

▶ **Theorem 13.** *For $p = 4$ and homogeneous transactions, PBC-D is NP-hard.*

Proof. Let $V = V_1 \cup V_2 \cup V_3$ be the vertex set of a 3D-matching instance, where V_1, V_2, V_3 are pairwise disjoint and $E \subseteq V_1 \times V_2 \times V_3$ is the set of hyperedges and an integer k. We construct an instance of PBC-D as follows.

- For each vertex $v \in V$ we construct a vertex-transaction τ_v with $\tau_v.t = 1$ and $\tau_v.reward = 1$.
- For each hyperedge $e = (v_1, v_2, v_3) \in E$ we construct a hyperedge-transaction τ_e with $\tau_e.t = 1$ and $\tau_e.reward = 1$.

We now define conflicts between these transactions. Let us define the sets $T_r = \{\tau_v : v \in V_r\}$, for $r \in \{1, 2, 3\}$. Then, we make all vertex-transactions in set T_r pairwise conflicting, by defining items c_r, and for each $\tau \in T_r$ including c_r in $\tau.W$. Thus, in any round, at most one transaction from each set V_r, $r \in \{1, 2, 3\}$, can be scheduled.

All hyperedge-transactions $\{\tau_e : e \in E\}$ are pairwise conflicting. Moreover, for every hyperedge $e = (v_1, v_2, v_3) \in E$, we make τ_e conflict with every vertex-transaction except its three endpoint vertex-transactions $\tau_{v_1}, \tau_{v_2}, \tau_{v_3}$ as follows. For each vertex $u \in V \setminus \{v_1, v_2, v_3\}$, introduce a fresh state key $c_{e,u}$ and include $c_{e,u}$ in both $\tau_e.W$ and $\tau_u.W$. Hence, τ_e conflicts with τ_u for every $u \notin e$, while this construction does not introduce additional conflicts among vertex-transactions in different partitions and in any feasible round, τ_e can be scheduled together only with τ_{v_1}, τ_{v_2} and τ_{v_3}.

We set the number of cores to $p = 4$ and the gas budget to $B = k$. Since all transactions have unit execution time and there is no preemption, the makespan equals the number of rounds used by the schedule. We have to show that the hypergraph has k independent hyperedges *if and only if* there exists a solution to Problem 3 on the corresponding instance of reward precisely $4k$.

($\Rightarrow$) If direction. Suppose the 3D-matching instance has a matching $E' \subseteq E$ of size k. Let S denote the schedule we construct below. For each hyperedge $e = (v_1, v_2, v_3) \in E'$, assign a distinct round $i_e \in \{1, \ldots, k\}$ and schedule the four transactions $\{\tau_{v_1}, \tau_{v_2}, \tau_{v_3}, \tau_e\}$ in parallel during round i_e. This schedule is feasible because:

- Within each round i_e, the three vertex-transactions $\tau_{v_1}, \tau_{v_2}, \tau_{v_3}$ belong to different partitions V_1, V_2, V_3 respectively, so they do not conflict with each other.

Algorithm 2 Exact algorithm for $p = 2$ for the homogeneous case of Problem 2.

1. Construct the following weighted graph $G' = (V', E')$:
   - For each transaction $\tau_i \in Mpool$, create a corresponding vertex $\tau_i \in V'$.
   - If two transactions τ_i and τ_j *do not conflict*, add an edge (τ_i, τ_j) in G' with weight $\tau_i.reward + \tau_j.reward$.
   - For each transaction τ_i, add a dummy vertex τ_i' in G' and an edge (τ_i, τ_i') with weight $\tau_i.reward$. (This represents scheduling τ_i alone in a round.)
   - Add $2R$ additional dummy vertices $d_1, \ldots, d_{2R} \in V'$ and, for each $r = 1, \ldots, R$, add an edge $(d_{2r-1}, d_{2r}) \in E'$ of weight 0. (This represents an unused/idle round.)
2. Compute a maximum-weight matching in G' consisting of exactly R edges.
3. Convert each matching edge into a round:
   - $(\tau_i, \tau_j) \rightarrow$ schedule τ_i and τ_j in parallel,
   - $(\tau_i, \tau_i') \rightarrow$ schedule τ_i alone,
   - $(d_{2r-1}, d_{2r}) \rightarrow$ idle round (execute nothing).

   Finally, remove all idle rounds.

- By construction, τ_e conflicts only with transactions other than $\tau_{v_1}, \tau_{v_2}, \tau_{v_3}$, so it can be scheduled together with exactly these three transactions.
- Since E' is a matching, the vertex sets $\{v_1, v_2, v_3\}$ for distinct $e \in E'$ are disjoint. Thus, no vertex-transaction τ_v appears in more than one round.

The schedule uses k rounds of length 1 (so $S.mspan = k = B$), uses 4 transactions per round on $p = 4$ cores, and has total reward $4k$.

(⇐) Only-if direction. Suppose there exists a feasible schedule S for Problem 3 with total reward exactly $4k$ and makespan $S.mspan \leq B = k$. Since each transaction has a reward 1 and there are exactly $4k$ transaction-slots available ($p \cdot B = 4k$), the schedule must use exactly 4 distinct transactions on all 4 cores in every one of the k rounds. To reach exactly 4 transactions per round, each round i must contain exactly:

- One vertex-transaction from each V_r: $\tau_{v_1^i} \in \{\tau_{v_1} : v_1 \in V_1\}$, $\tau_{v_2^i} \in \{\tau_{v_2} : v_2 \in V_2\}$, $\tau_{v_3^i} \in \{\tau_{v_3} : v_3 \in V_3\}$.
- One hyperedge-transaction τ_e and by construction, τ_e is compatible with $\{\tau_{v_1^i}, \tau_{v_2^i}, \tau_{v_3^i}\}$ if and only if $e = (v_1^i, v_2^i, v_3^i) \in E$.

Thus each round i corresponds to a hyperedge $e_i = (v_1^i, v_2^i, v_3^i) \in E$. Let $E' = \{e_1, \ldots, e_k\} \subseteq E$. Since all $4k$ transactions across all rounds are distinct (no transaction is reused), no vertex-transaction τ_v appears in more than one round, so the vertex sets $\{v_1^i, v_2^i, v_3^i\}$ for $i = 1, \ldots, k$ are pairwise disjoint. Therefore, E' is a matching of size k.

Polynomial size and time. The reduction can be constructed in polynomial time. It creates $|V|$ vertex-transactions and $|E|$ hyperedge-transactions. The conflict encoding uses a constant number of global keys (c_1, c_2, c_3, c_h), and for each hyperedge $e \in E$ it introduces at most $|V| - 3$ fresh keys $c_{e,u}$ (one for each non-endpoint vertex u), for a total of $O(|E| \cdot |V|)$ keys. Thus, the size of the constructed PBC instance and the time needed to build it are polynomial in the size of the 3D-Matching instance. ◀

▶ **Theorem 14.** *For $p = 4$ and homogeneous transactions, PBC-D is NP-complete.*

Proof. The problem is in NP, as argued at the beginning of this section and for $p = 4$ and homogeneous transactions, it is NP-hard by Theorem 13 via reduction from 3D-Matching. ◀

3.4 Exact Polynomial Time Algorithm for $p = 2$

In this section, we present an exact polynomial time algorithm for Problem 2 for $p = 2$, and all transactions are homogeneous, i.e., they have identical execution time $\tau.t$. In this setting, the runtime budget B bounds the number of rounds by $\lfloor B/\tau.t \rfloor$, and in each round we can schedule at most two transactions in parallel. Since every non-idle round contains at least one transaction and no transaction is scheduled twice, at most n rounds of any schedule are non-idle, so we may work with $R = \min\{\lfloor \frac{B}{\tau.t} \rfloor, n\}$ rounds without excluding any optimal schedule. The cap is what makes the construction polynomial: B is encoded in binary, so $\lfloor B/\tau.t \rfloor$ alone may be exponential in the input size, whereas $R \leq n$.

The key idea is to reduce this special case to a maximum-weight matching problem on a carefully constructed graph $G' = (V', E')$. Algorithm 2 shows the details. We assume rewards $\tau.reward$ are integers. Since the runtime budget B only imposes an upper bound on the makespan, an optimal solution may use fewer than R rounds, leaving part of the budget unused. We enforce a matching of exactly R edges so that each edge corresponds to one of the R available rounds; unused rounds are represented by 0-weight idle-round edges. These are proof artifacts and are not executed in the final schedule. Discarding idle rounds yields a feasible schedule for the original instance with the same total reward. We now show that Algorithm 2 is correct and runs in polynomial time.

▶ **Theorem 15.** *For $p = 2$ and homogeneous transactions, Algorithm 2 computes an optimal solution to Problem 2 in polynomial time.*

Proof. Step 1 runs in $O(n^2)$ time, since $R \leq n$ and the construction adds $O(n)$ dummy vertices and edges.

Implementing Step 2 in polynomial time. Let m denote the maximum cardinality of a matching in G'. By construction, $m \geq R$ since the R disjoint edges (d_{2r-1}, d_{2r}) form a matching of size R.

We compute a maximum-weight matching of *exactly* R edges in G' in polynomial time as follows. For an integer parameter λ, define modified edge weights $w_\lambda(e) = w(e) - \lambda$. Let M_λ denote a maximum-weight matching in G' under weights w_λ. White [21] shows that for every $k \leq m$ there exists a value λ such that *some* maximum-weight matching under w_λ has cardinality exactly k, and gives a polynomial-time algorithm (making only polynomially many calls to a maximum-weight matching routine) to find such a matching. Therefore, we can compute in polynomial time a maximum-weight matching of cardinality exactly R. Each subroutine call to compute a maximum-weight matching can be implemented in polynomial time using Edmonds' weighted matching algorithm (see also standard textbook treatments [18]).

Correctness: schedule ↔ matching. *(⇒)* Given any feasible schedule of reward W, at most n of its rounds are non-idle; discarding idle rounds and padding with $R - t$ idle rounds, where $t \leq R$ is the number of non-idle rounds, yields exactly R rounds. Construct a set of R edges in G' by:

- if a round schedules τ_i and τ_j together, include edge (τ_i, τ_j);
- if a round schedules only τ_i, include edge (τ_i, τ_i');
- if a round is idle, include one distinct idle-round edge (d_{2r-1}, d_{2r}).

Since no transaction is scheduled more than once and all idle-round edges are disjoint, these edges form a matching of size R. Its total weight is exactly W.

(⇐) Given a matching M of size R and weight W, construct an R-round schedule by interpreting edges as follows:

- each edge (τ_i, τ_j) corresponds to scheduling two non-conflicting transactions τ_i and τ_j in parallel;
- each edge (τ_i, τ_i') corresponds to scheduling τ_i alone;
- each edge (d_{2r-1}, d_{2r}) corresponds to an idle round.

Since M is a matching, no vertex τ_i appears in more than one edge, hence no transaction is scheduled more than once. Removing idle rounds preserves feasibility and does not change the total reward, so the resulting schedule is feasible and has reward W.

Therefore, Algorithm 2 computes an optimal schedule in polynomial time. ◀

4 Heterogeneous Transactions

4.1 NP-Completeness for $p = 2$, $\tau.t \in \{1, 3\}$

In this section, we show that Problem 3 (PBC-D) is NP-complete even when restricted to $p = 2$, unit rewards ($\tau.reward = 1\ \forall\tau$), and execution times $\tau.t \in \{1, 3\}$. The proof uses a reduction from 3D-Matching, similar to that of Section 3.3, but adapted to exploit the different execution times.

▶ **Theorem 16.** *PBC-D is NP-hard for $p = 2$, unit rewards, and execution time $\tau.t \in \{1, 3\}$.*

Proof. We reduce from 3D-Matching. Let (V_1, V_2, V_3, E) be a 3D-Matching instance with $|V_1| = |V_2| = |V_3| = n$ and $|E| = m$, where $V = V_1 \cup V_2 \cup V_3$. We construct an instance $\mathcal{I}$ of PBC-D as follows.

1. **Transactions.** Create $3n + m$ transactions, each with reward 1.
   - For each vertex $v \in V$, create a *vertex-transaction* τ_v with $\tau_v.t = 1$.
   - For each hyperedge $e = (v_1, v_2, v_3) \in E$, create a *hyperedge-transaction* τ_e with $\tau_e.t = 3$.
2. **Conflicts.** Define three conflict types:
   - All vertex-transactions $\{\tau_v : v \in V\}$ are *pairwise conflicting.*
   - All hyperedge-transactions $\{\tau_e : e \in E\}$ are *pairwise conflicting.*
   - For each hyperedge $e = (v_1, v_2, v_3) \in E$, τ_e conflicts with *every transaction except* $\tau_{v_1}, \tau_{v_2}, \tau_{v_3}$.
3. **Parameters.** Set $p = 2$ cores, runtime budget $B = 3k$, and $K = 4k$.

We prove that the 3D-Matching instance has a matching of size k if and only if $\mathcal{I}$ admits a feasible schedule of reward at least $4k$.

(⇒) If direction. Suppose $E' \subseteq E$ is a matching of size k. For each $e = (v_1, v_2, v_3) \in E'$, construct a round $i_e \in \{1, \ldots, k\}$ scheduling:

- Core 1: $\tau_{v_1}, \tau_{v_2}, \tau_{v_3}$ in *series* (total length $1 + 1 + 1 = 3$),
- Core 2: τ_e (length 3).

This schedule uses k rounds of length 3, so $S.mspan = 3k \leq B$. It is feasible because:

- The three vertex-transactions on Core 1 are executed in series; hence, although mutually conflicting, they do not overlap in time and therefore do not violate feasibility.
- By construction, τ_e is compatible exactly with $\tau_{v_1}, \tau_{v_2}, \tau_{v_3}$.
- E' is a matching, so vertex sets $\{v_1, v_2, v_3\}$ for each $e \in E'$ are pairwise disjoint; no transaction repeats.

The schedule uses $4k$ transactions, yielding a reward $4k$.

(⇐) Only-if direction. Suppose $\mathcal{I}$ admits a feasible schedule S with $S.mspan \leq 3k$ and $S.reward \geq 4k$.

By conflict rules, no round can schedule:

- Two vertex-transactions in parallel (since all vertex-transactions are pairwise conflicting).
- Two hyperedge-transactions in parallel (since all hyperedge-transactions are pairwise conflicting).

We first upper bound the total number of transactions that can be executed within a makespan $3k$. Since vertex-transactions are pairwise conflicting, at any time at most one vertex transaction can be executed across both cores; hence, within time $3k$, the schedule can execute at most $3k$ vertex-transactions in total. Moreover, since hyperedge-transactions are pairwise conflicting and each has length 3, within a makespan $3k$, the schedule can execute at most k hyperedge-transactions. Therefore, within a makespan $3k$ any feasible schedule can execute at most $3k + k = 4k$ transactions.

Since $S.reward \geq 4k$ and all rewards are 1, S must execute exactly $4k$ transactions; hence it executes exactly $3k$ vertex-transactions and exactly k hyperedge-transactions, and moreover $S.mspan = 3k$.

Now, we define a *round* to be the time interval during which one hyperedge-transaction is executed. Since hyperedge-transactions have length 3 and are pairwise conflicting, these k hyperedge-transactions define k disjoint rounds of length 3 that cover the entire time horizon $[0, 3k)$. Since the total processing time of the k hyperedge-transactions is $3k$, there can be no idle time between them (otherwise they would not all fit by time 3k). In each round, because at most one vertex can run at any time and we must execute $3k$ vertices overall, exactly three vertex-transactions are executed on the other core (in series) during that same round.

Let τ_{e_i} be the hyperedge-transaction executed in round i, where $e_i = (v_1^i, v_2^i, v_3^i) \in E$. By construction, τ_{e_i} conflicts with every transaction except $\tau_{v_1^i}, \tau_{v_2^i}, \tau_{v_3^i}$, so the three vertex-transactions executed during round i must be exactly $\tau_{v_1^i}, \tau_{v_2^i}, \tau_{v_3^i}$.

Let $E' = \{e_1, \ldots, e_k\} \subseteq E$. Since S executes $4k$ distinct transactions, no vertex-transaction is repeated, so the triples $\{v_1^i, v_2^i, v_3^i\}$ are pairwise disjoint. Hence E' is a matching of size k. ◀

▶ **Theorem 17.** *For $p = 2$, unit rewards, and execution times in $\{1, 3\}$, PBC-D is NP-complete.*

Proof. The problem is in NP: a certificate is a schedule S, and in polynomial time we can verify feasibility (conflict-freeness), compute $S.mspan$ and $S.reward$, and compare them to B and K. NP-hardness follows from Theorem 16. Hence, the decision problem is NP-complete. ◀

4.2 $(\frac{2}{3} - \delta)$-Approximation for $p = 2$, $\tau.t \in \{1, 2\}$

In this section, we consider the restricted setting of Problem 2 with $p = 2$, arbitrary rewards, and processing times $\tau.t \in \{1, 2\}$. Let B denote the runtime budget. We present a deterministic polynomial-time approximation algorithm that, for any $\delta > 0$, returns a feasible schedule with total reward at least $\left(\frac{2}{3} - \delta\right) OPT$. The algorithm is as follows.

1. We define a simplified *non-overlap* variant of Problem 2.
2. We compute a $(1 - \epsilon)$-approximation of its optimal solution OPT_{non} in polynomial time (for fixed $\epsilon > 0$).

We prove that the optimum reward of this variant satisfies $OPT_{\text{non}} \geq \frac{2}{3} OPT$ in this restricted setting. Hence, combining the two steps yields an approximation factor of $(1-\epsilon) \cdot \frac{2}{3}$. Equivalently, for any $\delta > 0$, by choosing $\epsilon \leq \frac{3}{2}\delta$ we obtain a $\left(\frac{2}{3} - \delta\right)$-approximation.

4.2.1 The non-overlap variant

The *non-overlap variant* has the same input as Problem 2, but restricts the schedule to consist of independent rounds, where each round executes either: (a) one transaction of length 1 or 2, or (b) two *non-conflicting* transactions in parallel, and the length of such a parallel round is the maximum of the two transaction lengths. In particular, the non-overlap variant does *not* allow scheduling one length-2 transaction in parallel with two length-1 transactions.

Reduction to a budgeted matching. We reduce the non-overlap variant to a maximum-profit matching problem under a time budget. Construct an undirected graph $H = (V_H, E_H)$ as follows.

- For each transaction τ_i, add two vertices $v_i, v_i' \in V_H$ and add the edge $(v_i, v_i') \in E_H$. To each of these edges, assign a profit $c(v_i, v_i') = \tau_i.reward$ and a cost $w(v_i, v_i') = \tau_i.t$.
- For every pair of non-conflicting transactions τ_i, τ_j, add an edge $(v_i, v_j) \in E_H$. To each of these edges, assign a profit $c(v_i, v_j) = \tau_i.reward + \tau_j.reward$ and a cost $w(v_i, v_j) = \max\{\tau_i.t, \tau_j.t\}$

The non-overlap variant is then equivalent to selecting a matching $M \subseteq E_H$ maximizing $\sum_{e\in M} c(e)$ subject to the time budget $\sum_{e\in M} w(e) \le B$, and then interpreting each matching edge $e \in M$ as one round of length $w(e)$, which yields a feasible non-overlap schedule with the same total reward.

Let OPT_{non} denote the optimal reward for the non-overlap variant.

4.2.2 Computing the Non-Overlap Solution

The above is a *budgeted maximum-profit matching* problem (matching with a knapsack-style budget constraint), which is NP-hard in general. We therefore do not attempt to solve it exactly in polynomial time.

Instead, we use a PTAS for budgeted matching (e.g., the deterministic PTAS of Berger et al. [4]): for any fixed accuracy parameter $\epsilon > 0$, we can compute in polynomial time a matching M_ϵ satisfying $\sum_{e\in M_\epsilon} w(e) \le B$ and $\sum_{e\in M_\epsilon} c(e) \ \ge \ (1-\epsilon)OPT_{\text{non}}$.

4.2.3 $\left(\frac{2}{3} - \delta\right)$-Approximation: Bounding the Loss

Let OPT be the optimal schedule reward for Problem 2 restricted to $p = 2$ and $\tau.t \in \{1, 2\}$. We show $OPT_{\text{non}} \ge \frac{2}{3} OPT$. Together with Step 1, this implies

$$\sum_{e\in M_\epsilon} c(e) \ \ge \ (1-\epsilon)OPT_{\text{non}} \ \ge \ (1-\epsilon)\cdot\frac{2}{3}OPT.$$

Hence, by choosing $\epsilon \le \frac{3}{2}\delta$, we obtain a $\left(\frac{2}{3} - \delta\right)$-approximation.

▶ **Lemma 18.** *There exists an optimal schedule for Problem 2 (in this restricted setting) such that one of the two cores is* gap-free*, i.e., it executes its scheduled transactions back-to-back starting at time* 0*, with no idle interval that is followed by a later transaction on the same core.*[1]

[1] Idle time at the end of the horizon, after that core's last transaction, is permitted and is not a gap; in particular, when the budget cannot be exactly filled, e.g. an odd budget with only length-2 transactions, the unused tail does not violate gap-freeness.

Proof. Assume for contradiction that every optimal schedule has idle time on *both* cores. Fix an optimal schedule, and let i be the earliest time at which core 1 is idle *and some transaction is scheduled on core 1 at a later time* (i.e., i is an interior gap, not trailing idle). If no such i exists, core 1 is already gap-free and we are done.

If core 2 is also idle at time i, then we can shift every transaction that starts at or after time i on both cores to the left by the same amount, until at least one transaction hits time i. This preserves feasibility and makespan while eliminating the idle time at i on core 1, contradicting the choice of the schedule.

Hence, core 2 is not idle at time i. If a transaction starts on core 2 exactly at time i, then swap the assignments of the two cores for all transactions scheduled at times $\geq i$. This preserves feasibility and makespan and removes the idle time of core 1 at time i.

Otherwise, since $\tau.t \in \{1, 2\}$, core 2 must be executing a length-2 transaction during the interval $[i-1, i]$ (i.e., a transaction that starts at time $i-1$ and ends at time $i+1$). In this case, swap the assignments of the two cores for all transactions scheduled at times $\leq i$. Again, feasibility and makespan are preserved, and core 1 is not idle at time i, a contradiction.

Therefore, there exists an optimal schedule in which at least one core has no idle time. ◀

▶ **Theorem 19.** *The algorithm presented above is a* $\left(\frac{2}{3} - \delta\right)$*-approximation for Problem 2 in the case where* $p = 2$ *and every transaction has length* 1 *or* 2*.*

Proof. Let OPT be the maximum reward achievable in Problem 2 restricted to $p = 2$ and $\tau.t \in \{1, 2\}$, and let OPT_{non} denote the optimal reward of the non-overlap variant from Section 4.2.1. Since Step 1 computes M_ϵ with $\sum_{e \in M_\epsilon} c(e) \geq (1 - \epsilon) OPT_{\text{non}}$, it suffices to show $OPT_{\text{non}} \geq \frac{2}{3} OPT$.

Fix an optimal feasible schedule $S^\star$ of total reward OPT and makespan at most B. By Lemma 18, we may assume without loss of generality that core 1 is gap-free in $S^\star$. All processing times are integral.

Cut points and coarse segments. Call an integer time $t \in \{0, 1, \ldots, B\}$ a *cut point* if, at time t, both cores are at a boundary, i.e., for each core, either the core is idle at time t or a transaction starts/ends at t. Let $0 = t_0 < t_1 < \cdots < t_q = B$ be the cut points, and let $I_k = [t_{k-1}, t_k]$ be the resulting *coarse segments*. Write $S^\star|_{I_k}$ for the restriction of $S^\star$ to segment I_k.

We will construct, independently for each I_k, a schedule $\widehat{S}|_{I_k}$ feasible for the non-overlap variant on I_k such that $\widehat{S}|_{I_k}.reward \geq \frac{2}{3} S^\star|_{I_k}.reward$. Concatenating the segment-wise schedules yields a non-overlap schedule $\widehat{S}$ of makespan at most B and reward at least $\frac{2}{3} OPT$, proving the theorem.

Distinguished gap and local move. Fix a coarse segment $I = [a, b]$. If there exists any idle unit on either core within I, define the *distinguished gap* to be the leftmost idle unit interval $[t, t+1] \subseteq I$ among all idle units in I (breaking ties arbitrarily but deterministically). If I is *tightly packed* (both cores are busy throughout I), we will delete one transaction in I with the minimum reward to create an idle interval and then define the distinguished gap as above.

Our only schedule modification is the following *unit alignment move*: whenever a length-2 transaction τ is adjacent to the distinguished gap on its core (i.e., the gap is exactly one unit immediately before or after τ), we shift τ by one time unit into the distinguished gap (left or right), choosing a feasible direction that makes τ become fully aligned with a non-conflicting length-2 transaction on the other core with which τ was half-overlapping. This move preserves feasibility and does not create new conflicts, and it preserves total reward.

Segment types. We distinguish three cases for the structure of $S^\star|_I$.

Case 1: $S^\star|_I$ is already non-overlap-standard. Then set $\widehat{S}|_I := S^\star|_I$, and $\widehat{S}|_I.reward = S^\star|_I.reward$.

Case 2: I is a 2-vs-$(1+1)$ segment. That is, on one core there is a length-2 transaction τ occupying all of $I = [a, a+2]$, and on the other core there are two consecutive length-1 transactions occupying $[a, a+1]$ and $[a+1, a+2]$. This pattern is feasible in the original problem but forbidden by the non-overlap variant. Define $\widehat{S}|_I$ by deleting the minimum-reward transaction among these three, and keeping the remaining two transactions scheduled as in $S^\star|_I$. Then $\widehat{S}|_I$ is feasible for the non-overlap variant on I, and $\widehat{S}|_I.reward \geq \frac{2}{3} S^\star|_I.reward$.

Case 3: I is neither Case 1 nor Case 2. If I is not tightly packed, the distinguished gap already exists. If I is tightly packed, delete one transaction of minimum reward among all transactions scheduled in I by $S^\star|_I$, thereby creating at least one idle unit; then define the distinguished gap to be the leftmost idle unit in I.

Starting from this distinguished gap, repeatedly apply the unit alignment move to the transaction adjacent to the distinguished gap until no such move is possible. Each move shifts a length-2 transaction by one unit into idle time and thus preserves feasibility and reward. Moreover, each move strictly decreases the number of misaligned non-conflicting length-2 half-overlaps within I (i.e., non-conflicting pairs of length-2 transactions overlapping by exactly one unit), since the moved transaction becomes fully aligned with its designated non-conflicting partner. As this number is finite, the process terminates.

At termination, there are no remaining misaligned non-conflicting length-2 half-overlaps in I. Observe that with processing times in $\{1, 2\}$, any local obstruction to being a concatenation of non-overlap-standard rounds is necessarily either (i) a misaligned non-conflicting 2–2 half-overlap, or (ii) a 2-vs-$(1+1)$ pattern over an interval of length 2.

Since at termination (i) does not occur and Case 2 has already been handled exclusively, it follows that the resulting schedule on I is a concatenation of non-overlap-standard rounds. Let $\widehat{S}|_I$ denote the resulting schedule; it is feasible for the non-overlap variant on I. In Case 3, if I is not tightly packed, then no deletion is performed and therefore $\widehat{S}|_I.reward = S^\star|_I.reward \geq \frac{2}{3} S^\star|_I.reward$.

Assume now that I is tightly packed and that we delete one transaction $\tau_{\min}$ of minimum reward among the $n_I := |T(I)|$ transactions scheduled in I (where $T(I)$ is the set of transactions scheduled by $S^\star$ in I). All subsequent unit alignment moves are reward-preserving, hence

$$\begin{aligned}\widehat{S}|_I.reward &= S^\star|_I.reward - \tau_{\min}.reward \geq S^\star|_I.reward - \frac{1}{n_I} S^\star|_I.reward \\ &= \frac{n_I - 1}{n_I} S^\star|_I.reward.\end{aligned}$$

Moreover, in this case we have $n_I \geq 3$ (otherwise I would consist of two perfectly aligned length-2 transactions and would already be non-overlap-standard, contradicting that we perform a deletion in Case 3). Therefore $\frac{n_I - 1}{n_I} \geq \frac{2}{3}$, and thus $\widehat{S}|_I.reward \geq \frac{2}{3} S^\star|_I.reward$.

Combining segments. Apply the above transformation independently to each coarse segment I_k and concatenate the resulting schedules $\widehat{S}|_{I_k}$. Since the I_k are consecutive and share cut points at their endpoints, the concatenation is a feasible schedule $\widehat{S}$ for the non-overlap variant of makespan at most B. Moreover,

$$\widehat{S}.reward = \sum_{k=1}^{q} \widehat{S}|_{I_k}.reward \geq \frac{2}{3} \sum_{k=1}^{q} S^\star|_{I_k}.reward = \frac{2}{3} S^\star.reward = \frac{2}{3} OPT.$$

Therefore $OPT_{\text{non}} \geq \widehat{S}.reward \geq \frac{2}{3}OPT$, and combining this with step 1 yields

$$\sum_{e \in M_\epsilon} c(e) \;\geq\; (1-\epsilon)OPT_{\text{non}} \;\geq\; (1-\epsilon)\cdot\frac{2}{3}OPT,$$

and choosing $\epsilon \leq \frac{3}{2}\delta$ gives a $(\frac{2}{3}-\delta)$-approximation. ◀

5 Experiments and Results

5.1 Experimental Setup

We empirically evaluate Algorithm 1 in the homogeneous (round-based) setting with constant p (see Section 3.2), and the approximation algorithm of Section 4.2 on heterogeneous instances with $p = 2$ and processing times in $\{1, 2\}$. Since the PTAS of Berger et al. [4] used in the algorithm of Section 4.2 is not straightforward to implement directly, we instead solve the non-overlap variant exactly via an MILP formulation for the experiments in this section.

Hardware and Software. All experiments were run on a Dell PowerEdge R7615 equipped with a single AMD EPYC 9654P (96 cores/ 192 threads) and ≈1.5TiB RAM, running Ubuntu 24.04.3 LTS. MATLAB R2025b is used to implement Algorithm 1 and invoke HiGHS 1.7.1 through MATLAB's `intlinprog` for MILPs. A maximum running time of $MaxTime := 6000$ s. was set for each experiment.

Workloads. We use Ethereum mainnet execution traces from blocks 21,631,019–21,635,079 (15–16 January 2025), from which we extract per-transaction gas used and read/write sets, and derive conflicts from overlapping read/write sets (see Section 2). To obtain homogeneous processing times to evaluate Algorithm 1, we restrict to pure ETH transfers (intrinsic gas 21,000), which do not invoke smart contracts. Across the dataset, the number of conflicts per transaction ranges from 0 to $11,369$, with a mean of $1,363$ and a median of 75, reflecting a heavily skewed distribution in which a small number of transactions generate the majority of conflicts. For the experiments of the algorithm in Section 4.2, we additionally generate heterogeneous instances by bundling consecutive transactions in a 1:2 ratio. Hence, we have single transactions (of length 1), and double transactions (with length 2) obtained by merging pairs of two consecutive transactions, inheriting the union of their access lists and the sum of their rewards. Length-1 and length-2 transactions each comprise 50% of the pool, distributed uniformly at random. For this heterogeneous dataset, the number of conflicts per transaction ranges from 0 to $18,895$, with a mean of 2159 and a median of 361.

Our workloads instantiate the specific theoretical regimes of this paper rather than the full diversity of Ethereum execution. The homogeneous workload matches the uniform-processing-time assumption; the heterogeneous $\{1, 2\}$ workload is a controlled synthetic construction realizing Section 4.2.

Experiment parameters. For the homogeneous transactions, experiment duration is discretized into equal-length rounds, so progress is naturally parameterized by the number of rounds R, with per-round capacity p. For a configuration (R, p, X), we emulate a mempool by selecting a sliding-window pool of $R \cdot p \cdot X$ consecutive transactions, where X is the pool factor (so the pool contains roughly X times the transactions that fit in R rounds). We evaluate each configuration with rounds $R \in \{10, 30, 100\}$, cores $p \in \{2, 4, 8\}$, and pool factor $X \in \{2, 4, 8\}$. For the heterogeneous experiments of Section 4.2, which concern the

Table 2 Random baseline, Algorithm 1, and the heuristics of [13]: block reward $\sum_\tau \tau.reward$ as a percentage of the MILP optimum from [13]. †Average of only instances that completed; the remainder timed out (See Section 5.2).

	X = 2								
	Random			**Algorithm 1**			**Heuristics from [13]**		
	p = 2	**p = 4**	**p = 8**	**p = 2**	**p = 4**	**p = 8**	**p = 2**	**p = 4**	**p = 8**
R = 10	73.07	69.72	61.02	99.66	99.08	99.92	99.66	98.46	99.84
R = 30	58.96	49.15	65.73	99.72	99.63	99.91	99.67	99.47	99.86
R=100	60.86	60.85	58.55	99.99	99.94	99.99†	99.99	99.94	99.95
	X = 4								
	Random			**Algorithm 1**			**Heuristics from [13]**		
	p = 2	**p = 4**	**p = 8**	**p = 2**	**p = 4**	**p = 8**	**p = 2**	**p = 4**	**p = 8**
R = 10	44.95	40.66	40.07	100	99.98	99.54	99.92	99.90	99.49
R = 30	40.14	46.68	42.34	99.56	99.44	99.86	99.47	99.44	99.85
R=100	41.3	48.8	39.73	100	99.99	99.89†	100	99.98	99.88
	X = 8								
	Random			**Algorithm 1**			**Heuristics from [13]**		
	p = 2	**p = 4**	**p = 8**	**p = 2**	**p = 4**	**p = 8**	**p = 2**	**p = 4**	**p = 8**
R = 10	30.81	25.27	28.33	99.99	99.99	98.76	99.98	99.99	98.75
R = 30	30.05	23.41	21.34	100	99.54	99.7	100	99.5	99.7
R=100	22.9	28.46	15.99	99.78	99.98	100†	99.78	99.97	100

$p = 2$ setting by definition, we parameterize by runtime budget $B \in \{210\text{K}, 630\text{K}, 2100\text{K}\}$ gas units (corresponding to $R \in \{10, 30, 100\}$ rounds of pure ETH transfers) and pool factor $X \in \{2, 4, 8\}$, where X controls the size of the mempool: a pool factor of X means the mempool contains roughly X times the number of transactions that fit within the budget B. For both experiment types, we run five non-overlapping workloads and report the mean.

Random baseline. This baseline applies to the homogeneous experiments of Algorithm 1 only. To test whether near-optimality is due to the algorithmic scheduling decisions (rather than the workload being inherently easy), we compare Algorithm 1 not only against the MILP-based optimum [13], but also against a simple *random feasible* baseline. In each round, we generate a random permutation of the remaining transactions and scan it from left to right. Whenever we encounter a feasible transaction (i.e., non-conflicting with the transactions already selected for the current round), we add it to the round and mark all transactions conflicting with it as *ineligible for this round*. We continue scanning the same permutation until selecting p transactions or exhausting the list, then proceed to the next round using the remaining unscheduled transactions.

5.2 Results

We evaluate the greedy $(1-1/e)$-approximation algorithm (Algorithm 1) on the homogeneous (round-based) PBC setting and compare its achieved block reward against the MILP-based optimal baseline from prior work on parallel block construction [13]. We compare against two further baselines: a random feasible schedule, which tests whether near-optimality stems from the scheduling decisions or from the workload being inherently easy; and the fast deterministic heuristics of [13], which represent the kind of lightweight strategy a production builder would realistically run.

Table 2 reports normalized reward as a percentage of the MILP optimum. Across all tested configurations (varying R, p, and the pool factor X), Algorithm 1 consistently produces blocks whose total reward $\sum_\tau \tau.reward$ is very close to the MILP optimum. In practice, our

algorithm almost always achieves rewards above 99% of the optimal, which is much better than the guaranteed factor of $1 - 1/e = 0.63$. In contrast, the random baseline performs substantially worse, and the gap widens as the pool factor X increases: for example, at $X = 8$, random achieves only about 16–31% of optimal across our R, p settings. This indicates that the near-optimal behavior of Algorithm 1 is driven by its conflict-aware selection, not by the workload being trivially easy. The deterministic heuristics of [13] also reach above 99% of the optimum, matching Algorithm 1 in reward in some configurations, performing a bit worse in many cases but never better. The distinction between the two lies not in reward but in guarantees and cost, which we examine next.

We set the same per-instance timeout as prior work (6,000 seconds) [13]. The heuristics run in well under a second across every configuration (at most 0.9 s, at R=100, p=8, X=8). In contrast, each round of Algorithm 1 solves an exact maximum-reward clique of size at most p, which costs $O(n^p)$ in the worst case. For $p = 2$ and $p = 4$ it likewise remains sub-second, but at $p = 8$ the runtime grows sharply: at R=100, p=8 it reaches 557 s at X=4 and 2499 s at X=8, and a small number of the largest instances (1, 1, and 4 instances out of the 5 runs, for X=2, 4, 8 respectively, all at R=100, p=8) exceed the time cap. This is the empirical face of the $O(n^p)$ per-round cost: exact selection becomes expensive exactly where the heuristic remains cheap. That the exact search nonetheless terminates for most $p = 8$ instances reflects the structure of Ethereum conflict graphs, which are dense enough that each transaction is compatible with relatively few others, keeping the effective clique search far below its worst case.

Table 3 Non-overlap reward as a percentage of optimal solution, for $p = 2$ and transaction gas amounts 21K and 42K.

	X=2	**X=4**	**X=8**
B=210K	99.65	99.78	99.69
B=630K	99.94	100	99.95
B=2100K	99.988	99.99	99.98

Taken together, these results show that on realistic homogeneous workloads a cheap heuristic is enough to recover almost all the reward, and that the value of Algorithm 1 is not empirical dominance but its provable $(1 - 1/e)$ guarantee: unlike the heuristics, it retains a principled performance floor on adversarial instances, where reward-ordered strategies can be forced to a $1/p$ fraction of the optimum.

Table 3 reports, for $p = 2$ and the heterogeneous workload of Section 4.2, the reward achieved by the non-overlap MILP as a percentage of the optimal MILP solution, averaged over five non-overlapping workloads per configuration. Please note that the theoretical algorithm of Section 4.2 computes a $(1 - \varepsilon)$-approximation of OPT_{non} via a PTAS for budgeted matching. In our experiments, we instead solve the non-overlap variant exactly by formulating it directly as an MILP. The reported ratio OPT_{non}/OPT therefore measures the true gap between the non-overlap optimum and the overall optimum, without any approximation error from the PTAS. Across all tested values of B and pool factor X, the non-overlap variant consistently achieves above 99% of the optimal reward, well above the theoretical guarantee of $2/3 - \delta$. The result aligns with the homogeneous setting: the theoretical worst-case bound of 2/3 is far from tight for Ethereum workloads, and the non-overlap restriction incurs negligible reward loss in practice.

6 Conclusions and Future Work

We studied the *Parallel-Block Construction* problem that arises when a block builder jointly decides which mempool transactions to include and how to schedule them on p identical cores under a hard runtime budget, subject to conflict constraints induced by shared state access. Our results provide a detailed complexity and approximation map that explains when algorithmic optimization is plausible and when it is provably out of reach. Our experiments on both homogeneous and heterogeneous Ethereum workloads confirm that the theoretical worst-case bounds are far from tight in practice, with both algorithms consistently achieving above 99% of optimal. Together, these results reveal sharp phase transitions: small changes in p or in the allowed processing-time set can drastically alter what can be computed efficiently.

Future work. Several directions remain open. First, on the homogeneous side, our results place the tractability boundary between $p = 2$ (exactly solvable, Theorem 15) and $p = 4$ (NP-complete, Theorem 14), leaving $p = 3$ unresolved in the present paper. We have preliminary results indicating that the problem is in fact NP-complete for every fixed $p \geq 3$, via a reduction from Partition into Triangles for $p = 3$ [8], lifted to larger p by a padding argument, which would sharpen the boundary to $p \leq 2$ tractable and $p \geq 3$ hard [3]. We leave the full development to future work.

Second, our complexity map for heterogeneous transactions leaves several natural regions open. We resolve only the length sets $\{1, 2\}$ (approximable) and $\{1, 3\}$ (NP-complete), and even these are not fully settled: the approximability of $\{1, 3\}$ remains open, as does the exact complexity of $\{1, 2\}$ and whether its 2/3 bound is tight. The behaviour of arbitrary length sets and of unbounded processing times is likewise open. Closing these cases, together with a better understanding of the parameterized complexity of PBC with respect to p, the number of rounds, or structural properties of the conflict graph, would complete the landscape.

Third, extending to richer workloads, both in conflict structure (e.g., DeFi and ERC-20 transactions) and in model assumptions, would clarify how quickly performance degrades as instances move away from the regimes where we have strong guarantees. Our model assumes transaction execution times and read/write sets are known at scheduling time; while systems such as Solana's Sealevel enforce this via declared access lists [19], Ethereum requires execution or simulation to recover these quantities, and a growing body of work on gas estimation and access-list prediction provides a starting point for such extensions.

Finally, bridging the gap between theory and practice suggests designing provably good but more scalable primitives for key subroutines (notably the per-round clique selection used in the $(1 - 1/e)$-approximation algorithm), as well as studying online/streaming variants that more closely reflect mempool dynamics during real-time block construction.

References

1 Aptos Dev Docs. Block-stm execution engine in aptos, 2026. URL: `https://aptos.dev/network/blockchain/execution`.

2 Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, USA, 1st edition, 2009.

3 Brenda S Baker and Edward G Coffman. Mutual exclusion scheduling. *Theoretical Computer Science*, 162(2):225–243, 1996. `doi:10.1016/0304-3975(96)00031-X`.

4 André Berger, Vincenzo Bonifaci, Fabrizio Grandoni, and Guido Schäfer. Budgeted matching and budgeted matroid intersection via the gasoline puzzle. *Mathematical Programming*, 128(1):355–372, June 2011. `doi:10.1007/s10107-009-0307-4`.

5 Sam Blackshear, Andrey Chursin, George Danezis, Anastasios Kichidis, Lefteris Kokoris-Kogias, Xun Li, Mark Logan, Ashok Menon, Todd Nowacki, Alberto Sonnino, Brandon Williams, and Lu Zhang. Sui lutris: A blockchain combining broadcast and consensus. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, pages 2606–2620, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3658644.3670286.

6 Chandra Chekuri and Sanjeev Khanna. A polynomial time approximation scheme for the multiple knapsack problem. *SIAM Journal on Computing*, 35(3):713–728, 2005. doi:10.1137/S0097539700382820.

7 Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965. doi:10.4153/CJM-1965-045-4.

8 Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1990.

9 Rati Gelashvili, Alexander Spiegelman, Zhuolun Xiang, George Danezis, Zekun Li, Dahlia Malkhi, Yu Xia, and Runtian Zhou. Block-stm: Scaling blockchain execution by turning ordering curse to a performance blessing. In *Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, PPoPP '23, pages 232–244, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3572848.3577524.

10 Ronald L Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(9):1563–1581, 1966.

11 Ronald L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17(2):416–429, 1969.

12 Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182:105–142, 1999.

13 Arivarasan Karmegam, Lucianna Kiffer, and Antonio Fernández Anta. Exploiting Multi-Core Parallelism in Blockchain Validation and Construction. In Martin Aumüller and Irene Finocchi, editors, *24th International Symposium on Experimental Algorithms (SEA 2026)*, volume 371 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:21, Dagstuhl, Germany, 2026. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SEA.2026.23.

14 Daniel Kowalczyk and Roel Leus. An exact algorithm for parallel machine scheduling with conflicts. *Journal of Scheduling*, 20(4):355–372, 2017. doi:10.1007/s10951-016-0482-0.

15 Dániel Marx. Graph colouring problems and their applications in scheduling. *Periodica Polytechnica Electrical Engineering*, 48(1-2):11–16, 2004. URL: https://pp5.omikk.bme.hu/ee/article/view/926.

16 Ulrich Pferschy and Joachim Schauer. The knapsack problem with conflict graphs. *Journal of Graph Algorithms and Applications*, 13(2):233–249, 2009. doi:10.7155/JGAA.00186.

17 Ankit Ravish, Yaron Hay, Piduguralla Manaswini, Rishabh Jain, Roy Friedman, and Sathya Peri. Efficient scheduling of smart contract transactions via conflict graph coloring. In *2025 IEEE 30th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 91–101, 2025. doi:10.1109/PRDC67299.2025.00019.

18 Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Algorithms and Combinatorics. Springer, 2003.

19 Solana. Sealevel — parallel processing thousands of smart contracts. https://solana.com/news/sealevel---parallel-processing-thousands-of-smart-contracts, 2026. Accessed: 2026-01-19.

20 J. D. Ullman. Polynomial complete scheduling problems. In *Proceedings of the Fourth ACM Symposium on Operating System Principles*, SOSP '73, pages 96–101, New York, NY, USA, 1973. Association for Computing Machinery. doi:10.1145/800009.808055.

21 Lee James White. *A parametric study of matchings and coverings in weighted graphs*. PhD thesis, University of Michigan, 1967.

22 Anatoly Yakovenko. Solana: A new architecture for a high performance blockchain. White paper, 2017. Version 0.8.13. URL: https://solana.com/solana-whitepaper.pdf.

Not All Reads Are Conflicts: A Write-Only Analysis of the Sui Blockchain

Haygen Tsoi
University College London (UCL), UK

Alberto Sonnino
Mysten Labs, Palo Alto, CA, USA
University College London (UCL), UK

Philipp Jovanovic
Mysten Labs, Palo Alto, CA, USA
University College London (UCL), UK

Abstract

Sui's object-centric data model enables parallel transaction execution, but realised performance is fundamentally bounded by workload contention. Prior empirical analyses of Sui have relied on "read+write" (R+W) conflict graphs inherited from account-based blockchains. Because Sui's engine serialises only on mutable shared access, R+W graphs contain spurious edges, bounding contention from above. In this paper, we adopt a complementary "write-set-only" (W-only) model in which every edge represents a real write-serialisation event, providing a lower bound on contention. Together, the two models bracket Sui's true execution-dependency structure. Applying the W-only analysis to Sui mainnet data through 2025 yields three primary findings. First, removing read-only dependencies (notably the system clock) causes previously reported "hub-and-spoke" structures to collapse. The remaining contention topology is highly assortative and clique-dominated, with the W-only bound shaving roughly 30–40% off the R+W estimate of Sui's optimal-parallelism headroom. Second, via union-find object grouping, we isolate DeepBook (Sui's native central limit order book). While it dominates contention by volume, its underlying logic does not impose disproportionate sequential bottlenecks. Finally, we quantify the economic cost of contention, showing that 10–50% of the network's USD-denominated value flows through sequentially constrained execution paths, exposing it to potential ordering effects.

2012 ACM Subject Classification Security and privacy → Distributed systems security

Keywords and phrases Sui, blockchain, parallel execution, conflict graphs, transaction analysis

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.10

Related Version *Full Version*: `https://arxiv.org/abs/2607.26691`

Supplementary Material
Software (Source Code): `https://github.com/krdecade-0/sui_analysis_code`
archived at `swh:1:dir:64aa158cc8a9f248d510070371c59fff336295f0`

Funding This work is partially funded by Mysten Labs.

Acknowledgements We would like to thank Mysten Labs for their generous support and funding of this research.

1 Introduction

Since the launch of Bitcoin in 2008 [23], blockchain research has focused on scaling transaction throughput to compete with traditional payment processors like VISA. While early smart-contract platforms like Ethereum [9] introduced expressive programmability, they have struggled with the "ordering curse": a sequential execution model where transactions must be interleaved through a single global state. Sui was introduced as a high-performance alternative



© Haygen Tsoi, Alberto Sonnino, and Philipp Jovanovic;
licensed under Creative Commons License CC-BY 4.0

designed specifically for parallel execution. By adopting an object-centric data model, Sui enables parallel execution for transactions that touch disjoint sets of state. However, as with any parallel system, the actual performance realised in production is limited by the structure of the workload and the frequency of state contention [6].

In this paper, we evaluate the residual parallelism available in Sui by analysing mainnet transaction data through the end of 2025. Prior empirical work [6] adopts a "read+write" (R+W) conflict model inherited from account-based blockchains, where a graph-based approach is used to measure contention. An edge is drawn between two transactions (nodes) if they touch the same object and at least one modifies it. Thus, more edges in the graph mean more contention, which is equivalent to more serialisation and less parallelism. We instead adopt a different "write-only" (W-only) model, where an edge is drawn only if both transactions attempt to modify the same object. Because Sui's execution engine serialises only on mutable shared accesses (Section 2.3), a R+W graph contains edges the engine never has to honour (e.g., every user transaction's read of the system clock is counted as a conflict with the system writer), so it bounds contention from *above*. By contrast, every edge in the W-only graph corresponds to a real write-serialisation event, so W-only bounds contention from *below* (and parallelism from above). The two models bracket Sui's true execution-dependency structure.

Our investigation is guided by four core research questions:

1. **RQ1:** Under a write-only conflict model, what structural topology does Sui's conflict graph exhibit, and how does it differ from the topology reported under R+W? (Section 4)
2. **RQ2:** How does the W-only estimate of achievable parallelism compare quantitatively to the R+W estimate at matched load regimes? (Section 4)
3. **RQ3:** Is write contention concentrated in a small number of smart-contract applications (e.g. DeepBook), or is it widely spread? (Section 5)
4. **RQ4:** What fraction of Sui's economic throughput (value transferred) flows through sequential-burden transactions vs. parallelisable ones? (Section 6)

To address these questions, we apply the W-only model to Sui mainnet transaction data through the end of 2025. The W-only graph reveals a highly assortative, clique-dominated topology, fundamentally different from the disassortative, star-like structures observed under R+W. The shift is driven by the collapse of read-side dependencies on the system clock: the edges that previously connected every user transaction to the clock disappear, leaving only a small system-to-system clique. Using the same LSP/χ metric as Biton & Friedman [6] and matching their load regimes, our W-only estimate of achievable parallelism is roughly 30–40% tighter at routine load ($\approx 1\times$ vs. their 1.2–1.4×) and trims their high-load ceiling from $\approx 5\times$ toward $\approx 3\times$. At the application level, a union-find grouping over write co-occurrence identifies the DeepBook ecosystem (a giant component containing 87.3% of all application-level conflicts), and shows that DeepBook contributes to contention strictly by transaction volume rather than by uniquely contention-inducing logic, a surprising outcome for a single shared resource like a central limit order book. Finally, mapping graph structures to USD-denominated balance flow shows that 10–50% of network value remains gated by sequential execution paths (with the remaining 50–90% flowing through parallel paths), bounding the share of value that is potentially currently exposed to reordering-based MEV strategies.

Our primary contributions are as follows:

- **Semantic re-evaluation of conflict modelling in Sui using a write-only conflict model.** We leverage a W-only conflict model whose edges correspond to guaranteed write-serialisation events that Sui's engine must preserve, and show that the R+W conflict

model inherited from account-based systems systematically introduces non-execution dependencies under Sui's object-centric model. We propose a complementary W-only model aligned with mutable shared-object serialisation semantics.
- **Application-level partitioning via union-find.** We identify the DeepBook ecosystem in the conflict graph and compare the contention structure of DeepBook-touching versus non-DeepBook subgraphs.
- **Mapping graph structure to economic flow.** We map graph structures to USD-denominated balance flows, measuring the fraction of network value gated by sequential execution paths.

2 Background

In this section, we present the background concepts and definitions that our analysis relies on. We first describe Sui's object-centric execution model, including the versioning rules that govern parallel mutability and serialisation, which together are the foundation of its parallelism (Section 2.1). This motivates our choice of a write-only conflict model for measuring contention (Section 2.3). Finally, we define the graph metrics we use to quantify parallelism (Section 2.2).

2.1 Sui's Object-Centric Execution Model

In Sui [8], every on-chain entity, such as a coin, a token, a smart contract's state, a deployed package, etc., is an *object*. Each object carries a globally unique 256-bit identifier that is fixed for its lifetime, a version number, and an owner, and is a typed instance of a Move resource. Sui exposes three ownership classes that differ in how transactions may access them: *address-owned* objects, reachable only through a single account; *immutable* objects, frozen and readable by anyone; and *shared* objects, mutable and reachable concurrently by any transaction. This object-level granularity is the foundation of Sui's parallelism: two transactions whose object sets are disjoint carry no execution dependency and may run on separate cores. Whether two transactions that share an object must be serialised depends on how each of them accesses it. Sui coordinates this through a form of deterministic *multi-version concurrency control* (MVCC) in which object versions are assigned by the consensus engine, Mysticeti [5], rather than at execution time, and distinguishes three primary access modes:

1. **Owned/Immutable:** These objects never trigger serialisation across transactions, since an owned object has a single writer by construction and an immutable object can only be read. The access pins to whichever version is current at consensus-sequencing time and does not advance the version counter.
2. **Shared (Read-Only):** Many transactions can read the same shared object concurrently without blocking, and such reads leave the version counter untouched.
3. **Shared (Mutable):** This is the only access that can induce an execution dependency. Two transactions that both write the same shared object must execute sequentially, and a mutating access consumes the object's current version v and produces version $v{+}1$, advancing the counter.

Because every version an access will read is fixed at consensus time, Sui needs no read locks. Readers of an object obj at version v (written obj@v) are mutually independent, and a reader of obj@v is independent even of the writer that produces obj@$v{+}1$, since the two touch different versions; the execution scheduler therefore treats an immutable access as a literal no-op.

User activity reaches the network as *programmable transaction blocks* (PTBs): an atomically executed sequence of commands (e.g., Move calls, coin splits and merges, object transfers) that either commits in full or aborts with no effect [16]. A PTB does not discover state at run time; it declares each input object up front, together with the version it will read. Its read and write sets are thus fixed before execution and can be reconstructed statically from a transaction trace, the property on which our analysis rests.

A transaction's route through the network follows from the objects it declares. One whose inputs are all owned or immutable takes the *fast path* and is certified directly by validators, without consensus; one declaring any mutable shared object takes the *consensus path* and is sequenced by Mysticeti against all competing transactions. Because only the consensus path can serialise transactions of distinct users, it is the conflict structure of that path that the rest of the paper measures.

Executed transactions are finalised and recorded in *checkpoints*. Unlike a conventional block, a checkpoint is assembled *after* execution: the checkpoint builder takes a contiguous run of consensus-ordered transactions, closes it under causal dependency, and certifies it with a quorum of validator signatures [6]. Each checkpoint carries a monotonically increasing sequence number and is irreversible once certified. Since the Mysticeti upgrade, Sui emits roughly four checkpoints per second, and one checkpoint may bundle several consensus commits, each contributing a system *prologue* transaction that advances on-chain system state (Section 2.3). A checkpoint is thus a finalised, deterministically ordered batch of transactions with fully known object accesses, the natural unit at which to study execution dependencies; following Biton & Friedman [6], we build one conflict graph per checkpoint.

2.2 Graph Properties and Structures

We use multiple graph properties to characterise our conflict graphs and determine potential parallelizability. These metrics are adapted from prior work [7].

- **Density:** The ratio of existing edges to the maximum possible edges, measuring how closely a graph approaches a complete graph.
- **Diameter:** The longest shortest path between any two nodes in a connected graph.
- **Chromatic number (χ):** The minimum number of colours needed to colour the nodes so that no two adjacent nodes share the same colour.
- **Clique number:** The number of nodes in the largest complete subgraph.
- **Longest simple path (LSP):** The length of the longest path that does not repeat any visited nodes.
- **Largest connected component (LCC):** The size of the connected subgraph with the highest number of nodes.

When parallel execution is not possible, Sui switches to sequential execution. The **LCC** provides the upper bound of sequential dependency for concurrently accessed transactions, while the **LSP** provides the lower bound of the sequential chain. We identify two primary structural archetypes:

- **Clique-like:** A near-complete subgraph formed when multiple transactions write to the same hot object(s), creating dense conflict edges.
- **Hub-and-spoke (Star-like):** A centralised structure where a "hub" transaction writes to multiple hot objects, while "spoke" transactions write to a single hot object, connecting them to the hub but not to each other.

2.3 Conflict Modelling: R+W vs. W-Only

Prior empirical analyses of Sui [6] adopt a *read+write* (R+W) conflict model inherited from account-based blockchains such as Ethereum. Under this model, two transactions T_1 and T_2 conflict if they touch the same object and at least one of them writes. Formally, two transactions T_1 and T_2 are connected by an edge in the R+W conflict graph iif

$$(\texttt{write}(\mathsf{T}_1) \cap (\texttt{read}(\mathsf{T}_2) \cup \texttt{write}(\mathsf{T}_2))) \cup (\texttt{write}(\mathsf{T}_2) \cap (\texttt{read}(\mathsf{T}_1) \cup \texttt{write}(\mathsf{T}_1))) \neq \emptyset$$

where $\texttt{write}(\mathsf{T}_i)$ denotes the write set and $\texttt{read}(\mathsf{T}_i)$ denotes the read set of transaction T_i. Because Sui's scheduler does not serialise on shared-object reads, an R+W graph contains edges that the engine never has to honour, such as every read of the system clock `0x6`. R+W therefore provides an *upper bound on contention*.

In this paper, we adopt a *write-only* (W-only) conflict model as a complementary view: two transactions conflict only if they both write the same object. Formally, two transactions T_1 and T_2 are connected by an edge in the W-only conflict graph iif

$$\texttt{write}(\mathsf{T}_1) \cap \texttt{write}(\mathsf{T}_2) \neq \emptyset$$

Every edge in our graph corresponds to a guaranteed write-serialisation event that Sui's engine must preserve under the mutable shared-object semantics, so W-only provides a *lower bound on contention* and a *practical upper-bound on achievable parallelism*.

Advantages of the W-Only Model: No Artificial Read Dependencies

Consider two transactions sequenced adjacent to each other in a single checkpoint, both touching a shared object obj at current version v_0:

$$\mathsf{T}_{r_0} : \text{read } \mathsf{obj}@v_0 \qquad \text{and} \qquad \mathsf{T}_{w_1} : \text{write } \mathsf{obj}\ (v_0 \to v_1).$$

Under Sui's MVCC model, both transactions pin to $\mathsf{obj}@v_0$ as input. The scheduler emits no dependency from T_{r_0}'s read, and the barrier it emits for T_{w_1}'s write is only against *other writers* of obj. The two transactions execute in parallel. An R+W graph places an edge between them; a W-only graph does not.

W-only edges align directly with the four Sui subsystems that enforce serialisation, all of which key off the exclusive-access (`Mutable`) flag [22]:

- **Execution scheduler:** `Mutable` access emits barrier dependencies; `NonExclusiveWrite` emits ordinary dependencies; `Immutable` access emits no edges at all [18].
- **Shared-object version manager:** Only exclusively-accessed inputs advance the per-object version counter. Read-only accesses do not bump the version [20].
- **Per-object congestion control:** Execution cost is accumulated only for exclusively-accessed inputs. Read-only accesses contribute zero cost and cannot cause deferral [19].
- **Checkpoint-level congestion accounting:** The filter explicitly drops read-only inputs and keeps only mutated objects for downstream processing [17].

By stripping read-side edges, the W-only model isolates application-driven contention from read-only hot system objects. The clearest example is the system clock (`0x6`). In a read+write model, almost every transaction in a checkpoint "conflicts" with the clock because they all read it for timestamps. This creates a massive, artificial "star" structure in the conflict graph. Each consensus commit emits a single *prologue* transaction that mutates the clock, and Sui's checkpoint builder aggregates multiple consensus commits into a single checkpoint. A typical checkpoint therefore contains several prologue transactions, each writing `0x6`.

Under R+W, the resulting picture is a giant hub-and-spoke web: every prologue is a hub, and every user transaction in the checkpoint is a spoke, because all of them read the clock for timestamps. Under W-only, the spokes disappear, user reads do not produce edges, and what remains is a small clique of prologue-to-prologue writes. The character of the conflict is now completely different: the clock no longer appears to contend with user transactions, only with itself. This lets us cleanly separate clock activity from application contention and quantify it: in Section 5 we report that this system-to-system clique accounts for 68% of W-only conflict events, isolating it as pure system housekeeping with no user-facing impact on parallelism.

The same logic applies to other per-commit system objects, such as the randomness state (`0x8`) and the accumulator root (`0xacc`) [21], whose system-only writers form analogous prologue-style cliques when checkpoints aggregate multiple commits.

Limits of the W-Only Model: Write-to-Read Forwarding

There is one dependency class where the W-only model may under-count: write-then-read *forwarding* within a single checkpoint. Consider the sequence:

$$\mathsf{T}_{w_1} : \text{write } \mathsf{obj}\ (v_0 \to v_1) \quad \Longrightarrow \quad \mathsf{T}_r : \text{read } \mathsf{obj}@v_1 \quad \Longrightarrow \quad \mathsf{T}_{w_2} : \text{write } \mathsf{obj}\ (v_1 \to v_2)$$

The W-only graph captures the structural $\mathsf{T}_{w_1} - \mathsf{T}_{w_2}$ edge. However, the actual execution DAG may also contain $\mathsf{T}_{w_1} \to \mathsf{T}_r$, because T_r pins to a version that only exists once T_{w_1} finishes. Whether this $\mathsf{T}_{w_1} \to \mathsf{T}_r$ edge actually constrains execution depends entirely on runtime timing. If T_{w_1} completes before T_r is scheduled, no wait occurs; T_r consumes the materialized version and runs in parallel with T_{w_2}. If T_{w_1} is slow, T_r blocks. Because the static checkpoint trace cannot reveal this runtime timing, the W-only graph conservatively excludes these conditional edges. The W-only model is therefore best understood as the conservative core of guaranteed orderings: it overcounts no edges, though it may undercount these conditional forwarding chains. We treat W-only and R+W as complementary bounds throughout the remainder of the paper.

3 Methodology

3.1 Data Acquisition and Pipeline

Transaction data was collected from the mainnet checkpoints into our PostgreSQL [3] database using a custom indexer written in Rust and Diesel [1]. Sui produces around 4 checkpoints per second, corresponding to approximately $345,600$ checkpoints per day. To collect a representative, long-term dataset across the history of the network without overwhelming our pipeline, we deterministically sampled every 3455th checkpoint. At post-Mysticeti checkpoint production rates, this corresponds to approximately 100 checkpoints per day. Earlier periods of the blockchain operated at a lower checkpoint frequency, around ~ 1 cp/s, resulting in fewer sampled checkpoints per day in the pre-Mysticeti era. This stride value was fixed globally rather than recalculated per day.

Once indexed, we extract the write-sets and construct the undirected conflict graphs for each checkpoint using NetworkX [2] in Python. The full database schemas and the SQL used to extract the write sets are reported in the full version of the paper.

3.2 Quantifying Execution Parallelism

To evaluate the potential for parallelism within Sui checkpoints, we employ two primary metrics adapted from prior works on Ethereum [7, 15].

First, we use the *Longest Simple Path to Chromatic Number Ratio (LSP/χ)*. This graph-based metric bounds the achievable parallel execution of a given checkpoint [14]. The chromatic number (χ) represents the theoretical minimum number of execution rounds required to process the graph without conflicts. The longest simple path (LSP) represents the largest unavoidable sequential dependency chain. Because finding the exact χ, clique number, and LSP are NP-hard problems, we estimate them using heuristic algorithms: χ is approximated via NetworkX's greedy colouring algorithm, the clique number via a greedy heuristic, and the LSP via a heuristic depth-first search. While heuristic estimates may not give the exact values, our study focuses on relative comparison between R+W and W-only under identical estimation procedures, reducing systematic bias.

Second, we use *Gas Usage* to weight these structural limits by computational cost. Gas usage represents the actual work done to execute transactions. By dividing the total gas used in a checkpoint by the gas consumed along the largest sequential path (the largest connected component or max clique), we estimate the potential speedup factor if all independent transactions were executed perfectly in parallel. This gas-weighted variant is adapted from Heimbach et al. [15], who applied it to Ethereum. The count-based LSP/χ ratio used by prior work on Sui [6] treats every transaction equally: a 1M-gas transaction contributes the same as a 1K-gas one. This is appropriate for asking "how many parallel rounds do we need?" but misleading for "how much wall-clock time can we save?". Gas-weighting is more faithful to actual execution cost, and more conservative against long sequential chains composed of cheap transactions: a long chain of 1K-gas transactions is correctly downweighted relative to a short chain of expensive ones.

3.3 Application Isolation and Economic Modelling

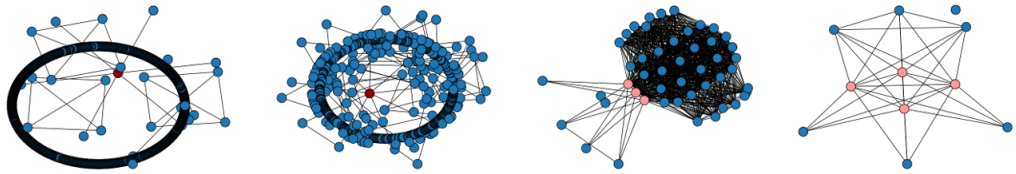
To answer our research questions regarding application-level contention and economic cost, we apply two specific modelling techniques to the base conflict graphs.

Union-Find Object Grouping

To isolate the impact of specific smart contracts (like DeepBook), we create union-find (disjoint set) groups over objects based on write co-occurrence. An object obj_1 is related to obj_2 if a transaction T_1 writes to both. This relationship is transitive; two objects obj_1 and obj_2 may end up in the same group via a chain of shared transactions, even if no single transaction touches both directly. This grouping provides an upper bound on the objects jointly involved in an application's execution ecosystem, which we use in Section 5 to partition the conflict graph.

Balance-Change and Price Anchoring

To measure the real-world economic value constrained by sequential execution, we analyse changes in coin balances using Sui's `derive_balance_changes` function. This function tracks value by subtracting custom fungible input coins, adding mutated/output coins, and tracking address balance changes from accumulator events. It focuses on fungible value, explicitly excluding NFTs or bespoke non-Coin objects.



(a) Checkpt 8,710,000 (733 txs) (b) Checkpt 33,190,000 (494 txs) (c) Checkpt 65,982,000 (168 txs) (d) Checkpt 149,483,500 (41 txs)

Figure 1 Biton & Friedman's R+W checkpoint conflict graphs visualised [6].

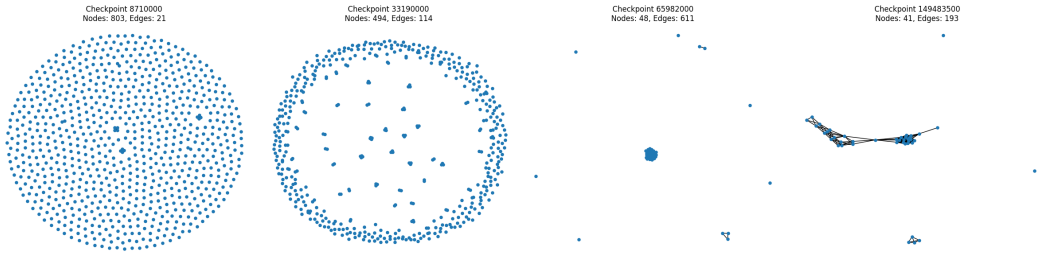


Figure 2 Our W-only conflict graphs visualised in the same checkpoints.

By anchoring these balance changes to historical CoinMarketCap price data, we can calculate the USD-denominated value processed by any given transaction. We aggregate these values into two buckets: transactions that lie on the sequential burden (the Largest Connected Component) and those that can be executed in parallel.

4 Conflict Graph Structure: Write-Only vs. Read+Write

The transition from a read+write (R+W) to a write-only (W-only) conflict model fundamentally alters the perceived structure of the Sui network. In this section, we characterise these structural shifts and explain why they yield a tighter bound on potential parallelism.

4.1 Structural Topology: From Stars to Cliques

The most striking difference between our W-only results and the prior R+W analysis by Biton & Friedman [6] is the shift in *assortativity*. In a read+write model, Sui conflict graphs appear highly disassortative, characterised by "stars" where many transactions conflict with a single central hub. As mentioned in Section 2, this artefact is explained almost entirely by the behaviour of system objects, such as the system clock (`0x6`). In an R+W analysis, the clock is a massive hub because virtually every transaction reads it for a timestamp, while only system transactions write to it. This creates a false-positive conflict edge between user transactions and the clock.

By utilising a W-only model, we remove these read-side edges, causing the artificial star structure to collapse. The user transactions disappear from the clock's component entirely. What remains are the true execution dependencies: multiple transactions attempting to modify the same application state.

Consequently, our W-only graphs are highly assortative. This suggests that actual execution contention on Sui is not a "hub-and-spoke" phenomenon, but rather a "clique-like" one. Transactions do not cluster around a single reader; they conflict in small, dense groups that are fundamentally unparallelizable.

Table 1 Comparison between Biton & Friedman [6] R+W and our W-only conflict graphs.

Metric	Biton & Friedman [6] (R+W)	This paper (W-only)
Median density	0.471	0.092
Median assortativity	-0.500	$\approx +1$
Median clique number	5.5	4
Median LCC	8.5	4
Median LSP/χ lower bound	1.271	0.667
Median LSP/χ upper bound	1.5	1

Figure 2 shows the comparison of Biton & Friedman's read+write conflict graphs with our write-only conflict graphs for the same checkpoints. The dense "hub-and-spoke"" structures observed in the read+write graphs largely disappear in our write-only graphs, indicating that much of the apparent contention originates from shared read access.

4.2 Quantifying the Lower Bound

Because we strip the non-execution read-edges, our measured contention is lower (fewer edges), but the resulting sequential chains are strictly execution-bound. This yields a sound lower bound on the contention Sui's scheduler must honour, and equivalently a (near-)upper bound on the speedup it could achieve.

Table 1 compares the core graph metrics between the two models. Our W-only conflict graphs exhibit lower median density, clique number, and Largest Connected Component (LCC). Crucially, the topology shifts from disassortative (high-degree nodes connected mostly to low-degree ones, the signature of hub-and-spoke stars) to highly assortative (high-degree nodes connected to each other in clique-like clusters). The corresponding degree-assortativity coefficient jumps from -0.500 to $\approx +1$, quantifying the elimination of the read-induced stars.

By evaluating the Longest Simple Path to Chromatic Number (LSP/χ) ratio across matched load regimes, we find that optimal parallel scheduling is limited under routine workloads. Biton & Friedman [6] report their 3–5× figure on the high-load 66M interval and on a synthetic 100×-aggregated 150Mx100 workload; on their routine 150M interval the same LSP/χ metric yields only $\approx$ 1.2–1.4×. Our W-only analysis gives $\approx 1\times$ at routine load and reaches up to $\approx 3\times$ at the largest checkpoints. In other words, stripping read-side edges shaves 30–40% off the optimal-parallelism estimate at routine load and trims the high-load ceiling from $\approx 5\times$ toward $\approx 3\times$.

Finding (RQ1). The "star" structures reported in prior Sui studies are largely artefacts of shared read access to the system clock. Under a write-only model, those edges disappear, and Sui's conflict graphs are clique-dominated; the median degree-assortativity flips from -0.5 under R+W to $\approx +1$ under W-only.

Finding (RQ2). At matched load regimes, the W-only estimate of achievable parallelism is $\approx$ 30–40% tighter than the R+W estimate: routine-load headroom drops from $\approx 1.4\times$ to $\approx 1\times$, and the high-load ceiling drops from $\approx 5\times$ toward $\approx 3\times$.

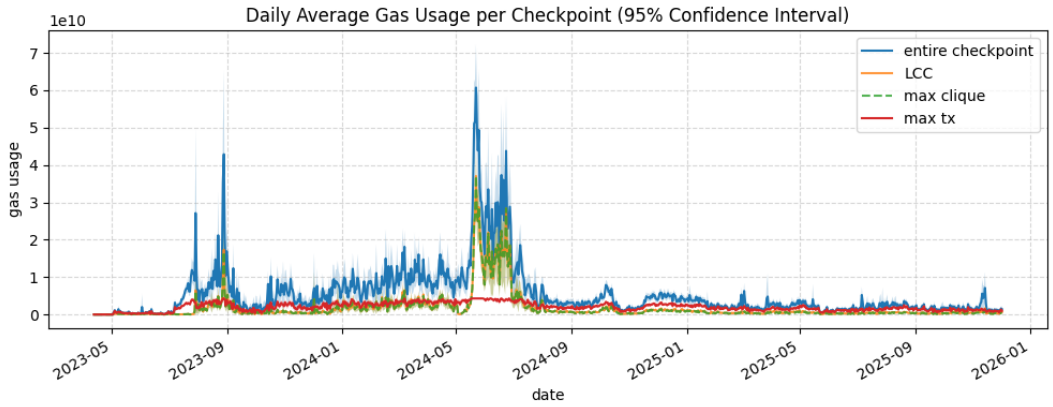


Figure 3 Historical development of gas used compared to entire checkpoint, clique number, max connected component, and max tx.

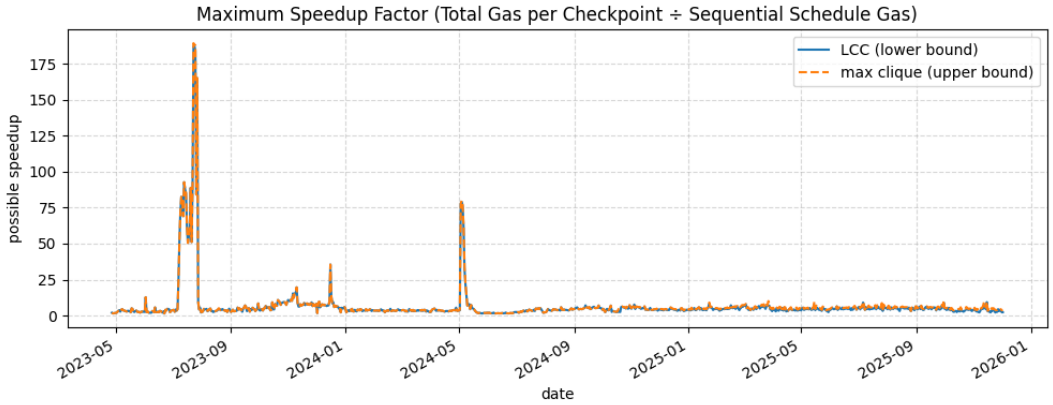


Figure 4 Lower and upper bounds of possible achievable execution speedup through parallelisation over the history of Sui.

4.3 Parallelizability Over Time via Gas-Based Analysis

To ground these graph-theoretic limits in computational cost, we consider parallelisation potential weighted by Gas Usage, adapting methods used in related work [15].

The largest connected component (LCC) represents the largest subset of transactions that must execute sequentially, forming the upper bound of a schedule. The clique number represents the lower bound of a schedule, as it forms a complete subgraph. We define LCC as the lower bound and clique number as the upper bound of the achievable speedup in this gas-based analysis.

As shown in Figure 3, the gas used by the maximally connected component and the maximal clique frequently overlap and approach the total gas of the entire checkpoint. This indicates the presence of short-lived workloads where write-hot objects dominate the entire checkpoint, a phenomenon made possible by Sui's small checkpoint sizes.

Figure 4 translates this into potential speedup factors. Consistent with our graph metrics, the possible speedup remains low and stable throughout most of Sui's history. The bounds are nearly indistinguishable. However, anomalous spikes do occur, most notably in June 2023, where potential speedup exceeded 175×. This spike correlates heavily with viral usage

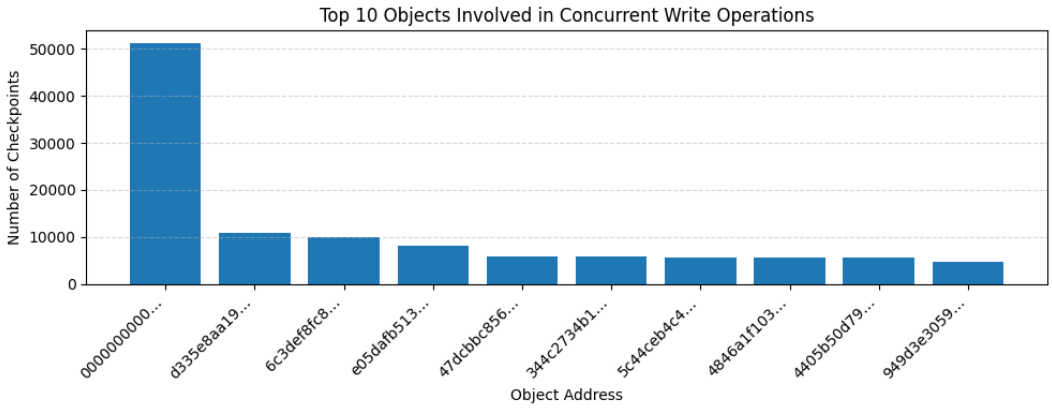


Figure 5 Top 10 objects involved in concurrent write access. The leftmost bar corresponds to the $0x6$ system clock.

of a fully on-chain game (Sui 8192) and the Bullshark Quests initiative [10]. During such periods, the massive influx of transactions interacting with a specific smart contract's shared objects temporarily shatters the typical tight bounds on parallelism.

Taken together, these two figures sharpen the count-based picture from Section 4.1. The gas-weighted bounds confirm that routine Sui checkpoints have effectively no residual parallelism to exploit: a small number of write-hot objects pull most of the checkpoint's gas into a single sequential chain, ruling out the "many cheap transactions strung along a long chain" loophole that count-based metrics could hide. The viral-period exceptions are the corollary: when the workload itself contains parallel structure, with independent users hitting disjoint objects, Sui's engine extracts very large speedups (over 175× in June 2023). The low routine-load number is therefore a statement about workload composition, not about the scheduler's ability to exploit parallelism.

Finding. The bottleneck on routine Sui load is workload composition, not the parallel engine. When the workload genuinely contains parallel structures (e.g. the Sui 8192 viral episode in June 2023, with achievable speedup of $\approx 175\times$), the engine extracts very large speedups; on routine load, it simply has little to parallelise. Operators should size for tail behaviour rather than average behaviour.

5 Application-Level Contention: The DeepBook Slice

A central question for any parallel blockchain is whether execution contention is diffused across the network or concentrated in a few highly utilised applications. To answer this, we must first separate conflicts caused by the blockchain's internal housekeeping from conflicts driven by user activity.

5.1 System vs. Application Contention

Before analysing specific decentralised applications (dApps), we establish a baseline by identifying which objects appear most frequently in our conflict graphs. Figure 5 lists the top 10 objects involved in concurrent write accesses across all analysed checkpoints, and Figure 6 shows the smart contracts associated with these top 10 objects.

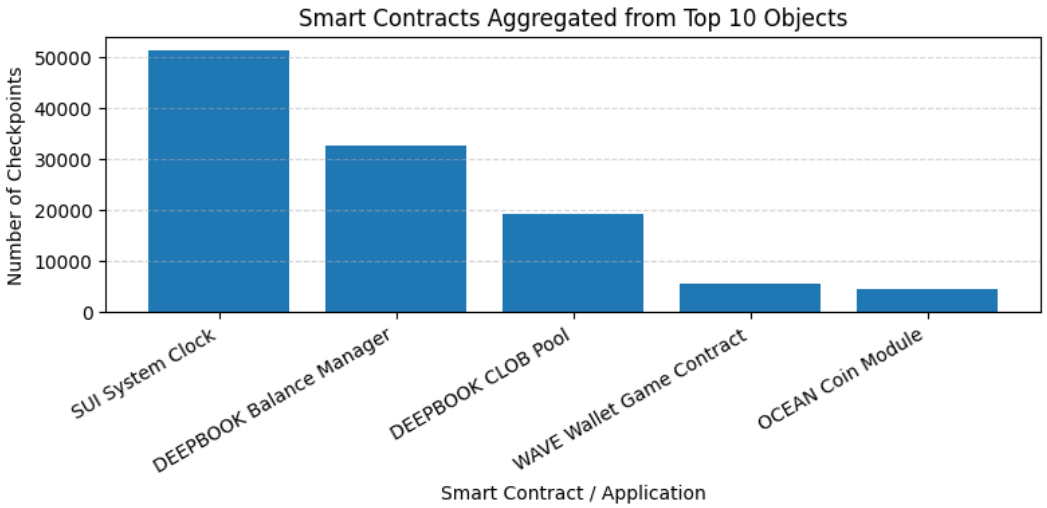


Figure 6 Smart contracts/applications associated with the top 10 objects involved in concurrent write access.

The most notable observation is the absolute dominance of Sui's system clock (`0x6`).

As discussed in Section 4.1, the clock is updated by system-generated prologue transactions. Our analysis reveals that the system clock is involved in more than 68% of all checkpoints exhibiting conflicts.

Because only system transactions are permitted to write to the `0x6` object, this finding places a strict upper bound on application-level contention. It indicates that the vast majority of "structural" contention in the Sui network is driven by consensus mechanics and internal state management, rather than by the logic of user-facing smart contracts. User-initiated application contention therefore accounts for, at most, the remaining 32% of write conflicts on the network.

Crucially, this 68% figure is best understood as a measurement of Sui's *design choice* to expose system state on chain via shared objects mutated by visible system transactions, rather than as a measurement of workload-driven contention. The same observation applies to the randomness beacon (`0x8`), the accumulator root (`0xacc`), and any other system objects (Section 2). A functionally equivalent design that exposes time and randomness as validator-side metadata available to smart contracts via a built-in primitive (as Ethereum does with `block.timestamp`, or Solana with its sysvars) would produce no prologue transactions, no writes to these system objects, and no system-to-system cliques in any execution trace. Application-driven contention on the same workload would be measured identically; what would vanish is the housekeeping share that is currently visible only because Sui chose to make it visible. We therefore treat the application-only subgraph (with the system objects of Section 2 stripped) as the cross-system-meaningful metric throughout the remainder of the paper.

Finding. Most (68%) of write-object accesses in our write-set modify a system object (the clock), with similar shares for the randomness beacon and other system objects. Under a design that handles these as validator-side metadata rather than on-chain mutable objects, this housekeeping contention would not appear in any trace; it is an artefact of Sui's specific implementation, not of the workload. Application-level contention accounts for at most the remaining 32% of write conflicts.

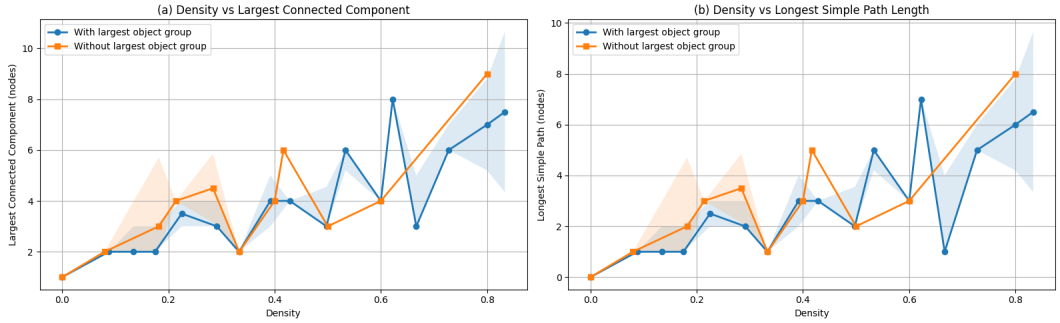


Figure 7 (a) Largest connected component (LCC) of conflict graphs, comparing checkpoints whose transactions touch the DeepBook object group against checkpoints that do not, as a function of graph density. (b) Longest simple path (LSP) of conflict graphs, comparing checkpoints whose transactions touch the DeepBook object group against checkpoints that do not, as a function of graph density.

5.2 Probing DeepBook's Role

Having isolated the 32% of the conflict graph driven by user applications, we turn our attention to the most prominent candidate for concentrated contention: DeepBook, Sui's native central limit order book (CLOB). We probe its role through two complementary experiments: a *partition probe* that compares conflict structure on DeepBook-touching checkpoints against the rest, and a *counterfactual probe* that simulates a perfectly parallel DeepBook by removing its internal conflict edges.

Partition probe: DeepBook vs. non-DeepBook checkpoints

To isolate the impact of DeepBook, we utilise the union-find grouping strategy detailed in Section 3. By clustering objects based on write co-occurrence, we identified a single "giant component" of objects that accounts for over 92% of the total object write relationships on the network. By cross-referencing the object IDs in this group with known DeepBook pool IDs, we confirmed that 18 out of 21 active DeepBook V3 pools are contained within this single component. This confirms that our "giant component" effectively captures the vast majority of DeepBook's on-chain execution ecosystem. This partitioning allows us to divide our conflict graphs into two distinct sets: checkpoints containing transactions interacting with the DeepBook ecosystem, and checkpoints that do not. Our analysis of these partitions reveals a nuanced picture regarding sequential burden. We compared the largest connected component (LCC) and the longest simple path (LSP) across both sets. As shown in Figure 7, the two sets exhibit nearly identical scaling behaviour as graph density increases: the DeepBook-touching checkpoints do not display longer sequential chains or larger connected components than the non-DeepBook ones at any given density.

This leads to a critical observation. DeepBook undeniably dominates Sui's application conflict graph by volume, participating in 87.3% of all application-level conflicts. However, its contribution to the sequential bottleneck is strictly proportional to its massive transaction volume. It does not exhibit uniquely "unparallelizable" logic compared to the rest of the network.

This is an interesting result for a central limit order book, which is conceptually a single shared resource: in an account-based execution model, a CLOB is naturally implemented as a single contract owning both the global order book and the balance ledger, so every order

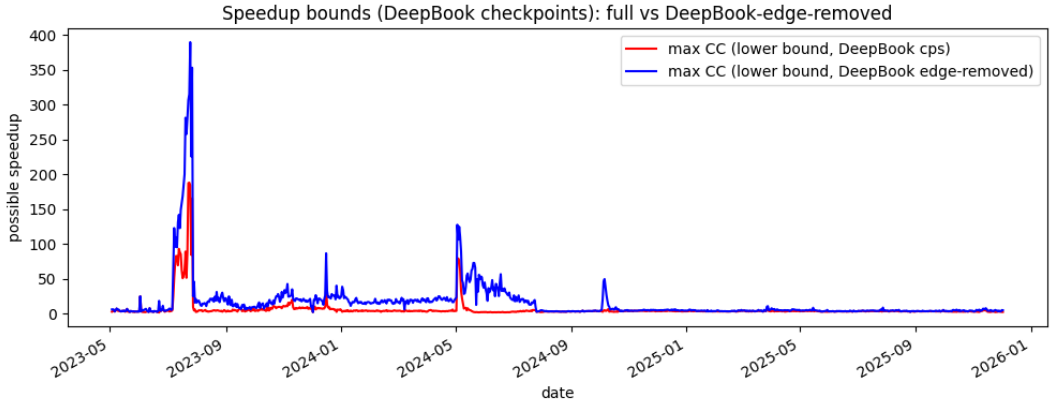


Figure 8 Potential speedup of the W-only conflict graphs with and without the DeepBook object group's internal conflict edges, across Sui's full history.

placement, match, or cancellation mutates the same contract storage and forces serialisation on every operation. The pattern is well-documented in Ethereum DEX measurements [15]. DeepBook's design instead exploits Sui's object model by sharding its state across separate pool objects (one per trading pair) and per-user balance manager objects, so transactions touching disjoint pools or disjoint balance managers can execute in parallel. The conflict structure we measure, contention that scales with volume but no faster than other high-volume applications, is the empirical signature of this sharding strategy. A direct head-to-head measurement against an Ethereum-style CLOB is out of scope, but the architectural argument suggests Sui's object model converts what would otherwise be a hot-contract star into a workload whose only bottleneck is volume.

Counterfactual probe: removing DeepBook's internal conflicts

To empirically verify that DeepBook's design is not an inherent bottleneck, we performed a counterfactual "edge-removal" experiment. In this scenario, we manually removed all conflict edges between transactions that write to the DeepBook object group. This effectively simulates a network where the DeepBook protocol is executed with "perfect" parallelism, free of any internal state contention.

As shown in Figure 8, the resulting increase in potential speedup across the network is only moderate, typically hovering around 20%–40%. The only exceptions are during extreme, anomalous load spikes where the entire network's capacity is stressed. Figure 9 restricts the same comparison to the 2025 period and shows the same pattern at a finer time resolution. This experiment reinforces our conclusion regarding Sui's object-centric architecture. DeepBook's application state is highly distributed across many distinct objects, such as individual liquidity pools and user balances. While thousands of transactions interact with the DeepBook protocol simultaneously, they frequently touch disjoint objects, allowing Sui's execution engine to process them in parallel. The sequential chains that do inevitably form are the natural result of high transaction density in a shared market, rather than a single, poorly designed "bottleneck" object within the DeepBook logic itself.

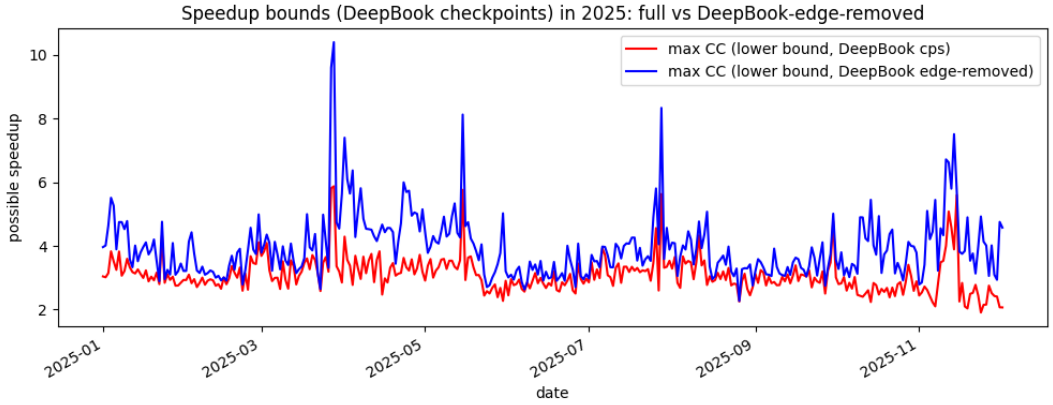


Figure 9 Potential speedup of the W-only conflict graphs with and without the DeepBook object group's internal conflict edges, restricted to 2025.

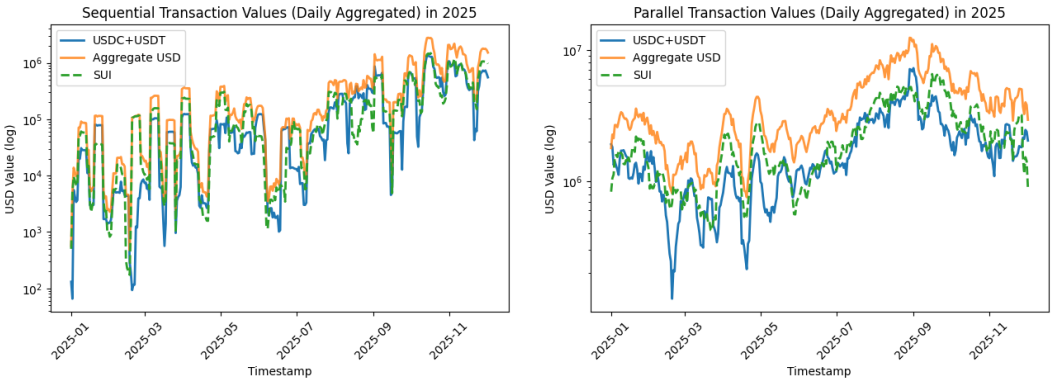
Finding (RQ3). DeepBook accounts for 87.3% of application-level conflicts on Sui by volume, but neither the partition nor the counterfactual probe shows per-transaction contention exceeding that of other high-volume applications. This is surprising for a central limit order book, normally a sequential bottleneck on account-based chains, and is explained by DeepBook's per-pool, per-balance-manager sharding of state under Sui's object model.

6 The Economic Cost of Sequentiality

To connect graph-theoretic contention to real-world economic flow, we measured the USD-denominated value processed in parallel versus sequential execution paths. We separate the flow of native SUI from stablecoins (USDC/USDT) to provide a dollar-denominated anchor for our analysis.

Figure 10 shows the economic value of sequential-burden and parallel transactions in 2025. We define "sequential-burden" transactions as those included in the largest connected component (LCC) of each conflict graph. Our findings suggest that native SUI transfers dominate the measured economic activity, with a correlation coefficient of 0.999 against the total USD-aggregated value. Interestingly, the value of parallel transactions appears more stable and larger in magnitude than that of sequential-burden transactions throughout the history of Sui. Figure 11 shows the fraction of transaction value that is subject to sequential execution over time. We find that this fraction consistently ranges between 0.1 and 0.5. This indicates that while Sui's parallel engine is highly effective at offloading simple transfers, a significant portion (up to 50%) of the network's economic throughput is gated by sequential dependency chains.

Beyond describing Sui's economic flow, this sequential-burden fraction has a direct interpretation in terms of how value is currently exposed to consensus ordering. Transactions outside the LCC are parallel: their outcomes are independent of any reordering of the consensus stream. Transactions inside the LCC are ordered with respect to other LCC members on shared state, and their outcomes can, in principle, depend on that order. The 50–90% of USD value flowing through parallel paths is therefore currently *not* subject to typical Maximal Extractable Value (MEV) strategies that rely on reordering existing transactions,



(a) Transactions included in the LCC. **(b)** Transactions excluded from the LCC.

Figure 10 Economic value of transactions included and excluded from the LCC of conflict graphs (2025).

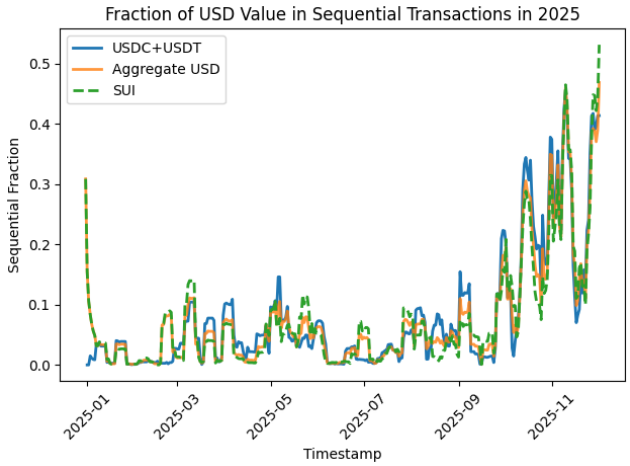


Figure 11 Fraction of transaction value subject to sequential execution (2025).

where MEV refers to the profit that validators can obtain by manipulating the ordering, inclusion, or exclusion of transactions within a checkpoint; the 10–50% flowing through the sequential burden currently *is potentially* subject to such strategies. This characterises only the workload as observed: a sophisticated MEV actor could submit additional transactions that introduce new conflicts and migrate some currently-parallel value into the sequential burden. Our numbers, therefore, describe the current state of the exposure of Sui's transaction value to ordering effects, not a universal MEV ceiling.

Finding (RQ4). In Sui's observed workload, 50–90% of USD-denominated value flows through parallel transactions whose outcomes do not depend on consensus ordering, and is currently outside the reach of typical reordering-based MEV strategies. The remaining roughly 0–50% flows through sequential-burden transactions whose outcomes are order-sensitive and are currently possibly affected by such strategies.

7 Related Works

This work builds on prior research on Ethereum transactions. The work of [7] used the concept of read+write conflict graphs to analyse how smart contracts caused the limitations of block throughput. They found that most graphs exhibit star-like structures, and it is common to have a single transaction connected to many other transactions. By utilising the graph-based longest simple path to chromatic number ratio, the results showed that Ethereum transactions can greatly benefit from parallel smart contract execution. Another work [15] analyses how decentralised finance (DeFi) and NFTs have limited concurrency of Ethereum, using gas-based analysis to show changes throughout the history of Ethereum. They showed that when transactions are disentangled, the possible speedup has increased dramatically after DeFi and NFTs were introduced. Our Sui's write-only analysis is not directly comparable to an Ethereum read+write analysis, as they have different execution models with different notions of "conflict". The work on [25] studied empirical concurrency in Ethereum smart contracts and found that the speedup has steadily declined from 2016 to 2017. However, the study was not able to determine which conflicts are real and which are artefacts. Further studies on other aspects of Ethereum include, but are not limited to, Dicker et al. adding concurrency to smart contracts [11], Gelashvili et al. parallel execution engine Block-STM [12], and Pejic et al. analytical study of large blocks on Ethereum [24].

Transaction execution frameworks have been widely explored in prior research. The work of [4] proposed an order-execute paradigm that supports concurrent execution of distributed applications. Another framework [24] supports deterministic distributed transaction execution by processing an identical batch of transactions across replicas, and resolves conflicts from the reservation mechanisms prior to the execution. Unlike traditional database systems, however, blockchain environments must tolerate Byzantine failures. To address this, this work leverages distributed proposing and verification, layered consensus protocols, and signature aggregation to achieve scalable and secure transaction execution [4]. To evaluate scalability under pressure, this work presented a framework to stress-test blockchain systems using decentralised applications such as gaming and web services. While the framework provides insight into blockchain performance under application-level workloads, it does not analyse transaction conflicts or constraints on parallel execution [13]. These works primarily focus on account-centric blockchains and optimistic concurrency control (OCC) execution models, whereas Sui adopts an object-centric architecture that has incorporated parallel execution directly into its design.

Sui Lutris [8] is a blockchain designed with sub-second finality with perpetual operation, using two approaches: Consensusless agreement and high-throughput consensus. The FastPath from the Mysticei consensus protocol [5] applies uncertified direct acyclic graphs to process transactions without or before reaching consensus by using reliable broadcast to commit transactions. The Mysticeti protocol is fundamental in shaping Sui's execution path.

Sui Lutris [8] is a high-performance blockchain designed for sub-second finality and continuous operation. Sui separates transaction processing into two execution paths: a fast path for transactions that involve only owned objects, and a consensus path for transactions accessing shared objects. The fast path leverages the Mysticeti consensus framework [5], which uses a DAG-based reliable broadcast protocol to achieve high throughput and low latency. This separation between owned-object execution and shared-object consensus is central to Sui's object-centric execution model, which directly shapes its parallelisation behaviour analysed in this work.

Biton & Frideman [6] perform a similar analysis on the Sui network using read+write conflicts, which, as mentioned previously, read+write conflicts do not capture the naive execution limits in Sui, as they capture read-only observational patterns. For example, transactions from MEV bots that watch-and-react will show up as many read edges in read+write conflict graphs. However, they are not true execution dependencies. Furthermore, the paper mentioned that the read set could be overestimated, causing the results to be a loose upper bound of possible parallelism, the set may include false dependencies, therefore, could overestimate sequentiality and underestimate achievable parallelism. This work focuses on write+write conflicts, yielding a sound, tighter bound, as the write-set contains dependencies that truly require ordering, hence, any remaining sequentiality is unavoidable. Each edge in our graph corresponds to a real, must-order dependency rather than a possibly read artefact. Any sequentiality is present and cannot be blamed on measurement noise in the read set.

8 Limitations

Our work contains several limitations. Firstly, we selectively sampled approximately 100 checkpoints per day in the post-Mysticeti era, and less per day in the pre-Mysticeti era. Our fixed-stride sampling method introduces temporal sampling bias between the pre and post-eras. A more rigorous approach would stratify sampling per calendar day rather than using a global checkpoint stride; Furthermore, Biton's paper [6] sampled the 1.5M full trace from a full node, some data may have been missed during our analysis. Secondly, we use greedy / DFS heuristics for the NP-hard graph properties (chromatic number, clique number, longest simple path), following the standard methodology in prior conflict-graph analysis on Sui and Ethereum [6, 7]. Exact computation is intractable at the scale of our checkpoint sample; the heuristic estimates may therefore deviate from the true values, but they remain directly comparable to the bounds reported in that prior work. Thirdly, the W-only model has two complementary blind spots relative to R+W. As described in Section 2, the W-only model treats *write-then-read forwarding* within a single checkpoint as outside its scope: when a transaction reads a version of a shared object produced by another transaction in the same checkpoint, the reader is a *candidate* execution-time consumer of the writer's output, yet the W-only conflict graph contains no edge between them. Whether the corresponding wait actually occurs depends on runtime overlap, which the consensus trace cannot resolve. Our LCC and LSP statistics may therefore under-count sequential chains in workloads with frequent intra-checkpoint forwarding, but never over-count. Lastly, our trace-based analysis cannot precisely determine which inferred dependencies are actually constrained during runtime execution. A possible extension would be to measure the frequency of candidate intra-checkpoint write-to-read ($W \rightarrow R$) forwarding relationships using indexed *object_changes*, where a shared object written by tx_i is later read at the new version by $tx_j > tx_i$ within the same checkpoint. Such relationships provide an upper bound on potential forwarding dependencies between transactions. However, the extent to which these dependencies truly reduce parallelism depends on runtime scheduling and execution timing information that is not captured purely in the checkpoint traces.

9 Conclusion

We have presented a complementary lower-bound analysis of contention in Sui based on a write-only conflict model that matches the engine's actual execution semantics. The W-only graph reveals a clique-dominated topology fundamentally different from the hub-and-spoke

structures reported under read+write models, and tightens the estimate of achievable optimal parallelism by roughly 30–40% at routine load. It also exposes that the network's largest application by transaction volume, DeepBook, contributes to contention strictly through its popularity rather than through contention-inducing smart-contract logic, a surprising property for a central limit order book that is made possible by Sui's object model and DeepBook's sharded state. Mapping contention to USD-denominated value further shows that 10–50% of network value flows through sequential paths, bounding the share of throughput currently exposed to reordering-based MEV. Beyond Sui, the W-only methodology could extend naturally to other modern blockchains that expose explicit atomic units of state (such as Aptos's resource-based execution under Block-STM or Solana's per-account writable-flag scheduling), where it offers a cross-platform vocabulary for comparing application-driven contention without the read-side noise that dominates account-centric R+W analyses.

References

1 Diesel: A safe, extensible orm and query builder for rust. `https://diesel.rs/`, 2024. Accessed: 2026-01-24.

2 Networkx: Network analysis in python. `https://networkx.org/`, 2024. Accessed: 2026-01-24.

3 Postgresql: The world's most advanced open source relational database. `https://www.postgresql.org/`, 2024. Accessed: 2026-01-24.

4 Mohammad Javad Amiri, Divyakant Agrawal, and Amr El Abbadi. Parblockchain: Leveraging transaction parallelism in permissioned blockchain systems. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 1337–1347. IEEE, 2019. `doi:10.1109/ICDCS.2019.00134`.

5 Kushal Babel, Andrey Chursin, George Danezis, Anastasios Kichidis, Lefteris Kokoris-Kogias, Arun Koshy, Alberto Sonnino, and Mingwei Tian. Mysticeti: Reaching the limits of latency with uncertified dags. *arXiv preprint*, 2023. `arXiv:2310.14821`.

6 Dvir David Biton and Roy Friedman. An analysis of sui's transactions and conflicts. In *2025 IEEE 30th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 79–90. IEEE, 2025.

7 Dvir David Biton, Roy Friedman, and Yaron Hay. Ethereum conflicts graphed. In *2025 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9. IEEE, 2025.

8 Sam Blackshear, Andrey Chursin, George Danezis, Anastasios Kichidis, Lefteris Kokoris-Kogias, Xun Li, Mark Logan, Ashok Menon, Todd Nowacki, Alberto Sonnino, et al. Sui lutris: A blockchain combining broadcast and consensus. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 2606–2620, 2024.

9 Vitalik Buterin et al. Ethereum white paper. *GitHub repository*, 1(22-23):5–7, 2013.

10 Cointelegraph. A simple tile game is spiking daily transactions on the sui network. `https://www.tradingview.com/news/cointelegraph:4d8bc2c51094b:0-a-simple-tile-game-is-spiking-daily-txs-on-sui-network-data-platform/`, 2023. Accessed: 2026-01-24.

11 Thomas Dickerson, Paul Gazzillo, Maurice Herlihy, and Eric Koskinen. Adding concurrency to smart contracts. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 303–312, 2017. `doi:10.1145/3087801.3087835`.

12 Rati Gelashvili, Alexander Spiegelman, Zhuolun Xiang, George Danezis, Zekun Li, Dahlia Malkhi, Yu Xia, and Runtian Zhou. Block-stm: Scaling blockchain execution by turning ordering curse to a performance blessing. In *Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, pages 232–244, 2023. `doi:10.1145/3572848.3577524`.

13 Vincent Gramoli, Rachid Guerraoui, Andrei Lebedev, Chris Natoli, and Gauthier Voron. Diablo: A benchmark suite for blockchains. In *Proceedings of the Eighteenth European Conference on Computer Systems*, pages 540–556, 2023. `doi:10.1145/3552326.3567482`.

14 Yaron Hay and Roy Friedman. Batch-schedule-execute: On optimizing concurrent deterministic scheduling for blockchains. In *2024 43rd International Symposium on Reliable Distributed Systems (SRDS)*, pages 163–174. IEEE, 2024. doi:10.1109/SRDS64841.2024.00025.

15 Lioba Heimbach, Quentin Kniep, Yann Vonlanthen, and Roger Wattenhofer. Defi and nfts hinder blockchain scalability. In *International conference on financial cryptography and data security*, pages 291–309. Springer, 2023. doi:10.1007/978-3-031-47751-5_17.

16 Mysten Labs. Sui Documentation. https://docs.sui.io/, 2026. Accessed: 2026-05-22.

17 Mysten Labs. Sui source: checkpoint-level congestion filter dropping `InputConsensusObject::ReadOnly` (`congestion_tracker.rs`, lines 100–104). https://github.com/MystenLabs/sui/blob/35abd5c201430ece1826e6fcf96ede6b4a756eeb/crates/sui-core/src/congestion_tracker.rs#L100-L104, 2026. Pinned to commit `35abd5c201`; accessed 2026-05-19.

18 Mysten Labs. Sui source: execution-scheduler match on `SharedObjectMutability` (`execution_scheduler_impl.rs`, lines 72–82). https://github.com/MystenLabs/sui/blob/35abd5c201430ece1826e6fcf96ede6b4a756eeb/crates/sui-core/src/execution_scheduler/execution_scheduler_impl.rs#L72-L82, 2026. Pinned to commit `35abd5c201`; accessed 2026-05-19.

19 Mysten Labs. Sui source: per-object congestion cost gate (`shared_object_congestion_tracker.rs`, line 223). https://github.com/MystenLabs/sui/blob/35abd5c201430ece1826e6fcf96ede6b4a756eeb/crates/sui-core/src/authority/shared_object_congestion_tracker.rs#L223, 2026. Pinned to commit `35abd5c201`; accessed 2026-05-19.

20 Mysten Labs. Sui source: shared-object version manager exclusivity filter (`shared_object_version_manager.rs`, line 464). https://github.com/MystenLabs/sui/blob/35abd5c201430ece1826e6fcf96ede6b4a756eeb/crates/sui-core/src/authority/shared_object_version_manager.rs#L464, 2026. Pinned to commit `35abd5c201`; accessed 2026-05-19.

21 Mysten Labs. Sui Source: System-Transaction Kinds and Their Mutable Shared-Singleton Inputs. https://github.com/MystenLabs/sui/blob/35abd5c201430ece1826e6fcf96ede6b4a756eeb/crates/sui-types/src/transaction.rs#L1955-L1985, 2026. Pinned to commit `35abd5c201`; `crates/sui-types/src/transaction.rs`, lines 1955–1985; accessed 2026-05-19.

22 Mysten Labs. Sui source: `SharedObjectMutability` enum (`sui-types/src/transaction.rs`, line 4629). https://github.com/MystenLabs/sui/blob/35abd5c201430ece1826e6fcf96ede6b4a756eeb/crates/sui-types/src/transaction.rs#L4629, 2026. Pinned to commit `35abd5c201`; accessed 2026-05-19.

23 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, 2008. Accessed: 2026-04-17.

24 Patrick Ocheja, Mikel Cortes-Goicoechea, Tarun Mohandas-Daryanani, Brendan Flanagan, Hiroaki Ogata, Jose Luis Munoz, and Leonardo Bautista-Gomez. An analytical study of large blocks on ethereum. In *Proceedings of the 2024 6th Blockchain and Internet of Things Conference*, pages 120–127, 2024.

25 Vikram Saraph and Maurice Herlihy. An empirical study of speculative concurrency in ethereum smart contracts. *arXiv preprint*, 2019. arXiv:1901.01376.

SAMM: Sharded Automated Market Maker

Hongyin Chen
Technion, Haifa, Israel

Amit Vaisman
Technion, Haifa, Israel

Ittay Eyal
Technion, Haifa, Israel

Abstract

The central challenge in addressing the security-performance trade-off in blockchain systems has moved from consensus protocols to enabling parallel execution. Various *smart contracts* (stateful blockchain programs) already benefit from parallelization. But we show that non-parallelizable operations within smart contracts remain a bottleneck. A natural approach is to apply *sharding*, that is, partitioning a *single logical service* into multiple identical parallelizable shards. However, we observe that this introduces game-theoretic issues in various applications: Participants may be incentivized to overload the system.

We focus on *Automated Market Makers* (*AMMs*), which enable token exchange: Traders swap tokens with the AMM for a fee, while *liquidity providers* pre-fund the AMM and receive these fees. We show experimentally that their demand will soon reach the single-thread performance limit.

We present *SAMM*, a sharded AMM system where multiple independent AMM shards operate on the same blockchain. The key challenge is ensuring SAMM's security against overloading and destabilization attacks. SAMM addresses this with careful mechanism design; we provide both game-theoretic analysis and simulation-based validation. We implement SAMM on Sui and Solana testnets, achieving 5x and 16x throughput improvements, respectively, with potential for more as blockchain parallelism advances. SAMM is deployable and has already attracted industry interest.

2012 ACM Subject Classification Security and privacy → Distributed systems security

Keywords and phrases Security, blockchain, automated market makers, game theory

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.11

Related Version *Full Version*: `https://arxiv.org/abs/2406.05568` [22]

Supplementary Material *Software (Source Code)*: `https://github.com/MountainGold/SAMM-Sui-Evaluation`, archived at `swh:1:dir:c5fa4a9c00c23d38762de20b654abbf301581ff3`
Software (Source Code): `https://github.com/MountainGold/spl-samm`
archived at `swh:1:dir:3b279df02c2cd754cbbdfa156c783c932aa68a1f`

Funding This work was supported in part at the Technion by Aly Kaufman Fellowship, IC3, the Sui Foundation, and the Avalanche Foundation.

1 Introduction

The trade-off between security and performance is a key challenge in blockchain systems. Recent advances in high-throughput consensus (e.g. [73, 28, 10]) have largely shifted the throughput bottleneck from consensus to smart-contract execution [86]. Modern blockchains like Sui [14], Aptos [49], and Solana [88] support parallel execution to address this challenge (though not Bitcoin [63] or Ethereum [84]). They automatically distribute independent transactions across multiple execution threads.

Practical smart-contract deployments adopt *sharding* [89, 46, 82], a classical throughput-scaling technique, to improve performance (§2). They trivially separate independent components of a contract into multiple independent instances called *shards*, and the underlying


© Hongyin Chen, Amit Vaisman, and Ittay Eyal;
licensed under Creative Commons License CC-BY 4.0

blockchain runs them in parallel threads. However, non-parallelizable operations within high-demand smart contracts remain a bottleneck. To the best of our knowledge, no prior work studies improving the throughput of non-parallelizable smart contracts via sharding.

In this work, we study the sharding of a *single non-parallelizable logical service* into multiple *identical* shards. For sharding to be effective, participants' demand should be evenly distributed across shards, without incurring additional overhead from splitting a single activity across multiple shards. But in decentralized systems we cannot assume participants behave benignly; security issues arise if participants deviate from the desired behavior.

We find that in several applications, such as *lending protocols* [61, 45, 35] and *centralized limit order books* (CLOBs) [74, 31], rational participants may be incentivized to launch overloading attacks: splitting their activity across multiple shards can increase their revenue (see the full version of our report [22]). This is not merely theoretical: In practice, users already split orders across multiple similar (but non-sharded) contracts to optimize revenue via so-called *routers* [8, 93].

The root cause of this security challenge is common in blockchain protocols [90, 43, 60, 19, 87] and applications [78, 56, 27, 21, 23, 62]: They are open to anyone and participants attack the system for more revenue. The resilience of the sharded contract stems from game-theoretic security [16]: Mechanism design prevents such attacks by eliminating additional revenue due to misbehavior.

Since different protocols have distinct incentive structures, mechanism design must be tailored to each protocol. The non-parallelizable part of many popular smart contracts is less than 1 *transaction per second* (*tps*; see the full version [22]), well below the single-thread performance limit. Therefore, we focus on addressing the only urgent context we are aware of: *Automated Market Makers* (AMMs) [38, 39]. AMMs allow users to immediately swap token pairs by maintaining *liquidity pools*: tokens of both types supplied by other users serving as *liquidity providers*. It requires sequential handling of transactions since the outcome of each transaction depends on the current state of the token-pair pool and, in turn, alters this state.

We show that AMM performance does not scale in two state-of-the-art blockchain systems, Sui [14] and Solana [88]. The throughput is limited by a single thread at $214tps$ in Sui (Figure 1, $n = 1$) and $129tps$ in Solana (Figure 3). However, AMMs' demand is growing exponentially and by 2029 neither system could satisfy the most popular token pairs' trading demand (see the full version [22]). This encourages races and lengthens the queue for AMM trades, widening the time window in which adversaries can manipulate transaction ordering and thereby exacerbating front-running attacks such as sandwiching [94, 52]. To overcome this, users defer workload to expansion systems called layer-2 protocols [55, 2]; but this entails additional security assumptions.[1] Moreover, the single-thread bottleneck still exists in these layer-2 systems (e.g., $138tps$ in ammBoost [57]).

We apply sharding to AMMs to improve their throughput. We model the system (§3) as a set of AMM shards with rational users of two types: traders, who purchase tokens and seek to minimize expenses, and liquidity providers, who deposit tokens and earn fees.

The shards are AMMs based on the standard *Constant Product Market Maker* (*CPMM*) contract: Roughly, the contract maintains the product of the two tokens constant after each trade. Thus, purchasing a larger amount of a token increases its unit cost, an effect called *slippage*. See the full version [22] for more details.

[1] Blockchains are secured by financially-invested nodes. A layer-2 protocol relies on an additional set of nodes, with significantly less at stake [29], making it vulnerable to cheaper attacks.

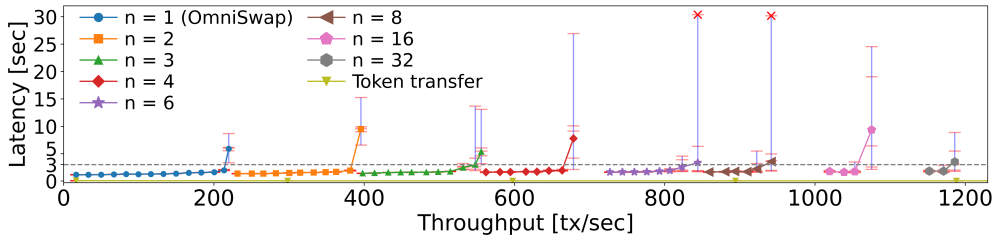


Figure 1 Sui trade latency as a function of demand with n SAMM shards.

We present SAMM (§4), an AMM protocol that uses multiple *shards* operating on the same blockchain, each with an independent liquidity pool. Ideally, all shards should be *balanced*, with the same liquidity; and traders should randomly select a shard to trade. Traders should not split trades across multiple shards, multiplying their overhead. Liquidity providers should not cause imbalances in shard sizes, making some shards less attractive to traders and reducing parallelism. Additionally, the system should re-stabilize if attackers unbalance the shards. Our goal is to disincentivize attacks that deviate from this ideal behavior, such as overloading the system by splitting trades across shards or concentrating trades in a subset of shards by unbalancing liquidity.

We prove that traditional CPMM shards are not secure. They charge a fixed-rate fee (e.g., 0.3% in Uniswap v2 [5]), which simply scales a trade's cost by a constant factor relative to the no-fee trade. Traders are incentivized to route a single trade across multiple shards to reduce slippage.

At the core of SAMM is a novel *trading fee function*. SAMM generalizes the fixed-rate fee to a non-constant rate, incentivizing traders to use the smallest shard and avoid splitting trades. It is a simple monomial family, which is easy to parameterize and to analyze.

The model gives rise to a game (§5) played by the users: In each step, either a liquidity provider adds liquidity to a subset of the shards, or a trader executes a trade using a subset of the shards. We assume myopic liquidity-provider behavior, under which the liquidity provider treats the revenue from the next trade as an estimation for long-term revenue. This reduces the analysis to a Stackelberg game. Myopic utility is a common assumption in prior work [69, 34], and we further justify it in Section 5.1. Our analysis shows that, in all best responses, traders use one of the smallest shards. This implies that when shards are balanced, traders will randomly select a shard to trade, as intended. We also prove that shards converge (e.g., following intentional destabilization) to a balanced state in all Subgame-Perfect Nash Equilibria (SPNE) and present a specific SPNE, showing that SAMM is robust against irrational destabilization attacks. Malicious attackers suffer ongoing losses, and when their budget runs out, the system returns to a balanced state.

Before validating the theoretical analysis with a simulation, we turn our attention to the performance of SAMM, which will inform this simulation. We implement the protocol and deploy it in Sui and Solana local test networks (§6). This local validation is not an oversimplification: Our setting suffices to demonstrate the improvement in execution time, which is the bottleneck of these modern blockchains, as they process consensus separately from execution. We test *tps* and latencies of trades under different demand rates and numbers of shards. In Sui, the throughput increase is limited by the serial elements of Sui's transaction processing, following Amdahl's Law. In Solana, the throughput increases linearly with the number of shards up to a limit due to the blockchain's constraints. SAMM achieves over a 5x throughput increase in Sui and a 16x increase in Solana, compared to a single-contract AMM.

Figure 1 shows that with more shards in Sui, SAMM achieves higher throughput (X axis) with low average latency (Y axis). Error bars show additional experiments, × marking failure due to overload. We observe similar results in Solana (see the full version [22]). Improving the blockchains' parallelism would allow for even better performance.

The Mysten Labs Team reproduced our results and intends to make SAMM part of their benchmark suite.[2,3] The RISE Chain benchmark demonstrates a 10x throughput improvement for multiple independent Uniswap instances. The team considers our design as necessary for scaling decentralized applications [67].[3]

Finally, we conduct a simulation to justify our theoretical assumptions (§7). We simulate traders that are free to split their trades using demand from real traces without synthetic arbitrage. Indeed, traders exhibit the behavior predicted by our theoretical analysis. Furthermore, simulation results quantify the economic impact of SAMM's throughput gain. Splitting liquidity across shards worsens per-trade slippage, but each shard now processes more trades. As a result, a lower per-trade fee still yields substantially higher aggregate revenue for liquidity providers, and the lower fee offsets the added slippage, keeping traders' costs comparable to a standard CPMM (and lower under some parameters). The same fee-slippage trade-off also lets SAMM absorb large trades (may split across shards), with smaller price impact than a single-pool CPMM. Hence, SAMM is attractive to both liquidity providers and traders when competing with other standard AMMs. Beyond the simulation, we also prove that, although split liquidity increases slippage, it neither worsens sandwich attacks nor harms liquidity providers when token prices fluctuate. Moreover, price fluctuations add only a negligible number of arbitrage transactions in the worst case.

In summary, our contributions are: (1) Identification of game-theoretic security challenges in sharded contracts, (2) quantification of the AMM performance bottleneck, (3) SAMM: a sharded AMM contract, (4) generalization of the AMM trading fee function and a novel function for SAMM, (5) game-theoretic analysis identifying a Subgame-Perfect Nash Equilibrium (SPNE), (6) evaluation demonstrating an increase in throughput of 5x in Sui and of 16x in Solana (up to each blockchain's limit), (7) simulation with real trade data confirming the effects of SAMM's incentive design, and (8) validation that sharding does not incentivize sandwich attacks, and that it neither increases losses nor incurs substantial transaction overhead under price fluctuations.

These results provide an initial demonstration that high-demand smart contracts can be scaled securely by sharding a single logical service into multiple identical instances (§8). They also hint at an upcoming challenge in smart-contract platform design: minimizing the serial elements of transaction processing. But SAMM can already be employed to scale AMM performance.

2 Related Work

Addressing blockchain throughput limitations has been a central research focus since its inception. The first generation of blockchains, starting with Bitcoin [63], suffered from throughput limitations due to their consensus protocols. A body of work overcame this with a variety of protocols [33, 91, 88, 73, 28, 10, 1, 41, 51, 68, 89, 46, 82]. With consensus constraints out of the way, the serial execution of transactions became the bottleneck.

[2] From personal communication.
[3] The authors are not affiliated with Mysten Labs, Solana, or RISE and have no financial interest in their tokens.

Subsequent approaches identify read/write set conflicts and parallelize non-conflicting smart contract transaction execution [72, 30, 54, 65, 37]. Numerous practical blockchain platforms adopt such parallel execution engines: some require applications to explicitly shard independent components to expose parallelism [14, 88], while others automatically parallelize non-conflicting transactions [49]. On the former type of platforms, existing smart contracts (stateful programs) often rely on naive sharding, which partitions an independent component into multiple instances [61, 45, 17, 44], to benefit from parallel execution. However, non-parallelizable operations within a smart contract remain a bottleneck. To the best of our knowledge, no prior work has addressed improving the throughput of non-parallelizable smart contracts via sharding and studied the associated security challenges.

Uniswap v1 and v2 [92, 5] established the *Constant Product Market Maker* (*CPMM*) as a widely adopted, order-book-free design for token exchange [76, 64]. Uniswap v3 [6] and Curve [26] further improved AMMs' ability to handle larger trades. Uniswap v4 [9] and UniswapX [3] introduced practical updates to how pools are organized and trades are executed. Our sharding design targets parallelism and incentive alignment and can be composed with them.

Research on AMMs has studied generalized market-making mechanisms [7], *routers* which maximize trading utility by using multiple AMMs [8, 32, 93], advanced arbitrage techniques [94, 47, 11, 81, 83], improving liquidity providers' returns [39], measuring loss due to arbitrage [59, 36, 58], ensuring transaction privacy [24], eliminating Miner Extractable Value (MEV) for fair trades [20, 85, 18, 42], and examining the synergy between blockchain-based AMMs and prediction market mechanisms [70]. They do not consider AMM throughput and its security implications.

AMM throughput is limited by a single thread in current blockchains. Some AMM contracts [48, 66, 57, 4, 79] are deployed on so-called layer-2 solutions [2, 40, 12, 15] to defer some workload off-chain, but this merely creates a separate environment for AMMs, with scaling issues persisting (e.g., 138*tps* in ammBoost [57]) within this realm while introducing additional security assumptions.[1] Some existing AMMs use multiple instances of a single token pair with different parameters for flexibility [6], but most of the demand concentrates on a single instance [50]. To the best of our knowledge, previous work did not address the single-thread bottleneck of smart contracts in general and specifically for AMMs.

3 Model

We abstract away the blockchain details for our analysis and model the system as a set of participants interacting directly with AMMs, exchanging tokens. The system progresses in discrete steps and there exists an external market used by arbitrageurs to arbitrage in our system. The model uses a generic AMM, which we will later instantiate based on previous work and with SAMM.

Participants. There are two types of participants, *liquidity providers* and *traders*. Each liquidity provider holds some *token A* and *token B*. They aim to increase their holdings. Traders are either AB or BA, based on their goals. Each BA trader occasionally wishes to get a certain amount of *token A*, and vice versa for AB traders. They have sufficiently many tokens of the opposite type to complete their trade, but they aim to minimize the cost of obtaining the desired tokens. Both liquidity providers and traders can send and receive tokens to and from the smart contracts.

Automated Market Makers. The system also includes Automated Market Makers, stateful programs that facilitate *depositing* and *trading* of tokens. Each AMM maintains some deposited amounts R^A of *token A* and R^B of *token B*. We call these tokens *liquidity*.

Within an AMM, there are three primary operations: *liquidity addition*, *liquidity removal*, and *trade*. Specifically, a liquidity provider deposits I^A *token A* and I^B *token B* to the contract (liquidity addition); withdraws O^A *token A* and O^B *token B* from the contract (liquidity removal); or a trader sends I^A *token A* (resp., I^B *token B*) to the contract and gets O^B *token B* (resp., O^A *token A*) from the contract (trade).

In a trade operation, the required input amount for a trader to receive a specific output amount of another token depends on both the output amount and the AMM's current state. We define the gross amount of a BA trade as the required input amount of *token B* to get O^A *token A* in the AMM. We denote it by $G(R^A, R^B, O^A)$ (resp., $G(R^B, R^A, O^B)$ for AB traders). Within the gross amount, traders pay a so-called *trading fee* which contributes to the liquidity providers' revenue.

Liquidity providers supply liquidity by depositing their tokens in the contract. Once contributing to the AMM, a provider receives tokens from trading fees, hence earning revenue. Liquidity providers can later withdraw their tokens from the AMM.

Following prior work [59, 39], we assume that the trading fee is directly paid to the liquidity providers. Although some practical AMMs (e.g., Uniswap v2 [5]) reinvest the trading fees into themselves, allowing liquidity providers to withdraw more tokens than they deposited as utilities, the trading fee's impact is negligible relative to the deposited amount: The average ratio of the output amount to the deposited amount is very small (less than 0.036% as we find in the full version [22]), and automated market makers (AMMs) typically charge a low trading fee relative to the gross amount (e.g., 0.3% in Uniswap), so the trading fee's impact is negligible relative to the amount of deposited tokens.

System State and Progress. There are n independent AMM shards $shard_1$, $shard_2, \cdots, shard_n$ in the system. The system progresses in discrete steps $k = 0, 1, 2, \cdots$ and is orchestrated by a *scheduler*. In each step, the scheduler randomly selects a participant and this participant executes transactions. It chooses a liquidity provider with probability $P_{lp} \geq 0$ and a trader with probability $P_t \geq 0$, where $P_{lp} + P_t = 1$. The scheduler assigns the liquidity provider l^A *token A* and l^B *token B*, where (l^A, l^B) follows a random distribution D_{lp}.

The trader is either a BA trader aiming to obtain *token A* or an AB trader aiming to obtain *token B*. The probability of drawing an AB trader (resp., BA trader) is $P_t^{AB} \geq 0$ (resp., $P_t^{BA} \geq 0$), with $P_t^{AB} + P_t^{BA} = P_t$. To avoid repetition, we only show the case of BA traders; the expressions for AB traders are symmetric. The scheduler assigns the BA trader an amount of b^{BA} *token A* to obtain, following a random distribution D^{BA}.

External Market and Arbitrageurs. Following prior work [39, 59], we assume there is an *external market* providing the price of *token A* and *token B*, p^A and p^B, respectively. These prices do not change due to trades; they serve as objective prices for *token A* and *token B*.

If the AMM sells tokens at a price lower than the external market, a principal can buy tokens from the AMM and sell them in the external market to make profits, and vice versa; this is called an *arbitrage*. Previous work [59, 39, 18, 20] assumes active and rational arbitrageurs who can use the external market and arbitrage to maximize their utility. We follow the assumption [59] that there are immediate arbitrages without trading fees when the token price in the AMM is different from the external market.

Threat Model and Our Goal. As is typical in blockchain security research [90, 43, 60, 19, 87, 78, 56, 27], rational participants are potential attackers that might deviate from the intended protocol to maximize their own utility. In our context, attackers might, for example, overload the system or create temporary imbalances that reduce parallelism. Our goal is to design a set of AMM shards such that all rational participants behave in a way that improves the overall throughput of the system.

4 SAMM: Sharded AMM

We present SAMM, an AMM comprising multiple shards. Each shard is an independent AMM (§4.1), and SAMM's goal is to have operations distributed among the different shards (§4.2). To enforce this desired behavior despite attacks and untrusted users, we present a novel trading fee function (§4.3) and find parameters to fulfill SAMM's goal (§4.4).

4.1 SAMM Structure

SAMM enables parallel processing by deploying multiple AMM shards, each with an independent liquidity pool. There is no data dependency among the shards and there are no global elements. The operation of each shard is based on the Constant Product Market Maker (See detailed overview in the full version [22]), which keeps the product of the two tokens constant.

When liquidity providers add tokens to a shard, they receive *share tokens*, which signify their portion of its liquidity. Assume the shard has R^A *token A* and R^B *token B* deposited, and R^S is the number of all share tokens distributed before the operation. The amount of share tokens acquired by the liquidity provider is $O^S = R^S \times \min\left\{\frac{I^A}{R^A}, \frac{I^B}{R^B}\right\}$. If a liquidity provider pays I^S share tokens to withdraw deposited tokens, she gets $O^A = \frac{I^S}{R^S} \times R^A$ *token A* and $O^B = \frac{I^S}{R^S} \times R^B$ *token B*.

When a trader wants to get O^A *token A* from the AMM, ignoring trading fees, she needs to pay the *net amount*

$$net(R^A, R^B, O^A) = \frac{R^B \times O^A}{R^A - O^A} \ . \tag{1}$$

The *net price* for a single *token A* is $\frac{R^B}{R^A - O^A}$ *token B*. This value increases as the output amount of *token A*, O^A, increases, which is the *Slippage* of the trade.

4.2 Desired Properties

Most operations in an AMM are trades (We observe 99.5% in historical blockchain records, see the full version [22]). Our main goal is therefore that traders distribute the workload evenly among the shards. The following properties will suffice.

c-Non-Splitting Property. Our first goal is that trades only use a single shard, i.e., traders do not split a trade into smaller ones on multiple shards. This avoids the overhead caused by trade splits. So, the cost of a single-trade transaction should be less than the combined cost of multiple, split-trade transactions. However, this principle faces a significant challenge as highlighted by Equation 1: When the output amount is not small enough in comparison to the shard's deposited amount, the resulting slippage could incentivize traders to split their transactions to mitigate this slippage. Consequently, we refine our requirement to ensure that

this principle is adhered to only when the output amount is relatively small compared to the deposited amount, specifically when the ratio is below a predefined constant, c. Indeed, the prominent pairs in Ethereum's Uniswap v2 data support this approach, with 99% of trades having a ratio of output amount to deposited amount below 0.0052 (See the full version [22] for more details). We call this the *c-non-splitting* property.

▶ **Property 4.1** (c-non-splitting). *Let $m \geq 2$. Given a set of m output amounts $\{O_1^A, O_2^A, \ldots, O_m^A\}$ such that all amounts are positive, i.e., $\forall 1 \leq j \leq m : O_j^A > 0$, denote by $\tilde{O}^A$ the sum of the amounts in the set, $\tilde{O}^A = \sum_{j=1}^{m} O_j^A$. For the constant $0 < c < 1$ and the deposited amount of tokens R^A and R^B, if $\frac{\tilde{O}^A}{R^A} \leq c$, then the cost of trading $\tilde{O}^A$ token A is less than the sum of the cost of trading O_j^A token A for $1 \leq j \leq m$, i.e., $G_{SAMM}(R^A, R^B, \tilde{O}^A) < \sum_{j=1}^{m} G_{SAMM}(R^A, R^B, O_j^A)$.*

c-Smaller-Better Property. Our second goal is to maintain balanced volumes in all shards. This is crucial because the volume affects the slippage. When there are differences in shard sizes, with some being much smaller than others, the slippage in trading within these smaller shards is significantly greater than in larger ones. This discrepancy can result in transactions clustering in the larger shards rather than spread evenly, reducing parallelism. Moreover, an attacker could disrupt the system by adding or removing liquidity from certain shards, thereby harming overall system performance.

We address this by incentivizing liquidity providers to allocate their tokens to the shards with lower volumes. Intuitively, this ensures that smaller shards receive more frequent fees from traders when shard volumes are not balanced. This incentivizes liquidity providers to deposit tokens in these smaller shards. Therefore, the system would converge to the state where all shards have balanced volumes, and traders then randomly select shards for trading. As with the c-non-splitting property, large transactions suffer from high slippage, resulting in an advantage for larger shards. So here too, we refine this requirement to scenarios where the ratio of the traded amount to the deposited amount is below a threshold c. We call this the *c-smaller-better property.*

▶ **Property 4.2** (c-smaller-better). *Given an output amount $O^A > 0$, for any two shards with deposited token amounts (R_i^A, R_i^B) and (R_j^A, R_j^B), respectively, and $R_i^A < R_j^A$. For the constant $0 < c < 1$, if $O^A/R_j^A < O^A/R_i^A \leq c$ and $R_i^A/R_i^B = R_j^A/R_j^B$, then the cost of trading O^A token A in the smaller shard is less than that in the larger shard, i.e., $G_{SAMM}(R_i^A, R_i^B, O^A) < G_{SAMM}(R_j^A, R_j^B, O^A)$.*

***c* value.** If the above properties are satisfied for a particular c but a trade occurs with a larger ratio of output amount to the deposited amount, traders may split their transactions or tend to use larger shards to minimize their cost. Therefore, c should be as large as possible to ensure such occurrences are rare.

Note. CPMMs charge a constant ratio of the net amount as the trading fee. Therefore, their cost function does not satisfy either property due to slippage (see the full version [22]).

4.3 Trading Fee Design

To satisfy properties 4.1 and 4.2, we first generalize the trading fee function to provide flexibility in the incentive design. Then, we propose a specific trading fee function.

Generic Trading Fee Function. The gross amount of a trade comprises the trading fee and the net amount, with the latter being determined by the CPMM curve. To maintain the foundational characteristics of AMMs, we do not modify the CPMM curve. Instead, we generalize the trading fee function beyond simply charging a ratio of the net amount as in previous work (e.g., [7, 39, 8]).

Denote the trading fee function of the AMM by $tf(R^A, R^B, O^A)$, which takes the deposited amount of *token A*, R^A, and *token B*, R^B, in the shard and the output amount of *token A*, O^A, and outputs the amount of *token B* the trader needs to pay as the trading fee. Then, the gross amount of getting O^A *token A* is:

$$G(R^A, R^B, O^A) = tf(R^A, R^B, O^A) + net(R^A, R^B, O^A) \ . \tag{2}$$

Bounded-Ratio Trading Fee Function. In order to achieve the desired properties, we need flexibility for the trading fee function design. A monomial function is easy to parameterize and to differentiate, and is sufficient to achieve most of our goals. The function takes the variables available on a trade, R^A, R^B, O^A. It is parameterized by four values, $\beta_1, \beta_2, \beta_3, \beta_4$: $tf(R^A, R^B, O^A; \beta_1, \beta_2, \beta_3, \beta_4) \coloneqq \beta_1 (R^A)^{\beta_2} (R^B)^{\beta_3} (O^A)^{\beta_4}$.

The monomial is unbounded: Without limits, the trading fee can be too high to attract traders or too low to reward liquidity providers. Analogously to the fixed ratio in prior work, we thus add adjustable bounds $r_{\min}, r_{\max}$ and a base offset β_5 to fine-tune the fee's absolute value, yielding the bounded-ratio polynomial:

$$\begin{aligned} tf_{BRP}(R^A, R^B, O^A; \beta_1, \beta_2, \beta_3, \beta_4, \beta_5) \coloneqq \frac{R^B}{R^A} O^A \times \\ \max\{r_{\min}, \min\{r_{\max}, \beta_1 (R^A)^{\beta_2} (R^B)^{\beta_3} (O^A)^{\beta_4} + \beta_5\}\} \ . \end{aligned} \tag{3}$$

The ratio R^B/R^A represents the market price of *token A* relative to *token B* (see the full version [22]). The product $\frac{R^B}{R^A} O^A$ is thus the trader's net payment in terms of *token B* without slippage. Then $\max\{r_{\min}, \min\{r_{\max}, \beta_1 (R^A)^{\beta_2} (R^B)^{\beta_3} (O^A)^{\beta_4} + \beta_5\}\}$ acts as the constrained ratio of the trading fee to that net payment. We hereinafter omit the parameters and write $tf_{BRP}(R^A, R^B, O^A)$ for brevity.

4.4 Parameter Selection

Now we turn to the selection of parameters for the SAMM trading fee function. First, we identify the necessary conditions under which the bounded-ratio polynomial trading fee function aligns with the c-smaller-better property.

▶ **Proposition 4.3.** *Let $tf_{SAMM}(R^A, R^B, O^A) = tf_{BRP}(R^A, R^B, O^A)$, then the following conditions are necessary for c-smaller-better to hold for $G_{SAMM}(R^A, R^B, O^A)$: (1) $\beta_3 = 0$; (2) $\beta_2 + \beta_4 = 0$; (3) $\beta_1 < 0$; (4) $0 < \beta_4 \leq 1$; (5) $r_{\min} < \beta_5 \leq r_{\max}$; and (6) $\frac{\beta_5 - r_{\min}}{-\beta_1} \geq c^{\beta_4}$.*

We require the polynomial value to be between $r_{\min}$ and $r_{\max}$. Then, we need to make sure that the derivative of the gross amount with respect to the shard size is non-negative to ensure the c-smaller-better property. Required items come directly from these two restrictions. The proof is in the full version [22].

Based on Proposition 4.3, we require that $\beta_1 < 0$, $\beta_2 + \beta_4 = 0$, $\beta_3 = 0$, $0 < \beta_4 \leq 1$, and $r_{\min} < \beta_5 \leq r_{\max}$. Next, we identify additional conditions that are sufficient for both properties to hold.

▶ **Theorem 4.4.** *Let* $tf_{SAMM}(R^A, R^B, O^A) = tf_{BRP}(R^A, R^B, O^A)$*, if* $\beta_1 < 0, \beta_2 + \beta_4 = 0$*,* $\beta_3 = 0, 0 < \beta_4 \leq 1, r_{\min} < \beta_5 \leq r_{\max}$ *and* $\frac{\beta_5 - r_{\min}}{-\beta_1} \geq c^{\beta_4}$*, then the following items are sufficient for the c-non-splitting and c-smaller-better properties to hold for* $G_{SAMM}(R^A, R^B, O^A)$*: (1)* $\beta_1\beta_4(\beta_4+1)c^{\beta_4-1}(1-c)^3 \leq -2$*; and (2)* $-\beta_1\beta_4 \geq \frac{c^{1-\beta_4}}{(1-c)^2}$*.*

The c-smaller-better property is satisfied when the derivative of the gross amount is positive. It is sufficient to ensure the c-non-splitting property when the gross amount is concave with respect to the output amount, which is ensured by a negative second derivative over the output amount. The proof is in the full version [22].

By setting $\beta_2 = -1, \beta_3 = 0, \beta_4 = 1, \beta_5 = r_{\max}$, and choosing $\beta_1 < -1$, the fee function satisfies the above requirements, leaving just three parameters:

$$tf_{SAMM}(R^A, R^B, O^A) = \frac{R^B}{R^A} \times O^A \times \max\left\{r_{\min}, \beta_1 \times \frac{O^A}{R^A} + r_{\max}\right\} .$$

By setting $\beta_4 = 1$ and $\beta_5 = r_{\max}$ in the sufficient condition of the c-non-splitting and c-smaller-better properties (Theorem 4.4), the sufficient condition becomes:

▶ **Corollary 4.5.** *For any* $\beta_1 < -1$ *and* c *satisfying* $c \leq \min\left\{1 - (-\beta_1)^{-\frac{1}{3}}, \frac{r_{\max} - r_{\min}}{-\beta_1}\right\}$*, the SAMM cost function* $G_{SAMM}(R^A, R^B, O^A)$ *satisfies the c-non-splitting and c-smaller-better properties.*

For instance, the parameters $\beta_1 = -1.05$, $r_{\max} = 0.012$, $r_{\min} = 0.001$, and $c = 0.0104$ meet the specified criteria. According to historical records (see the full version [22]), over 99% of Uniswap v2 transactions have a ratio below 0.0052, suggesting that if its liquidity is split into two shards, 99% of transactions would fall within our targeted range. Specifically, in the AMMs with the highest trading volumes, USDC-ETH and USDT-ETH, the ratio remains below 0.00128, which can manage eight shards. Increasing the number of shards could be achieved by adjusting the value of c by increasing $r_{\max}$ and β_1.

5 Analysis of Game-Theoretic Security

The model gives rise to a game (§5.1) played by traders and liquidity providers. Our goal (§5.2) is to ensure that at equilibrium, traders randomly select shards to trade in without splitting, thereby enhancing throughput. SAMM achieves this (§5.3) and converges to a state where all shards have balanced volumes, overcoming attacks that unbalance the shards.

5.1 Game Model

We derive from the model a sequential game with discrete steps $k = 0, 1, \cdots$. Denote the game, parameterized by the number n of shards and their trading fee function *tf*, by $\Gamma_n(tf)$.

System State. In $\Gamma_n(tf)$, the state of each shard $shard_i$ in step k consists of the amount of deposited *token A*, the amount of deposited *token B*, and the amount of share tokens, $R_i^A(k), R_i^B(k), R_i^S(k)$, respectively. Recall that share tokens are not deposited in the shard but are held by liquidity providers, so R_i^S is the total amount of $shard_i$'s share tokens held by liquidity providers. Denote by $\mathbf{R}(k)$ the state of all AMM contracts in step k, $\left(\left(R_1^A(k), R_1^B(k), R_1^S(k)\right), \cdots, \left(R_n^A(k), R_n^B(k), R_n^S(k)\right)\right)$.

Liquidity Provider Actions. The liquidity provider decides the amount of tokens she deposits in each shard. Denote the amount of *token A* and *token B* she deposits in $shard_i$ by $l_i^A, l_i^B \geq 0$, respectively. Recall that the scheduler assigns the liquidity provider l^A *token A* and l^B *token B*. The total amount of tokens deposited should not exceed the amount she holds:

$$\forall 1 \leq i \leq n, l_i^A, l_i^B \geq 0, \sum_{i=1}^{n} l_i^A \leq l^A, \sum_{i=1}^{n} l_i^B \leq l^B \ .$$

The action of a liquidity provider is thus the vector $a_{lp} = \left(\left(l_1^A, l_1^B\right), \cdots, \left(l_n^A, l_n^B\right)\right)$. When the liquidity provider takes this action with the system state $\mathbf{R}(k)$, the liquidity provider receives O_i^S share tokens from $shard_i$, where $O_i^S = R_i^S(k) \times \min\left\{\frac{l_i^A}{R_i^A(k)}, \frac{l_i^B}{R_i^B(k)}\right\}$.

There is no arbitrage opportunity only if $R_i^A/R_i^B = p^B/p^A$ (see the full version [22]), so arbitrageurs enforce this equality. When $l_i^A/l_i^B = p^B/p^A$, increasing just one of l_i^A or l_i^B would not increase the share token the liquidity provider receives, which means more payment without more revenue. Therefore, we only consider actions where $l_i^A/l_i^B = p^B/p^A$ and require $l^A/l^B = p^B/p^A$. The action space of a liquidity provider is denoted by $\mathcal{A}_{lp}(l^A, l^B)$:

$$\mathcal{A}_{lp}(l^A, l^B) = \left\{ a_{lp} \left| \begin{array}{l} \forall 1 \leq i \leq n, l_i^A = \dfrac{p^B}{p^A} l_i^B \geq 0, \\ \displaystyle\sum_{i=1}^{n} l_i^A \leq l^A, \sum_{i=1}^{n} l_i^B \leq l^B \end{array} \right. \right\} .$$

We denote the updated state of shards from the previous state $\mathbf{R}$ and the action of a liquidity provider a_{lp} by $\mathbf{R} + a_{lp}$. Then for $\mathbf{R}' = \left(\left(R_1^{A'}, R_1^{B'}, R_1^{S'}\right), \cdots \left(R_n^{A'}, R_n^{B'}, R_n^{S'}\right)\right) = \mathbf{R} + a_{lp}$, we have

$$R_i^{A'} = R_i^A + l_i^A, R_i^{B'} = R_i^B + l_i^B, R_i^{S'} = R_i^S + O_i^S = \left(1 + \frac{l_i^A}{R_i^A}\right) R_i^S.$$

After the liquidity addition operation, there is no arbitrage opportunity for the arbitrageurs since $\frac{R_i^A + l_i^A}{R_i^B + l_i^B} = \frac{R_i^A}{R_i^B} = \frac{p^B}{p^A}$ (see the full version [22]). Then the update of state in step $k+1$ is $\mathbf{R}(k+1) = \mathbf{R}(k) + a_{lp}$.

Trader Actions. The action of a *BA* trader determines the amount of *token A* she acquires from each shard. Denote by $b_i^{BA} \geq 0$ the amount of *token A* she acquires in $shard_i$. Recall that the scheduler assigns the *BA* trader b^{BA} *token A* to acquire in total. The action of a *BA* trader is thus the vector $a^{BA} = \left(b_1^{BA}, \cdots, b_n^{BA}\right)$. The action space of a *BA* trader is denoted by $\mathcal{A}^{BA}(b^{BA})$, which is the set of all feasible actions:

$$\mathcal{A}^{BA}(b^{BA}) = \left\{ a^{BA} \left| \forall 1 \leq i \leq n, b_i^{BA} \geq 0, \sum_{i=1}^{n} b_i^{BA} = b^{BA} \right. \right\} \ . \tag{4}$$

After the trade operation and the arbitrage, R_i^A and R_i^B remain unchanged (see the full version [22]). Consequently, the state of the shards remains unchanged in the subsequent step: $\forall 1 \leq i \leq n, R_i^A(k+1) = R_i^A(k), R_i^B(k+1) = R_i^B(k)$.

Utility and Strategies. For traders and liquidity providers, we first discuss their revenue and then define their strategies and utility, respectively. Players determine the value of tokens according to the external market, namely p^A and p^B.

Consider a BA trader whose goal is to acquire b^{BA} units of *token A*. This trader needs to pay the gross amount and may derive some fixed reward from getting these tokens. We consider her revenue only as the inverse of the gross amount in terms of *token B* times the value of each *token B*:

$$U^{BA}(\mathbf{R}, a^{BA}) = -p^B \times \sum_i G(R_i^A, R_i^B, b_i^{BA}) = -p^B \times \sum_i G(R_i^A, \frac{p^A}{p^B} R_i^A, b_i^{BA}) \ . \tag{5}$$

For the trader aiming to get b^{BA} *token A*, the strategy of the trader $\pi^{BA}(\mathbf{R}, b^{BA}, a^{BA})$ takes $\mathbf{R}$, b^{BA} and an action a^{BA} as input, then outputs the probability of taking action a^{BA}. The total probability of all feasible actions should be 1: $\sum_{a^{BA} \in \mathcal{A}^{BA}(b^{BA})} \pi^{BA}(\mathbf{R}, b^{BA}, a^{BA}) = 1$.

The utility of the trader over the strategy is a function of the system state $\mathbf{R}$, the assigned requirement b^{BA}, and the strategy of the trader π^{BA}. It is the expected revenue under the distribution of actions:

$$U^{BA}(\mathbf{R}, b^{BA}, \pi^{BA}) = \sum_{a^{BA} \in \mathcal{A}^{BA}(b^{BA})} \left(\pi^{BA}(\mathbf{R}, b^{BA}, a^{BA}) \times U^{BA}(\mathbf{R}, a^{BA}) \right) \ . \tag{6}$$

The revenue of a liquidity provider comes from the trading fees paid by traders. Ignoring the effect of other liquidity providers, all future steps are identical due to arbitrageurs, so the long-term mean revenue is proportional to the next-step mean revenue. Hence, we consider the *myopic* setting (as in, e.g., [69, 34]), where the liquidity provider regards the revenue in the next step as her utility.

Denote the revenue of a liquidity provider with her action $a_{lp} = \left(\left(l_1^A, l_1^B\right), \cdots, \left(l_n^A, l_n^B\right) \right)$, the action of the BA trader in the next step $a^{BA} = \left(b_1^{BA}, \cdots, b_n^{BA}\right)$ and the system state $\mathbf{R} = \left(\left(R_1^A, R_1^B, R_1^S\right), \cdots \left(R_n^A, R_n^B, R_n^S\right) \right)$, by the function $U_{lp}(\mathbf{R}, a_{lp}, a^{BA})$. In the next step, $shard_i$ receives a trading fee of $tf(R_i^A + l_i^A, R_i^B + l_i^B, b_i^{BA})$. The liquidity provider receives a fraction of that fee proportional to her fraction of share tokens out of all shares in the shard. Therefore, the revenue function is

$$\begin{aligned} U_{lp}(\mathbf{R},\, a_{lp}, a^{BA}) =& p^B \times \sum_{i=1}^{n} \left\{ tf(R_i^A + l_i^A, R_i^B + l_i^B, b_i^{BA}) \times \frac{\frac{l_i^A}{R_i^A} R_i^S}{\left(1 + \frac{l_i^A}{R_i^A}\right) R_i^S} \right\} \\ =& p^B \times \sum_{i=1}^{n} \left\{ tf(R_i^A + l_i^A, R_i^B + l_i^B, b_i^{BA}) \times \frac{l_i^A}{l_i^A + R_i^A} \right\} \ . \end{aligned} \tag{7}$$

For the liquidity provider with l^A *token A* and l^B *token B*, the strategy of the liquidity provider $\pi_{lp}(\mathbf{R}, l^A, l^B)$ takes $\mathbf{R}$, l^A, l^B and an action a_{lp} as input, and outputs the probability of taking action a_{lp}. The total probability of all feasible actions should be 1,

$$\sum_{a_{lp} \in \mathcal{A}_{lp}(l^A, l^B)} \pi_{lp}(\mathbf{R}, l^A, l^B, a_{lp}) = 1 \ . \tag{8}$$

The utility of the liquidity provider over strategies is a function of the system state $\mathbf{R}$, the amount of tokens she is assigned l^A, l^B, and the strategy of the liquidity provider and traders, $\pi_{lp}, \pi^{BA}, \pi^{AB}$; denote it by $U_{lp}(\mathbf{R}, l^A, l^B, \pi_{lp}, \pi^{BA}, \pi^{AB})$. Before calculating this, we show the revenue given the action of the liquidity provider and the strategies of traders, denoted by $U_{lp}(\mathbf{R}, l^A, l^B, a_{lp}, \pi^{BA}, \pi^{AB})$. It takes the system state $\mathbf{R}$, the action of the liquidity provider a_{lp}, the strategies of traders π^{BA} and π^{AB} as input, then outputs the expected utility over the strategies and distributions of traders. The strategy of traders is affected by the state after the liquidity provider's action, namely $\mathbf{R} + a_{lp}$. Denote by $E_{b^{BA} \sim D^{BA}}[f(\cdot)]$ the expected value of $f(\cdot)$ with b^{BA} sampled from D^{BA}. The revenue is:

$$U_{lp}(\mathbf{R}, l^A, l^B, a_{lp}, \pi^{BA}, \pi^{AB}) = P_t^{BA} \times E_{b^{BA}\sim D^{BA}}\left[\sum_{a^{BA}\in\mathcal{A}^{BA}(b^{BA})}\binom{\pi^{BA}(\mathbf{R}+a_{lp}, b^{BA}, a^{BA})\times}{U_{lp}(\mathbf{R}, a_{lp}, a^{BA})}\right] + P_t^{AB} \times E_{b^{AB}\sim D^{AB}}\left[\sum_{a^{AB}\in\mathcal{A}_{AB}(b^{AB})}\binom{\pi^{AB}(\mathbf{R}+a_{lp}, b^{AB}, a^{AB})\times}{U_{lp}(\mathbf{R}, a_{lp}, a^{AB})}\right].$$

To simplify the presentation, we assume the liquidity provider is always followed by a BA trader. The expressions for an AB trader are symmetric. Then, the above equation can be simplified as

$$U_{lp}(\mathbf{R}, l^A, l^B, a_{lp}, \pi^{BA}) = E_{b^{BA}\sim D^{BA}}\left[\sum_{a^{BA}\in\mathcal{A}^{BA}(b^{BA})}\binom{\pi^{BA}(\mathbf{R}+a_{lp}, b^{BA}, a^{BA})\times}{U_{lp}(\mathbf{R}, a_{lp}, a^{BA})}\right]. \tag{9}$$

Then, the utility function of the liquidity provider over strategies is the expected utility under the distribution of actions:

$$U_{lp}(\mathbf{R}, l^A, l^B, \pi_{lp}, \pi^{BA}, \pi^{AB}) = \sum_{a_{lp}\in\mathcal{A}_{lp}(l^A, l^B)}\binom{\pi_{lp}(\mathbf{R}, l^A, l^B, a_{lp})}{\times U_{lp}(\mathbf{R}, l^A, l^B, a_{lp}, \pi^{BA}, \pi^{AB})}. \tag{10}$$

Solution Concept. In a Subgame-Perfect Nash Equilibrium (SPNE), players cannot gain higher utility by changing strategies at any step [71], knowing that subsequent players will take their best responses.

When a trader takes an action in a given step, her utility is influenced solely by her immediate strategy and the current state of AMMs, as outlined in Equation 5. Crucially, future actions do not affect this calculation, allowing the trader to directly optimize her utility, thereby establishing dominant strategies.

In the case of a liquidity provider being chosen in a step, the situation is different. Given their myopic viewpoint, liquidity providers only need to account for the strategy of the trader in the ensuing step. Their actions in subsequent steps do not affect their own utility. Thus, the sequential game is reduced to a two-stage Stackelberg game and SPNE to a Stackelberg Equilibrium [80].

To formalize this, we denote the strategies of the liquidity provider, the BA trader, and the AB trader in the SPNE by τ_{lp}, τ^{BA} and τ^{AB}, respectively. The BA trader would always get the optimal utility in equilibrium, namely $\forall \mathbf{R}, b^{BA}$, we have $U^{BA}(\tau^{BA}, \mathbf{R}, b^{BA}) = \max_{\pi^{BA}} U^{BA}(\pi^{BA}, \mathbf{R}, b^{BA})$.

Note that τ^{BA} is a best response for the BA trader. The strategy of liquidity provider in equilibrium is just the optimal strategy when traders adopt their best response, namely for all $\mathbf{R}$, l^A and l^B, we have $U_{lp}(\tau_{lp}, \mathbf{R}, \tau^{BA}, \tau^{AB}, l^A, l^B) = \max_{\pi_{lp}} U_{lp}(\pi_{lp}, \mathbf{R}, \tau^{BA}, \tau^{AB}, l^A, l^B)$.

Game Assumptions. We briefly discuss three non-trivial model assumptions.

First, we did not explicitly consider liquidity removal in the game. If a liquidity provider removes all her liquidity, it just changes the state of the AMM pools and our analysis still holds. If the liquidity provider strategically removes liquidity, it is equivalent to removing all liquidity first and then adding liquidity again. Therefore, we do not lose generality by ignoring liquidity removals.

Second, in a blockchain, each transaction consumes resources measured in a metric called *gas*. The transaction pays a fee according to its gas consumption. In practice, gas fees are volatile, often very low in high-performance blockchains (e.g., in Sui under \$0.005

per transaction on average [75, 77]). The model conservatively neglects gas fees to avoid arbitrary assumptions on gas costs. Gas fees disincentivize trade splitting, which helps with parallelization. In particular, with a set gas fee, Properties 4.1 and 4.2 continue to hold. Therefore, it can only strengthen the equilibrium analysis.

Third, perfect arbitrageurs are a common assumption in the literature [59, 39, 18, 20]. We nonetheless demonstrate later (§7) that our results hold with real workloads, without this assumption. In addition, we do not model arbitrage that trades across SAMM shards or between SAMM and other decentralized exchanges (DEXs). We call this type of arbitrage *DEX-to-DEX*. Any DEX-to-DEX arbitrage can be replaced by two independent trades, each between a DEX and the external market, yielding the same payoff. Therefore, DEX-to-DEX arbitrage is weakly dominated by arbitrages with the external market. Empirically, DEX-to-DEX arbitrage accounts for less than 10% of observed arbitrage activity [25].

5.2 Desired Property

Our goal is to improve the throughput by allowing parallelism. Specifically, we would like traders to evenly distribute their transactions among all AMM shards without splitting them. That is, a dominant strategy for the *BA* trader should be to randomly select an AMM shard to acquire all her needed *token A*. Denote the action of getting all b^{BA} *token A* in $shard_i$ by

$$a_i^{BA}(b^{BA}) = \left(0, \cdots, b_i^{BA} = b^{BA}, \cdots, 0\right) \ . \tag{11}$$

Denote the set of these actions by $\mathcal{A}_{\mathbf{1}}(b^{BA}) = \left\{a_i^{BA}(b^{BA}) \,|1 \leq i \leq n\right\} \subset \mathcal{A}_t^{BA}(b^{BA})$. The strategy that uniformly at random selects an AMM contract to acquire all her needed *token A* is the *perfect parallelism strategy*:

▶ **Definition 5.1.** *The perfect parallelism strategy of the BA trader is*

$$\hat{\tau}^{BA}(\boldsymbol{R}, b^{BA}, a^{BA}) = \begin{cases} \frac{1}{n}, & \text{if } a^{BA} \in \mathcal{A}_{\mathbf{1}}(b^{BA}) \\ 0, & \text{otherwise} \end{cases} .$$

Our goal is thus to have the perfect parallelism strategy be a dominant strategy:

▶ **Property 5.2.** *The perfect parallelism strategy of the BA trader is a dominant strategy:*

$$\forall \pi^{BA} : U^{BA}(\pi^{BA}, \boldsymbol{R}, b^{BA}) \leq U^{BA}(\hat{\tau}^{BA}, \boldsymbol{R}, b^{BA}) \ .$$

Using multiple CPMMs does not satisfy the perfect parallelism property and would be counterproductive: Each trader would split her transactions across all AMM contracts. Thus, although the total number of trades increases due to parallelism, the satisfied trade demand is no higher than a single AMM contract and possibly lower since the total throughput might only increase sublinearly in the number of AMM contracts. See more details in the full version [22].

5.3 SAMM Equilibrium

We analyze the SPNE of the above game. We only provide a roadmap here, from trader strategy to liquidity provider strategy; the complete analysis is in the full version [22].

5.3.1 Trader Strategy

Consider the case that the system state is $\mathbf{R} = \left(\left(R_1^A, R_1^B, R_1^S\right), \cdots \left(R_n^A, R_n^B, R_n^S\right)\right)$. As discussed in Section 4.4, the SAMM gross amount satisfies the c-non-splitting property and c-smaller-better property for a certain $0 < c < 1$. We assume that the required amount of *token A*, b^{BA}, is at most a fraction c of the amount of deposited *token A* in all shards, i.e., $\forall 1 \leq i \leq n, b^{BA} \leq cR_i^A$.

The c-non-splitting property and c-smaller-better property give a trader the incentive to randomly select one of the smallest shards to trade all her required tokens. Recall that $a_i^{\mathrm{BA}}(b^{BA})$ is the action of acquiring all b^{BA} *token A* in *shard*$_i$ (Equation 11). We define the set of actions that trade in one of the smallest shards:

▶ **Definition 5.3.** *The* Smallest Shard Action Set *is the set of actions that acquire all b^{BA} token A in one of the smallest shards under state $\boldsymbol{R}$:* $\mathcal{A}_{\mathbf{1},\min}(b^{BA}, \boldsymbol{R}) = \left\{a_i^{BA}(b^{BA}) | \forall j, R_i^A \leq R_j^A\right\}$.

The cardinality of $\mathcal{A}_{\mathbf{1},\min}(b^{BA}, \mathbf{R})$ is the number of smallest shards in $\mathbf{R}$. We denote this by $n_{\min}(\mathbf{R}) = \left|\mathcal{A}_{\mathbf{1},\min}(b^{BA}, \mathbf{R})\right|$.

We prove that randomly selecting a shard to trade without splitting is a dominant strategy for the trader, and it is strictly better than splitting or trading in a larger shard:

▶ **Theorem 5.4.** *In $\Gamma_n(tf_{SAMM})$, considering the following dominant strategy of the BA trader which randomly selects one of the smallest shards to acquire all required tokens:*

$$\tau^{BA}(\boldsymbol{R}, b^{BA}, a^{BA}) = \begin{cases} \frac{1}{n_{\min}(\boldsymbol{R})}, & \text{if } a^{BA} \in \mathcal{A}_{\mathbf{1},\min}(b^{BA}, \boldsymbol{R}) \\ 0, & \text{otherwise} \end{cases},$$

then for all strategies π^{BA} that have a positive probability of actions not trading in one of the smallest shards, i.e., $\exists a^{BA} = \left(b_1^{BA}, \cdots, b_i^{BA}, \cdots, b_n^{BA}\right) \notin \mathcal{A}_{\mathbf{1},\min}(b^{BA}, \boldsymbol{R})$, $\pi^{BA}(\boldsymbol{R}, b^{BA}, a^{BA}) > 0$, the utility of the BA trader is strictly lower than with strategy τ^{BA}:

$$U^{BA}(\tau^{BA}, \boldsymbol{R}, b^{BA}) > U^{BA}(\pi^{BA}, \boldsymbol{R}, b^{BA}) .$$

Proof Sketch. Due to the c-non-splitting property, trading in a single shard is strictly better than trading in multiple shards. Then the revenue of trading in one of the smallest shards is strictly better than that in any other shard due to the c-smaller-better property. The complete proof is in the full version [22]. ◀

If shards are balanced, which means all shards have the same amount of deposited tokens, we have $n_{\min}(\mathbf{R}) = n$. Then it is a dominant strategy for the trader to randomly select one of the n shards to trade, as we intended:

▶ **Corollary 5.5.** *In $\Gamma_n(tf_{SAMM})$, the system state is $\boldsymbol{R} = \left(\left(R_1^A, R_1^B, R_1^S\right), \cdots \left(R_n^A, R_n^B, R_n^S\right)\right)$. If $\forall i, j, R_i^A = R_j^A$ and $R_i^B = R_j^B$, then the perfect parallelism strategy*

$$\hat{\tau}^{BA}(\boldsymbol{R}, b^{BA}, a^{BA}) = \begin{cases} \frac{1}{n}, & \text{if } a^{BA} \in \mathcal{A}_{\mathbf{1}}(b^{BA}) \\ 0, & \text{otherwise} \end{cases}$$

is a dominant strategy for the BA trader.

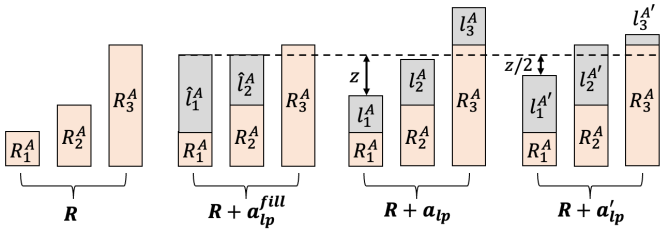


Figure 2 An example construction of a'_{lp}.

5.3.2 Liquidity Provider Strategy

Having shown that traders behave as intended if the shards are balanced, we consider the strategies of liquidity providers in equilibrium. They should maintain the shard balance. Moreover, their incentives should rebalance the system state even in the face of attacks that break this balance.

We want a liquidity provider to fill up smaller shards to keep shards balanced. We call such an action the *fillup action*, where if the liquidity provider adds tokens to a shard, then the shard is the smallest shard after this action. We denote the fillup action by $a_{lp}^{fill}(\mathbf{R}, l^A, l^B)$. Note that there is only one such action.

▶ **Definition 5.6.** *The fillup action of a liquidity provider* $a_{lp}^{fill}(\boldsymbol{R}, l^A, l^B) = \left(\left(\hat{l}_1^A, \hat{l}_1^B\right), \cdots, \left(\hat{l}_n^A, \hat{l}_n^B\right)\right)$ *is the action where if the liquidity provider adds tokens to a shard, then the shard is one of the smallest shards after this action:*

$$\forall 1 \le i \le n: \quad (1)\ \ \hat{l}_i^A \ge 0, \sum_{i=1}^{n} \hat{l}_i^A = l^A \ , (2)\ \ \forall \hat{l}_i^A > 0, \forall j : \hat{l}_i^A + R_i^A \le \hat{l}_j^A + R_j^A \ .$$

The *fillup strategy* only uses the fillup action:

▶ **Definition 5.7.** *The fillup strategy of a liquidity provider* $\tau_{lp}^{fill}(\boldsymbol{R}, l^A, l^B)$ *is the strategy that only takes the fillup action:*

$$\tau_{lp}^{fill}(\boldsymbol{R}, l^A, l^B, a_{lp}) = \begin{cases} 1, & \text{if } a_{lp} = a_{lp}^{fill}(\boldsymbol{R}, l^A, l^B) \\ 0, & \text{otherwise} \end{cases} .$$

We first prove that, when all shards have identical sizes, the fillup action is to add tokens to all shards evenly, which is the best response of the liquidity provider:

▶ **Theorem 5.8.** *Denote by* $\hat{a}_{lp} = \left(\left(\frac{1}{n}l^A, \frac{1}{n}l^B\right), \cdots\right)$ *the action of evenly depositing tokens in all shards. In* $\Gamma_n(tf_{SAMM})$*, if for all* i *and* j *that the liquidity amounts are the same,* $R_i^A = R_j^A$ *and* $R_i^B = R_j^B$*, the liquidity provider strategy which only takes action* $\hat{a}_{lp}$*,*

$$\tau_{lp}(\boldsymbol{R}, l^A, l^B, a_{lp}) = \begin{cases} 1, & \text{if } a_{lp} = \hat{a}_{lp} \\ 0, & \text{otherwise} \end{cases}$$

and any best response of the trader constitutes an SPNE.

Proof Sketch. Traders prefer trading in smaller shards to reduce their costs. At the same time, fees are higher in larger shards. So the liquidity provider should increase her share in the smallest shards. This dual objective is optimally achieved by uniformly distributing tokens across all shards. The complete proof is in the full version [22]. ◀

The above theorem indicates that once the system reaches a balanced state, this state is stable. We now show that even if the system reaches an unbalanced state, maybe due to an attacker, it converges to a balanced state since the liquidity provider uses the fillup strategy.

We show that the fillup strategy is the only best response in all SPNE:

▶ **Theorem 5.9.** *In $\Gamma_n(tf_{SAMM})$, in all SPNE, the liquidity provider's best response is the fillup strategy:*

$$\tau_{lp}^{fill}(\boldsymbol{R}, l^A, l^B, a_{lp}) = \begin{cases} 1, & \text{if } a_{lp} = a_{lp}^{fill}(\boldsymbol{R}, l^A, l^B) \\ 0, & \text{otherwise} \end{cases}.$$

Proof Sketch. Given any action a_{lp} that is not the fillup action, we can construct a new action a'_{lp} that is strictly better than a_{lp}. By ensuring the smallest shard in $\mathbf{R} + a'_{lp}$ is larger than that in $\mathbf{R} + a_{lp}$, we increase trading fees garnered from each transaction. Moreover, this smallest shard is also the smallest before the action, maximizing the liquidity provider's share. Consequently, the liquidity provider earns higher revenue under a'_{lp} than under a_{lp}. Thus, any strategy incorporating an action other than the fillup action is not optimal. Figure 2 illustrates an example of the a'_{lp} construction. The complete proof is in the full version [22]. ◀

Theorem 5.9 does not prove the existence of an SPNE. We prove one exists by constructing an SPNE for a general (perhaps unbalanced) starting state (see the full version [22]): Liquidity providers follow the fillup strategy, and traders use the smallest shard with the largest share of the previous liquidity provider.

In summary, we showed that the system achieves a stable state with perfect parallelism. After events that create heterogeneous shard sizes (like imbalancing liquidity additions by an attacker or asymmetric liquidity removals by liquidity providers), liquidity providers rebalance shards once sufficient liquidity has been provided. Malicious attackers incur ongoing losses, and when their budget is exhausted, the system returns to a balanced state. These results indicate that SAMM is robust to such attacks.

6 Evaluation

To evaluate the performance, we use the state-of-the-art blockchains Sui (§6.1) and Solana. We find that the throughput of a single-contract AMM is limited (§6.2), and that it improves with the number of SAMM shards (§6.3). Our analysis shows that further improvement is possible by increasing the platform's parallelism (§6.4).

6.1 Experimental Setup

We conduct experiments on the Sui [14] blockchain. Smart contracts in Sui are independent *objects*, and Sui executes transactions on different objects in parallel. We implement SAMM[4] in the Move language [13]. We deploy a local testnet, which follows the default configuration, consisting of 4 validators maintaining consensus of the blockchain. The latency of transactions is always higher than 1 second due to Sui's consensus protocol. We issue transactions using Sui's Rust RPC interface.

[4] `https://github.com/MountainGold/SAMM-Sui-Evaluation`

As a baseline, we test the latency of simple token transfers. As expected, unencumbered by smart-contract coordination constraints, the latency is consistently smaller than 200 msec (Figure 1) in Sui at 2360*tps* (outside the figure range). This throughput is about twice the maximum rate observed in our AMM experiments on Sui. The latency remains under one second because token transfers in Sui do not require immediate consensus among validators.

We also conduct experiments on Solana [88], which has a different architecture and consensus protocol. Details of the Solana experimental setup are in the full version [22].

For all reported results, we use a machine with 2 TB of memory and 256 CPU cores. We run 100 trader processes for Sui (and 50 for Solana). Several experiments with more traders or with the testnet deployed and transactions sent from separate machines produce similar results. This shows that workload generation does not introduce bottlenecks.

Each trader sends transactions at random intervals following an exponential distribution with an expected frequency of λ. A trader can send another transaction before the previous one is confirmed. Note this experiment is only for performance evaluation, so traders follow the perfect parallelism strategy. We vary the overall frequency of transactions by setting different values of the individual λ values. In each test, we set a target throughput. We first warm up the system by sending transactions for 500 seconds and then measure actual throughputs and latencies for the following 100 seconds. If the latency is stable during the 100 measurement seconds, we report the mean value.

6.2 Single-Contract Bottleneck

To demonstrate the bottleneck of a single AMM, we first deploy a standard CPMM. We use the OmniSwap contract [64] in Sui and the token-swap contract from the Solana Program Library [53], which are generalizations of Uniswap v2.

For the Sui experiment, Figure 1 ($n = 1$) shows latencies of transaction processing (Y axis) in workloads with varying transaction frequencies (X axis) using a single OmniSwap contract. We test each frequency 5 times and calculate the truncated average, excluding the two extreme values. Error bars show all measured values. The average latency increases gradually with the transaction frequency up to 214*tps* before crossing the 3-second line. With higher frequencies, transactions frequently fail.

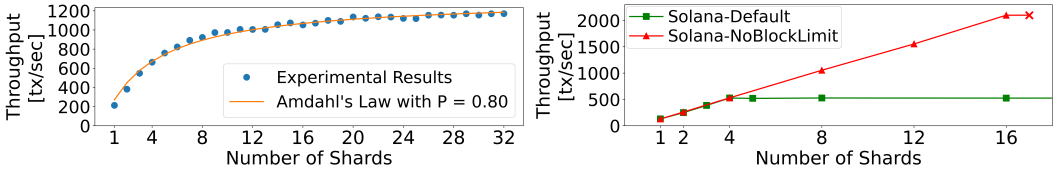
Our Solana experiments produce similar results (see the full version [22]). The throughput bottleneck of a single CPMM contract in both Solana-Default and Solana-NoBlockLimit is 129*tps*.

6.3 SAMM Evaluation

We implement SAMM by modifying the fee mechanism of CPMM contracts and deploying a varying number of shards (contracts). Each trader randomly selects a shard to trade.

Figure 1 reports the average latency under varying demand (*tps*) across different shard counts in our Sui experiments. The latency in the case of one SAMM contract is indistinguishable from a single CPMM contract. In some instances, more than half of the transactions failed, which is marked with $\times$ in the graph. As the number of SAMM contracts increases, the system can process higher demand.

To quantify SAMM's performance enhancements in Sui, we evaluate the throughput with a varying number of shards. We set a latency cap of 3 seconds. As depicted in Figure 1, the latency rises rapidly once it surpasses 2 seconds. Therefore, choosing other latency caps beyond 2 seconds does not significantly affect the results. The throughput with n shards is thus the highest frequency that produced a latency lower than 3 seconds. Figure 3a shows the



(a) SAMM's experimental throughput in Sui. **(b)** SAMM's experimental throughput in Solana.

Figure 3 Maximal throughput as a function of the number of SAMM shards in Sui and Solana.

throughput increases almost linearly in the beginning and then converges to a bound. With 32 shards, the maximal throughput exceeds 1185*tps*, more than five times the throughput of a single OmniSwap contract.

On Solana, we test two configurations: *Solana-Default* uses the standard mainnet gas limits, while *Solana-NoBlockLimit* removes the per-block gas cap to expose the architecture's limit (see the full version [22]). Figure 3b shows that the maximal throughput scales almost linearly up to 4 shards under Solana-Default and up to 16 shards under Solana-NoBlockLimit, with 17 shards failing to match the *tps* of 16 shards, marked with $\times$.

6.4 Parallelization in the Underlying Platform

When there are n concurrently operating AMM shards, each with a maximum throughput of $T_{\max}$, the total maximum throughput, $T_{\text{total}}(n)$, is influenced by the fraction P of the transaction that can be parallelized, according to Amdahl's Law. This law states that the speedup ratio, $S(n)$, is the total throughput relative to a single AMM's throughput, according to the expression $S(n) = \frac{1}{(1-P)+\frac{P}{n}}$. For fully sequential systems like Ethereum, $P = 0$, resulting in no throughput gain. For stability, Solana sets a gas limit that keeps throughput well below its serial bottleneck. Its throughput is thus linear in n, up to the artificial limit (Figure 3b).

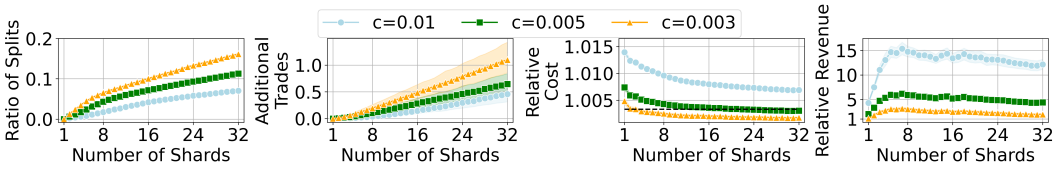
In Sui, we fit the experimental data with Amdahl's law and conclude the parallelizable part of a transaction is $P = 0.80$ ($R^2 = 0.99$). Since the serial components of transactions are invariant, throughput improvements are inherently limited. According to the fitted curve, the throughput is bounded by 1330*tps*. We find that if transactions in Sui have a larger parallelizable portion, the throughput improvement due to SAMM is even better (see the full version [22]).

7 Incentive-Compatibility Validation and Secondary Effects

We simulate SAMM on real trading data (§7.1): Incentive compatibility holds even without assuming perfect arbitrageurs (§7.2). SAMM also raises liquidity-provider revenue at trader costs comparable to a standard CPMM, without incentivizing sandwich attacks or amplifying losses and overhead under price fluctuations (§7.3).

7.1 Simulation Setup

We simulate SAMM with workloads from public CPMM trading data, namely the Uniswap v2 Ethereum AMM for the pair of tokens USDC and ETH (hereinafter the *workload*). We assume both Uniswap and SAMM operate at the maximal throughput measured in our performance evaluation. In each test, we simulate over 1 second, following a 1-second warm-up period. We repeat each simulation 100 times and calculate averages. Note that a 1-second



(a) Ratio of transaction splits. **(b)** Additional trades due to splits. **(c)** Traders' costs relative to the external market. **(d)** Relative lp's revenue compared to Uniswap.

Figure 4 Trade splits, trader costs, and lp's (liquidity providers') revenue in SAMM.

simulation corresponds to several hours' trading in the real world. We conservatively choose the throughput of SAMM according to the performance evaluation of Sui where throughput improves sublinearly with the number of shards.

We simulate SAMM using 1 to 32 shards and Uniswap. For SAMM with n shards, we evenly distribute the liquidity of the workload among the n shards to establish the initial state of SAMM. We test three different values of c, namely 0.003, 0.005, and 0.01. We still do not consider the gas fee in the simulation, as it would only make the results better since it discourages trade splitting (§5.1).

In each test, we replay the trades in the workload. Each trader selects the shard offering the lowest price and may split the trade into several smaller trades if this reduces her costs.

Our simulation differs from the game-theoretic analysis in two key aspects. First, the simulation does not assume perfect arbitrageurs, as there are no synthetic arbitrage transactions. Second, the trade amount of some transactions exceeds the c fraction of a single shard's liquidity. This is particularly relevant in our simulation, where each shard holds only $1/n$ of the total liquidity, occasionally resulting in trade splits. See more details in the full version [22].

7.2 Incentive-Compatibility Validation

We analyze the proportion of trades that are split into multiple transactions within SAMM. With $c = 0.01$, trade splits are infrequent, occurring in less than 8% of trades, even with 32 shards (Figure 4a). Furthermore, the total number of trades (the trader might split a trade into more than 2 parts) results in less than a 45% increase in overall trade count (Figure 4b), a relatively minor increment given the five-fold throughput improvement. When the number of shards is larger or c is smaller, the proportion of trade splits is higher. We also confirm that the distribution of trades in SAMM is balanced (see the full version [22]). This confirms that our theoretical analysis holds without assumptions of perfect arbitrageurs or trade size limits.

7.3 Secondary Effects of Sharding

Sharding results in smaller shards, since liquidity is split. This results in higher slippage for trades. We analyze how this affects participant revenues and AMM vulnerabilities.

Traders' Costs and Liquidity Provider Revenue. We now analyze how sharding affects the revenues of traders and liquidity providers. In a trade with a set output amount and fixed deposited amounts of tokens in a shard, adjusting the fee parameters is zero-sum for the trader and liquidity provider: with higher fees, the liquidity provider gains more revenue,

while the trader incurs higher costs (Equation 2). However, sharding allows more trades to be processed, so liquidity providers can gain more revenue while each trader's cost still remains comparable to that in a standard CPMM.

Our simulation also validates this economic impact. Figure 4c illustrates traders' costs: while increasing with the fee parameter c, costs decrease as the number of shards increases, remaining comparable to Uniswap or even lower. This shows how higher throughput can offset increased slippage. Figure 4d reveals a significant increase in total liquidity provider revenue; it climbs with the number of shards (reaching up to 15 times that of Uniswap) due to more trades, before eventually declining as individual fees per trade diminish in very small shards. These trends demonstrate a key advantage of SAMM: It increases the exchange volume, benefiting all participants, and is not a zero-sum game that simply reallocates costs. Hence, SAMM attracts participants when competing with other AMMs. More mathematical analysis and empirical details are provided in the full version [22].

We also observe that SAMM outperforms Uniswap v2 on *volume capacity*, which measures the ability of handling large trades with small price impact (see the full version [22]).

The analysis of revenue, volume capacity and incentive-compatibility (§7.2) implies a trade-off between participant incentives and system performance. A larger c results in fewer trade splits and higher revenue for liquidity providers, but the cost for traders increases. Fixing c and increasing the number of shards results in lower trader costs but more trade splits, which can negatively impact system performance. Additionally, liquidity provider revenue peaks at a certain number of shards. System designers can tune c and the number of shards to balance participant incentives and system performance.

Sandwich Attacks and Price-Fluctuation Losses and Overhead. A critical consideration is whether the increased slippage inherent in smaller shards exacerbates known AMM vulnerabilities, specifically sandwich attacks and losses due to price fluctuations. Our analysis demonstrates that, counter-intuitively, the profitability of sandwich attacks decreases as liquidity decreases, rendering smaller shards less sensitive (see the full version [22]). When the price of tokens in the external market fluctuates, arbitrageurs can arbitrage between the external market and the AMM, causing losses for the liquidity providers, called loss-versus-rebalancing (LVR) [59, 36, 58]. For CPMM-based AMMs like SAMM, with perfect arbitrageurs, the loss per unit of liquidity is independent of the size of the AMM [59, Example 3], meaning the loss ratio is unaffected by sharding. Thus, SAMM enhances throughput without amplifying these economic risks.

Note that the fluctuation of token prices in the external market may lead to arbitrage opportunities, and with more shards there are more such opportunities. However, in the worst case, each external price move triggers one arbitrage per shard per block. An n-shard system (say, $n = 32$) delivering hundreds of *tps* would suffer at most n arbitrage transactions per block. This overhead is minor.

8 Conclusion

We present SAMM, a scalable AMM that employs smart contract sharding. SAMM addresses incentive-driven security challenges through a novel trading fee function, which we validate via game-theoretic analysis and simulations on real trade traces. We implement and deploy SAMM on local Sui and Solana testnets, demonstrating throughput improvements of over 5x and 16x, respectively, up to the limits of the underlying platforms. SAMM is directly applicable and has already attracted industry interest. Nonetheless, how to choose and adapt

the fee parameters and the threshold c as trade-size distributions shift, and how sensitive the system is to their miscalibration, remain to be studied and verified in real-world deployments. Our results provide the first demonstration that high-demand smart contracts can be securely scaled via sharding. They also highlight the importance of minimizing serial bottlenecks in smart-contract platforms to enable further scaling (see the full version [22]).

References

1 Ittai Abraham, Danny Dolev, Ittay Eyal, and Joseph Y Halpern. Colordag: An incentive-compatible blockchain. In *37th International Symposium on Distributed Computing (DISC 2023)*, pages 1:1–1:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.DISC.2023.1.

2 Austin Adams. Layer 2 be or layer not 2 be: Scaling on uniswap v3. https://arxiv.org/abs/2403.09494, 2024.

3 Hayden Adams. Introducing the uniswapx protocol. https://blog.uniswap.org/uniswapx-protocol, 2023.

4 Hayden Adams, Mark Toda, Alex Karys, Xin Wan, Daniel Gretzke, Eric Zhong, Zach Wong, Daniel Marzec, Robert Miller, Hasu, Karl Floersch, and Dan Robinson. Unichain. https://developers.uniswap.org/whitepaper_unichain.pdf, 2024.

5 Hayden Adams, Noah Zinsmeister, and Dan Robinson. Uniswap v2 core. https://app.uniswap.org/whitepaper.pdf, 2020.

6 Hayden Adams, Noah Zinsmeister, Moody Salem, River Keefer, and Dan Robinson. Uniswap v3 core. https://app.uniswap.org/whitepaper-v3.pdf, 2021.

7 Guillermo Angeris and Tarun Chitra. Improved price oracles: Constant function market makers. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 80–91, 2020. doi:10.1145/3419614.3423251.

8 Guillermo Angeris, Alex Evans, Tarun Chitra, and Stephen Boyd. Optimal routing for constant function market makers. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, pages 115–128, 2022. doi:10.1145/3490486.3538336.

9 Mohammad Ali Asef and Seyed Mojtaba Hosseini Bamakan. From x* y= k to uniswap hooks: A comparative review of decentralized exchanges (dex). https://arxiv.org/abs/2410.10162, 2024.

10 Vivek Bagaria, Sreeram Kannan, David Tse, Giulia Fanti, and Pramod Viswanath. Prism: Deconstructing the blockchain to approach physical limits. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 585–602, 2019. doi:10.1145/3319535.3363213.

11 Massimo Bartoletti, James Hsin-yu Chiang, and Alberto Lluch Lafuente. Maximizing extractable value from automated market makers. In *International Conference on Financial Cryptography and Data Security*, pages 3–19. Springer, 2022. doi:10.1007/978-3-031-18283-9_1.

12 Mihailo Bjelic, Sandeep Nailwal, Amit Chaudhary, and Wenxuan Deng. Pol: One token for all polygon chains. https://polygon.technology/papers/pol-whitepaper. Accessed, April 2024.

13 Sam Blackshear, Evan Cheng, David L Dill, Victor Gao, Ben Maurer, Todd Nowacki, Alistair Pott, Shaz Qadeer, Dario Russi Rain, Stephane Sezer, et al. Move: A language with programmable resources. https://developers.diem.com/papers/diem-move-a-language-with-programmable-resources/2019-06-18.pdf.

14 Sam Blackshear, Andrey Chursin, George Danezis, Anastasios Kichidis, Lefteris Kokoris-Kogias, Xun Li, Mark Logan, Ashok Menon, Todd Nowacki, Alberto Sonnino, et al. Sui lutris: A blockchain combining broadcast and consensus. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 2606–2620, 2024. doi:10.1145/3658644.3670286.

15 Lee Bousfield, Rachel Bousfield, Chris Buckland, Ben Burgess, Joshua Colvin, Edward W. Felten, Steven Goldfeder, Daniel Goldman, Braden Huddleston, Harry Kalodner, Frederico Arnaud Lacs, Harry Ng, Aman Sanghi, Tristan Wilson, Valeria Yermakova, and Tsahi Zidenberg. Arbitrum nitro: A second-generation optimistic rollup. `https://github.com/OffchainLabs/nitro/blob/master/docs/Nitro-whitepaper.pdf`. Accessed, April 2024.

16 Lea Salome Brugger, Laura Kovács, Anja Petkovic Komel, Sophie Rain, and Michael Rawson. Checkmate: Automated game-theoretic security reasoning. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 1407–1421, 2023. `doi:10.1145/3576915.3623183`.

17 Bucket Protocol. Bucket Protocol Documentation. `https://docs.bucketprotocol.io/`. Accessed, December 28, 2025.

18 Andrea Canidio and Robin Fritsch. Batching trades on automated market makers. In *5th Conference on Advances in Financial Technologies (AFT 2023)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2023. `doi:10.4230/LIPIcs.AFT.2023.24`.

19 Miles Carlsten, Harry Kalodner, S Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 154–167, 2016. `doi:10.1145/2976749.2978408`.

20 TH Chan, Ke Wu, and Elaine Shi. Mechanism design for automated market makers. In *7th Conference on Advances in Financial Technologies (AFT 2025)*, pages 7:1–7:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. `doi:10.4230/LIPIcs.AFT.2025.7`.

21 Hongyin Chen, Yubin Ke, Xiaotie Deng, and Ittay Eyal. Prrr: Personal random rewards for blockchain reporting. In *2026 IEEE Symposium on Security and Privacy (SP)*, pages 3080–3098. IEEE, 2026. `doi:10.1109/SP63933.2026.00136`.

22 Hongyin Chen, Amit Vaisman, and Ittay Eyal. Samm: Sharded automated market maker. `https://arxiv.org/abs/2406.05568`, 2024.

23 Hongyin Chen, Xu Zheng, Jichen Li, and Ittay Eyal. Time is money: Incentivized causal transaction ordering. arXiv preprint, 2026. `doi:10.48550/arXiv.2607.11496`.

24 Tarun Chitra, Guillermo Angeris, and Alex Evans. Differential privacy in constant function market makers. In *International Conference on Financial Cryptography and Data Security*, pages 149–178. Springer, 2022. `doi:10.1007/978-3-031-18283-9_8`.

25 Christoph Rosenmayr and Mateusz Dominiak. Statistical arbitrage on AMMs and block building on Ethereum – Part 1. `https://greenfieldcapital.com/2024/09/10/statistical-arbitrage-on-amms-and-block-building-on-ethereum-part-1/`, 2024.

26 Curve.fi. Curve documentation release 1.0.0. `https://curve.readthedocs.io/_/downloads/en/latest/pdf/`, 2022.

27 Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 910–927. IEEE, 2020. `doi:10.1109/SP40000.2020.00040`.

28 George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. Narwhal and tusk: a dag-based mempool and efficient bft consensus. In *Proceedings of the Seventeenth European Conference on Computer Systems*, pages 34–50, 2022. `doi:10.1145/3492321.3519594`.

29 DefiLlama. Total value locked all chains. `https://defillama.com/chains`. Accessed, July 2024.

30 Thomas Dickerson, Paul Gazzillo, Maurice Herlihy, and Eric Koskinen. Adding concurrency to smart contracts. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 303–312, 2017. `doi:10.1145/3087801.3087835`.

31 Ellipsis Labs. phoenix-cli. `https://github.com/Ellipsis-Labs/phoenix-cli`. Accessed, December 28, 2025.

32 Daniel Engel and Maurice Herlihy. Composing networks of automated market makers. In *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, pages 15–28, 2021. doi:10.1145/3479722.3480987.

33 Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. Bitcoin-NG: A scalable blockchain protocol. In *13th USENIX symposium on networked systems design and implementation (NSDI 16)*, pages 45–59, 2016. URL: https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/eyal.

34 Matheus VX Ferreira, Daniel J Moroz, David C Parkes, and Mitchell Stern. Dynamic posted-price mechanisms for the blockchain transaction-fee market. In *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, pages 86–99, 2021. doi:10.1145/3479722.3480991.

35 Emilio Frangella and Lasse Herskind. Aave V3 Technical Paper. https://raw.githubusercontent.com/aave/aave-v3-core/master/techpaper/Aave_V3_Technical_Paper.pdf, 2022.

36 Robin Fritsch and Andrea Canidio. Measuring arbitrage losses and profitability of amm liquidity. In *Companion Proceedings of the ACM on Web Conference 2024*, pages 1761–1767, 2024. doi:10.1145/3589335.3651961.

37 Péter Garamvölgyi, Yuxi Liu, Dong Zhou, Fan Long, and Ming Wu. Utilizing parallelism in smart contracts on decentralized blockchains by taming application-inherent conflicts. In *Proceedings of the 44th International Conference on Software Engineering*, pages 2315–2326, 2022. doi:10.1145/3510003.3510086.

38 Bikramaditya Ghosh, Hayfa Kazouz, and Zaghum Umar. Do automated market makers in defi ecosystem exhibit time-varying connectedness during stressed events? *Journal of Risk and Financial Management*, 16(5):259, 2023. doi:10.3390/jrfm16050259.

39 Mohak Goyal, Geoffrey Ramseyer, Ashish Goel, and David Mazières. Finding the right curve: Optimal design of constant function market makers. In *Proceedings of the 24th ACM Conference on Economics and Computation*, pages 783–812, 2023. doi:10.1145/3580507.3597688.

40 Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. Sok: Layer-two blockchain protocols. In *International Conference on Financial Cryptography and Data Security*, pages 201–226. Springer, 2020. doi:10.1007/978-3-030-51280-4_12.

41 Guy Golan Gueta, Ittai Abraham, Shelly Grossman, Dahlia Malkhi, Benny Pinkas, Michael Reiter, Dragos-Adrian Seredinschi, Orr Tamir, and Alin Tomescu. Sbft: A scalable and decentralized trust infrastructure. In *2019 49th Annual IEEE/IFIP international conference on dependable systems and networks (DSN)*, pages 568–580. IEEE, 2019. doi:10.1109/DSN.2019.00063.

42 Lioba Heimbach and Roger Wattenhofer. Eliminating sandwich attacks with the help of game theory. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, pages 153–167, 2022. doi:10.1145/3488932.3517390.

43 Charlie Hou, Mingxun Zhou, Yan Ji, Phil Daian, Florian Tramèr, Giulia Fanti, and Ari Juels. Squirrl: Automating attack analysis on blockchain incentive mechanisms with deep reinforcement learning. In *28th Annual Network and Distributed System Security Symposium (NDSS)*, 2021. doi:10.14722/ndss.2021.24188.

44 Hubble Protocol. Hubble Protocol Documentation. https://docs.hubbleprotocol.io/. Accessed, December 28, 2025.

45 Kamino Finance. Kamino Lend Litepaper: Unified Liquidity Market. https://kamino.com/docs. Accessed, July 25, 2026.

46 Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. Omniledger: A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE symposium on security and privacy (SP)*, pages 583–598. IEEE, 2018. doi:10.1109/SP.2018.000-5.

47 Kshitij Kulkarni, Theo Diamandis, and Tarun Chitra. Routing mev in constant function market makers. In *International Conference on Web and Internet Economics*, pages 456–473. Springer, 2023. doi:10.1007/978-3-031-48974-7_26.

48 L2 Lab. Zkswap: A layer-2 token swap protocol based on zk-rollup. https://github.com/l2labs/zkswap-whitepaper/blob/master/zkswap_en.pdf, 2020.

49 Aptos Labs. The aptos blockchain: Safe, scalable, and upgradeable web3 infrastructure. https://aptosfoundation.org/whitepaper/aptos-whitepaper_en.pdf, 2022.

50 Alfred Lehar, Christine Parlour, and Marius Zoican. Fragmentation and optimal liquidity supply on decentralized exchanges. https://arxiv.org/abs/2307.13772, 2023.

51 Chenxin Li, Peilun Li, Dong Zhou, Zhe Yang, Ming Wu, Guang Yang, Wei Xu, Fan Long, and Andrew Chi-Chih Yao. A decentralized blockchain with high throughput and fast confirmation. In *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*, pages 515–528, 2020. URL: https://www.usenix.org/conference/atc20/presentation/li-chenxing.

52 Zihao Li, Jianfeng Li, Zheyuan He, Xiapu Luo, Ting Wang, Xiaoze Ni, Wenwu Yang, Xi Chen, and Ting Chen. Demystifying defi mev activities in flashbots bundle. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 165–179, 2023. doi:10.1145/3576915.3616590.

53 Solana Program Library. Token swap program. https://spl.solana.com/. Accessed, July 2024.

54 Jian Liu, Peilun Li, Raymond Cheng, N Asokan, and Dawn Song. Parallel and asynchronous smart contract execution. *IEEE Transactions on Parallel and Distributed Systems*, 33(5):1097–1108, 2021. doi:10.1109/TPDS.2021.3095234.

55 MK Manoylov. Arbitrum surpasses $150 billion in total transaction volume on uniswap. https://www.theblock.co/post/292655/, 2024.

56 Patrick McCorry, Alexander Hicks, and Sarah Meiklejohn. Smart contracts for bribing miners. In *Financial Cryptography and Data Security: FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers 22*, pages 3–18. Springer, 2019. doi:10.1007/978-3-662-58820-8_1.

57 Nicolas Michel, Mohamed E Najd, and Ghada Almashaqbeh. ammboost: State growth control for amms. In *2025 55th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 143–156. IEEE, 2025. doi:10.1109/DSN64029.2025.00027.

58 Jason Milionis, Ciamac C Moallemi, and Tim Roughgarden. Automated market making and arbitrage profits in the presence of fees. In *International Conference on Financial Cryptography and Data Security*, pages 159–171. Springer, 2024. doi:10.1007/978-3-031-78676-1_9.

59 Jason Milionis, Ciamac C Moallemi, Tim Roughgarden, and Anthony Lee Zhang. Automated market making and loss-versus-rebalancing. https://arxiv.org/abs/2208.06046, 2022.

60 Michael Mirkin, Yan Ji, Jonathan Pang, Ariah Klages-Mundt, Ittay Eyal, and Ari Juels. Bdos: Blockchain denial-of-service. In *Proceedings of the 2020 ACM SIGSAC conference on Computer and Communications Security*, pages 601–619, 2020. doi:10.1145/3372297.3417247.

61 Morpho. Morpho documentation: Market. https://docs.morpho.org/learn/concepts/market/. Accessed, December 24, 2025.

62 Marwa Mouallem, Lorenz Breidenbach, Ittay Eyal, and Ari Juels. Resilient alerting protocols for blockchains. To appear in the 2026 ACM SIGSAC Conference on Computer and Communications Security (CCS), 2026. doi:10.48550/arXiv.2602.10892.

63 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, 2008.

64 OmniBTC. Omniswap. https://github.com/OmniBTC/OmniSwap. Accessed, April 2024.

65 George Pîrlea, Amrit Kumar, and Ilya Sergey. Practical smart contract sharding with ownership and commutativity analysis. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, pages 1327–1341, 2021. doi:10.1145/3453483.3454112.

66 QuickSwap. Quickswap documentation. https://docs.quickswap.exchange/. Accessed: April 2024.

67 risechain. pevm. https://github.com/risechain/pevm. Accessed, December 23, 2025.

68 Team Rocket, Maofan Yin, Kevin Sekniqi, Robbert van Renesse, and Emin Gün Sirer. Scalable and probabilistic leaderless bft consensus through metastability. https://arxiv.org/abs/1906.08936, 2019.

69 Tim Roughgarden. Transaction fee mechanism design. *ACM SIGecom Exchanges*, 19(1):52–55, 2021. doi:10.1145/3476436.3476445.

70 Jan Christoph Schlegel, Mateusz Kwasnicki, and Akaki Mamageishvili. Axioms for constant function market makers. In *Proceedings of the 24th ACM Conference on Economics and Computation*, page 1079, 2023. doi:10.1145/3580507.3597720.

71 Reinhard Selten. Spieltheoretische behandlung eines oligopolmodells mit nachfrageträgheit: Teil i: Bestimmung des dynamischen preisgleichgewichts. *Zeitschrift für die gesamte Staatswissenschaft/Journal of Institutional and Theoretical Economics*, 121(2):301–324, 1965. URL: https://www.jstor.org/stable/40748884.

72 Ilya Sergey and Aquinas Hobor. A concurrent perspective on smart contracts. In *Financial Cryptography and Data Security: FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers 21*, pages 478–493. Springer, 2017. doi:10.1007/978-3-319-70278-0_30.

73 Alexander Spiegelman, Neil Giridharan, Alberto Sonnino, and Lefteris Kokoris-Kogias. Bullshark: Dag bft protocols made practical. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2705–2718, 2022. doi:10.1145/3548606.3559361.

74 Sui Foundation. DeepBookV3. https://docs.sui.io/onchain-finance/deepbook/deepbookv3/deepbook. Accessed, July 25, 2026.

75 suiscan. Average gas fee. https://suiscan.xyz/mainnet/analytics/avg-gas-fee. Accessed, September 2024.

76 Sushi. Be a crypto chef with sushi. https://www.securities.io/sushiswap-whitepaper/. Accessed: July 2026.

77 TheBlock. Sui price (sui): Price index and live chart. https://www.theblock.co/price/248445/sui-sui-eur. Accessed, September 2024.

78 Itay Tsabary, Matan Yechieli, Alex Manuskin, and Ittay Eyal. Mad-htlc: because htlc is crazy-cheap to attack. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1230–1248. IEEE, 2021. doi:10.1109/SP40001.2021.00080.

79 Uniswap Labs. Uniswap on Optimism. https://blog.uniswap.org/uniswap-optimism-alpha, 2021.

80 Heinrich Von Stackelberg. *Market structure and equilibrium*. Springer Science & Business Media, 2010. doi:10.1007/978-3-642-12586-7.

81 Jianhuan Wang, Jichen Li, Zecheng Li, Xiaotie Deng, and Bin Xiao. n-mvtl attack: Optimal transaction reordering attack on defi. In *European Symposium on Research in Computer Security*, pages 367–386. Springer, 2023. doi:10.1007/978-3-031-51479-1_19.

82 Jiaping Wang and Hao Wang. Monoxide: Scale out blockchains with asynchronous consensus zones. In *16th USENIX symposium on networked systems design and implementation (NSDI 19)*, pages 95–112, 2019. URL: https://www.usenix.org/conference/nsdi19/presentation/wang-jiaping.

83 Ye Wang, Yan Chen, Haotian Wu, Liyi Zhou, Shuiguang Deng, and Roger Wattenhofer. Cyclic arbitrage in decentralized exchanges. In *Companion Proceedings of the Web Conference 2022*, pages 12–19, 2022. doi:10.1145/3487553.3524201.

84 Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. https://ethereum.github.io/yellowpaper/paper.pdf, 2015.

85 Matheus Venturyne Xavier Ferreira and David C Parkes. Credible decentralized exchange design via verifiable sequencing rules. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 723–736, 2023. doi:10.1145/3564246.3585233.

86 Tianjing Xu, Yongqi Zhong, Yiming Zhang, Ruofan Xiong, Jingjing Zhang, Guangtao Xue, and Shengyun Liu. Vegeta: Enabling parallel smart contract execution in leaderless blockchains. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*, pages 795–811, 2025. URL: https://www.usenix.org/conference/nsdi25/presentation/xu-tianjing.

87 Aviv Yaish, Gilad Stern, and Aviv Zohar. Uncle maker:(time) stamping out the competition in ethereum. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 135–149, 2023. doi:10.1145/3576915.3616674.

88 Anatoly Yakovenko. Solana: A new architecture for a high performance blockchain v0. 8.13. https://solana.com/solana-whitepaper.pdf. Accessed, April 2024.

89 Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 931–948, 2018. doi:10.1145/3243734.3243853.

90 Ren Zhang and Bart Preneel. Lay down the common metrics: Evaluating proof-of-work consensus protocols' security. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 175–192. IEEE, 2019. doi:10.1109/SP.2019.00086.

91 Ren Zhang, Dingwei Zhang, Quake Wang, Shichen Wu, Jan Xie, and Bart Preneel. NC-max: Breaking the security-performance tradeoff in nakamoto consensus. In *Proceedings 2022 Network and Distributed System Security Symposium*, pages 1–18. NDSS, 2022. doi:10.14722/ndss.2022.24370.

92 Yi Zhang, Xiaohong Chen, and Daejun Park. Formal specification of constant product (xy= k) market maker model and implementation. https://github.com/runtimeverification/publications/blob/main/reports/smart-contracts/Uniswap-V1.pdf, 2018.

93 Liyi Zhou, Kaihua Qin, and Arthur Gervais. A2mm: Mitigating frontrunning, transaction reordering and consensus instability in decentralized exchanges. https://arxiv.org/abs/2106.07371, 2021.

94 Liyi Zhou, Kaihua Qin, Christof Ferreira Torres, Duc V Le, and Arthur Gervais. High-frequency trading on decentralized on-chain exchanges. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 428–445. IEEE, 2021. doi:10.1109/SP40001.2021.00027.

Defensive Rebalancing in Networks of Automated Market Makers

Sam Devorsetz
Computer Science Department, Brown University, Providence, RI, USA

Maurice Herlihy
Computer Science Department, Brown University, Providence, RI, USA

Abstract

This paper introduces and analyzes *defensive rebalancing*, a novel mechanism for preventing arbitrageurs from extracting surplus from networks of constant-function market makers (CFMMs).

A *rebalancing* transfers assets directly from one CFMM's pool to another's, bypassing the CFMMs' standard trading protocols. In any *arbitrage-prone* configuration, we prove there exists a rebalancing to an *arbitrage-free* configuration that strictly increases the liquidity of at least one participating CFMM without decreasing the liquidity of any other participating CFMM. Moreover, we prove that a configuration is arbitrage-free if and only if it is *Pareto efficient* under rebalancing.

We prove that for any log-concave trading function, including the ubiquitous constant-product market maker, finding an optimal arbitrage-free rebalancing that maximizes the aggregate liquidity of the participating CFMMs while preserving each participant's liquidity can be formulated as a convex optimization problem with a unique, computationally tractable solution.

We extend this framework to *mixed rebalancing*, where a subset of participating CFMMs use a combination of direct transfers and standard trades to transition to an arbitrage-free configuration while capturing arbitrage opportunities from non-participating CFMMs and external price-setting oracles such as centralized exchanges.

Our results provide a rigorous foundation for future CFMM protocols that coordinate defensive rebalancing to internalize arbitrage surplus within networks of CFMMs.

2012 ACM Subject Classification Information systems → Distributed storage

Keywords and phrases Automated Market Makers, Arbitrage, Convex Optimization

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.12

Funding *Maurice Herlihy*: Supported by a grant from Nunchi corporation.

1 Introduction

A *constant-function automated market maker* [3] (CFMM) $\mathcal{A}$ is an automaton that trades one token for another. It owns two pools of tokens, let's call them dollar tokens (\$) and euro tokens (€). $\mathcal{A}$ holds x_1 dollars in its dollar pool, and x_2 euros in its euro pool. Token exchange rates are determined by a *trading function* $F(x_1, x_2)$, satisfying some common-sense technical conditions given below. Initially $F(x_1, x_2) = k$, for some $k > 0$, called $\mathcal{A}$'s *liquidity*. A trader who transfers $\delta_\$$ dollars to $\mathcal{A}$ receives $\delta_€$ euros in return, where $F(x_1 + \delta_\$, x_2 - \delta_€) = k$. That is, a *trade* changes $\mathcal{A}$'s pool sizes but leaves its liquidity unchanged.

Consider the following *circular trade* scenario. There are three CFMMs, each trading one token type for another. $\mathcal{A}$ trades dollars for euros, $\mathcal{B}$ trades euros for pounds (£), and $\mathcal{C}$ trades pounds for dollars. All three have trading function $F(x_1, x_2) := x_1 \cdot x_2 = 3$.

- $\mathcal{A}$ has €1 and \$3,
- $\mathcal{B}$ has £1 and €3, and
- $\mathcal{C}$ has \$1 and £3.


© Sam Devorsetz and Maurice Herlihy;
licensed under Creative Commons License CC-BY 4.0

Alice (an *arbitrageur*) sees an opportunity to make a quick risk-free profit. She sells \$1 to $\mathcal{C}$ for £$\frac{3}{2}$, which she sells to $\mathcal{B}$ for €$\frac{9}{5}$, which she sells to $\mathcal{A}$ for \$$\frac{27}{14}$, collecting a profit of \$$\frac{13}{14}$. Her profit comes at the expense of the *liquidity providers* (LPs), the investors who loaned their tokens to $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$.

Suppose instead that before Alice strikes, the LPs for $\mathcal{A}, \mathcal{B}$ and $\mathcal{C}$, recognizing their danger, agree to transfer €1 directly from $\mathcal{B}$'s euro pool to $\mathcal{A}$'s euro pool, £1 from $\mathcal{C}$'s pound pool to $\mathcal{B}$'s pound pool, and \$1 from $\mathcal{A}$'s dollar pool to $\mathcal{C}$'s dollar pool. These are direct pool-to-pool transfers, bypassing the CFMMs' usual trading protocols.

The state resulting from this *rebalancing* has three interesting properties. First, arbitrage across $\mathcal{A}, \mathcal{B}, \mathcal{C}$ has become impossible: no circular trade results in a profit. Second, rebalancing has improved the liquidity levels of *all three* CFMMs (each went from 3 to 4), without reducing the value of any CFMM's assets (in a sense formalized below). Third, no subsequent rebalancing can increase one CFMM's liquidity without reducing another's. The first condition improves social welfare by harmonizing exchange rates; the second improves individual welfare because higher liquidity implies lower price slippage, making that CFMM more attractive to traders. The third condition means individual welfare cannot be further improved.

This paper shows that the first and third conditions are equivalent, and the second follows as a consequence of the others. As explained in Section 2, a *configuration* is the global state of a network of CFMMs. A configuration is *arbitrage-free* if no agent can make a profit simply by executing a sequence of trades. As noted, a *rebalancing* is a set of direct transfers from one CFMM pool directly to another. A configuration c is *Pareto efficient* (under rebalancing) if any rebalancing starting from c that increases one CFMM's liquidity must reduce another's. In Section 3, we show that *the arbitrage-free configurations are precisely the Pareto-efficient configurations*. In short, the Pareto-efficient configurations are exactly those that eliminate arbitrage opportunities while preserving or improving every participating CFMM's liquidity.

This equivalence has several implications.

- From any arbitrage-prone configuration, there exist rebalancings that simultaneously eliminate arbitrage exposure and strictly increase some CFMMs' liquidities without decreasing the others'.
- One could provide "rebalancing as a service" where a set of participating CFMMs periodically check whether their configuration is arbitrage prone. If so, they agree on a rebalancing that simultaneously protects them from arbitrage and renders their liquidity levels Pareto efficient.
- Section 4 shows that for any log-concave trading function, the problem of finding a rebalancing leading to an arbitrage-free/Pareto-efficient configuration can be formulated as a convex optimization problem, computationally tractable using widely available solvers. Moreover, the equivalence between arbitrage-free and Pareto-efficient configurations simplifies the optimization program's formulation.
- Section 5 shows that this convex optimization program can be readily extended to systems where some CFMMs refuse to allow direct pool-to-pool transfers, and some act like powerful centralized exchanges, setting global market prices for certain tokens.

Our analysis assumes that a proposed rebalancing is executed before the corresponding arbitrage opportunity has been exhausted by external traders. We focus on the existence and properties of such rebalancings rather than on the mechanisms required to schedule or prioritize them.

2 Model

A function $F : \mathbb{R}^k \to \mathbb{R}$ is *strictly convex* if for all $t \in (0,1)$ and distinct x_1, x_2,

$$F(tx_1 + (1-t)x_2) < tF(x_1) + (1-t)F(x_2).$$

A positive-valued function F is *log-concave* if for all $t \in (0,1)$ and distinct x_1, x_2,

$$\log F(tx_1 + (1-t)x_2) > t \log F(x_1) + (1-t) \log F(x_2).$$

We use "=" for equality and ":=" for definitions.

2.1 Automated Market Makers

Agents own *tokens*. A token is any valuable item that can be transferred electronically. An *automated market maker* (AMM) is an automaton that trades between two token types, T_1, T_2. An AMM manages a *pool* of each token. An AMM's *state* is a pair (x_1, x_2), meaning the AMM has x_1 units of T_1 in its first pool, and x_2 units of T_2 in its second pool.[1] A *configuration* c is an ordered collection of AMM states.

We consider *constant-function market makers* [3] (CFMMs) in which each trade preserves an invariant of the form $F(x_1, x_2) = k$, where $k > 0$, and $F : \mathbb{R}^2 \to \mathbb{R}$ is twice differentiable, strictly increasing in each argument, and log-concave. The value k is called the CFMM's *liquidity*.[2] Perhaps the most popular CFMM is the *constant-product market maker* (CPMM) of the form $F(x_1, x_2) := x_1 \cdot x_2 = k$. Uniswap [1], Balancer [14], Bullish [7], CoinEx [10], and Skate [19] are examples of commercial CPMMs.

A *liquidity provider* (LP) is an agent who invests in a CFMM by loaning tokens to its liquidity pools. LPs expect to profit from trading fees, and typically receive governance rights over the CFMM.

For an AMM holding $x_\$$ dollars and $x_€$ euros, a *trade* is a pair $(\delta_\$, \delta_€)$ such that

$$F(x_\$ + \delta_\$, x_€ - \delta_€) = F(x_\$, x_€).$$

Thus, if a trader sends $\delta_\$$ dollars to the AMM, it receives $\delta_€$ euros in return, where $\delta_€$ is chosen so that the liquidity remains unchanged. Formally, the AMM's possible states under trading correspond to a *level set* of F:

$$\{(x_1, x_2) \mid F(x_1, x_2) = k\},$$

where the liquidity k remains constant for the duration of the trade.

For a CFMM with trading function $F(x_1, x_2)$, the *spot price* of token T_1 in terms of T_2 in state (x_1, x_2) is:

$$-\frac{\partial F/\partial x_1}{\partial F/\partial x_2}(x_1, x_2).$$

Trades change spot prices, an effect known as *slippage*. For example, a CPMM $x_1 x_2 = 2$ in state $(1, 2)$ has a dollar spot price of 2 euros, but a trader who deposits 2 euros changes the state to $(\frac{1}{2}, 4)$, and receives only $\frac{1}{2}$ dollar. The trade increased the spot price as dollars become more scarce compared to euros.

[1] Our results generalize to AMMs that trade more than two types of tokens, but we focus on two-pool AMMs for simplicity.

[2] This term is sometimes called the CFMM's *invariant* or *liquidity constant*. In our model, however, it is neither invariant nor constant over the course of repeated rebalancings.

Consider two CFMMs $\mathcal{A}, \mathcal{B}$, both trading dollars against euros. If $\mathcal{A}$ and $\mathcal{B}$ have different spot prices, then they are vulnerable to *arbitrage*. Suppose euros are cheaper in $\mathcal{B}$ than in $\mathcal{A}$. An *arbitrageur* agent spends d dollars to buy euros from $\mathcal{B}$, then sells those euros to $\mathcal{A}$ where they are more valuable, receiving $d' > d$ dollars in return. The agent has made a risk-free profit at the expense of the CFMMs' LPs. Arbitrage can of course involve multiple AMMs.

▶ **Definition 1.** *A configuration is* arbitrage-prone *if an agent starting with a basket of tokens* $(b_1, \ldots, b_m)$ *can execute a sequence of trades and end up with a basket* $(b'_1, \ldots, b'_m)$ *where* $b'_i \geq b_i$*, and at least one inequality is strict. A configuration that is not arbitrage-prone is* arbitrage-free.

A *rebalancing* transfers tokens directly between liquidity pools belonging to different CFMMs. Unlike trades, rebalancing typically changes both AMMs' liquidities and their spot prices. A rebalancing is self-funding in the sense that it does not change the total amounts of each token held in participants' pools. A rebalancing *improves liquidity* if every AMM's post-rebalancing liquidity is at least its original liquidity, and at least one such inequality is strict.

Note that trades and rebalancings are distinct operations. A trade takes place between a party and a CFMM, and always leaves the AMM's liquidity unchanged. A rebalancing takes place among multiple CFMMs, and typically alters their liquidities. Trades are a standard part of classical AMMs, while rebalancings are (as far as we know) analyzed for the first time in this paper.

Formally, a rebalancing is defined as follows. Starting from a configuration

$$c := (x_{1,1}, x_{1,2}), \ldots, (x_{n,1}, x_{n,2}),$$

a rebalancing carries c to a new configuration

$$c' := (x'_{1,1}, x'_{1,2}), \ldots, (x'_{n,1}, x'_{n,2}).$$

Tokens can be transferred only between pools holding the same token type. We encode permitted transfers in the index set:

$$\mathcal{T} := \{(i,j,k,\ell) \mid i < k \text{ and the pools corresponding to } x_{i,j}, x_{k,\ell} \text{ have the same token type}\}.$$

A rebalancing is defined by a collection:

$$\{\delta_{i,j,k,\ell} \mid (i,j,k,\ell) \in \mathcal{T}\}.$$

Each $\delta_{i,j,k,\ell}$ represents a transfer of $|\delta_{i,j,k,\ell}|$ tokens. If $\delta_{i,j,k,\ell} > 0$, then $|\delta_{i,j,k,\ell}|$ tokens are transferred from the pool corresponding to $x_{i,j}$ to the pool corresponding to $x_{k,\ell}$; if $\delta_{i,j,k,\ell} < 0$, the transfer is in the opposite direction. More precisely, the rebalancing sends each pool $x_{i,j}$ to $x'_{i,j}$, where:

$$x'_{i,j} = x_{i,j} + \sum_{\substack{k,\ell: \\ (k,\ell,i,j)\in\mathcal{T}}} \delta_{k,\ell,i,j} - \sum_{\substack{k,\ell: \\ (i,j,k,\ell)\in\mathcal{T}}} \delta_{i,j,k,\ell}.$$

A rebalancing is *feasible* if all final pool balances are positive. All rebalancings considered here are assumed to be feasible.

▶ **Definition 2.** *A configuration* c *is* Pareto efficient *if no rebalancing starting from* c *can strictly increase the liquidity of any CFMM without strictly reducing the liquidity of another CFMM.*

Protecting a network of CFMMs from arbitrage is a public good that benefits the network as a whole, although those benefits may be distributed unevenly among the individual CFMMs. Because a given arbitrage-prone configuration typically admits a continuum of Pareto-efficient rebalancings, there is considerable freedom in selecting a target configuration. For example, in Section 4, we consider a rebalancing that selects the Pareto-efficient configuration maximizing the product of the squares of the CFMMs' resulting liquidities. Each participating CFMM may communicate additional constraints to the rebalancing agent (e.g., "I must retain at least 100 euros"). The agent, in turn, is free to exclude any CFMM whose individual constraints make rebalancing infeasible. Ultimately, however, participation is voluntary: every CFMM retains veto power and may decline to participate in any proposed rebalancing for any reason.

We assume each sequence of trades or rebalancings executes *atomically*, so that no other agent's actions are interleaved with it.

Our results characterize the allocations that are achievable under rebalancing. They do not address how a rebalancing transaction obtains priority over competing arbitrage transactions on a blockchain.

As discussed in Section 6, rebalancing is more powerful than trading alone for protection against arbitrage. Standard CFMM trades do not improve liquidity. Moreover, no self-funding sequence of trades can, in general, harmonize prices: the agent executing those trades generally cannot eliminate arbitrage while simultaneously breaking even.

3 Arbitrage and Pareto Efficiency

▶ **Theorem 3.** *Arbitrage-prone configurations are not Pareto efficient under rebalancing.*

Proof. Let

$$c := (x_{1,1}, x_{1,2}), \ldots, (x_{n,1}, x_{n,2})$$

be an arbitrage-prone configuration. We construct a rebalancing that leads to a configuration

$$c' := (x'_{1,1}, x'_{1,2}), \ldots, (x'_{n,1}, x'_{n,2})$$

where $F_i(x_{i,1}, x_{i,2}) \leq F_i(x'_{i,1}, x'_{i,2})$ for $i \in \{1, \ldots, n\}$, and at least one inequality is strict, contradicting the hypothesis that c is Pareto efficient.

Because c is arbitrage prone by hypothesis, an arbitrageur starting with a basket of tokens $(b_1, \ldots, b_m)$ can execute a sequence of trades, ending with a basket $(b'_1, \ldots, b'_m)$, where each $b'_j \geq b_j$, for $j \in \{1, \ldots, m\}$, and at least one inequality is strict.

To construct a rebalancing, borrow a basket $(b_1, \ldots, b_m)$, transfer tokens among the pools in a way that mimics that arbitrageur's sequence of trades, producing $(b'_1, \ldots, b'_m)$, where each $b'_j \geq b_j$, and at least one inequality is strict. Repay the loan, leaving the basket $(b'_1 - b_1, \ldots, b'_m - b_m)$. Recall that each of these trades leaves each CFMM's liquidity unchanged.

For each token type T_k for which $b'_k - b_k > 0$, choose any CFMM that trades token type T_k, and transfer $b'_k - b_k$ tokens of type T_k to that pool. This transfer strictly increases that CFMM's liquidity. Repeating this procedure for every token type with $b'_k > b_k$ produces a rebalancing in which every CFMM's liquidity is at least its original value and at least one CFMM's liquidity is strictly greater, contradicting the hypothesis that c is Pareto efficient. ◀

▶ **Definition 4.** *A* valuation *for a network of CFMMs trading token types $T_1, \ldots, T_m$ is a vector $V = (V_1, \ldots, V_m) \in \mathbb{R}^m_{>0}$, where V_i is the value assigned to one unit of token T_i.*

▶ **Definition 5.** *A valuation $V = (V_1, \ldots, V_m)$ is a* global valuation *for a configuration if, for every CFMM that trades token types T_p and T_q,*

$$\frac{V_p}{V_q} = P_{pq},$$

where P_{pq} is the spot price of T_p measured in units of T_q.

The proof of the following lemma appears in the appendix.

▶ **Lemma 6.** *Any arbitrage-free configuration has a global valuation.*

The remainder of this section uses the global valuation to assign a linear value to each CFMM. We show that increasing a CFMM's liquidity necessarily increases its value, while the total value of all CFMMs is preserved under rebalancing.

In an arbitrage-free configuration with global valuation V, define the *value* of the CFMM state $(x_{i,1}, x_{i,2})$, which trades token types T_j and T_k, by

$$\Phi_V(x_{i,1}, x_{i,2}) := V_j x_{i,1} + V_k x_{i,2}.$$

The tangent line to the level set

$$\{(X_1, X_2) \mid F_i(X_1, X_2) = F_i(x_{i,1}, x_{i,2})\}$$

at point $(x_{i,1}, x_{i,2})$ is:

$$L_i := \{(X_1, X_2) \mid \Phi_V(X_1, X_2) = \Phi_V(x_{i,1}, x_{i,2})\}. \tag{1}$$

The tangent L_i has slope $-\frac{V_j}{V_k}$, and divides $\mathbb{R}^2_{>0}$ into two half-planes. Points in the half-plane below L_i have lower value than $\Phi_V(x_{i,1}, x_{i,2})$ and points above have higher values.

▶ **Lemma 7.** *$\Phi_V(x'_{i,1}, x'_{i,2}) < \Phi_V(x_{i,1}, x_{i,2})$ if and only if $(x'_{i,1}, x'_{i,2})$ lies in the half-plane below L_i.*

Proof. From Equation (1). ◀

Every point with higher liquidity lies above L_i.

▶ **Lemma 8.** *The region $\{(X_1, X_2) \mid F_i(X_1, X_2) \geq F_i(x_{i,1}, x_{i,2})\}$ lies entirely in the half-plane above L_i.*

Proof. Since the upper level set of a log-concave function is convex, the tangent line L_i is a supporting line. ◀

▶ **Corollary 9.** *Any rebalancing that increases the liquidity $F_i(x_{i,1}, x_{i,2})$ must also increase the value $\Phi_V(x_{i,1}, x_{i,2})$.*

▶ **Corollary 10.** *Any rebalancing that decreases the value $\Phi_V(x_{i,1}, x_{i,2})$ must also decrease the liquidity $F_i(x_{i,1}, x_{i,2})$.*

▶ **Theorem 11.** *Any arbitrage-free configuration is Pareto efficient under rebalancing.*

Proof. We will show that from an arbitrage-free configuration, any rebalancing that increases one CFMM's liquidity must decrease another's.

Consider a rebalancing that carries each $(x_{i,1}, x_{i,2})$ to $(x'_{i,1}, x'_{i,2})$, where $F_i(x'_{i,1}, x'_{i,2}) > F_i(x_{i,1}, x_{i,2})$. By Corollary 9, $\mathcal{A}_i$'s value increases: $\Phi_V(x'_{i,1}, x'_{i,2}) > \Phi_V(x_{i,1}, x_{i,2})$.

Because rebalancing neither creates nor destroys tokens, the sum of the CFMMs' values is invariant:

$$\sum_{i=1}^{n} \Phi_V(x_{i,1}, x_{i,2}) = \sum_{i=1}^{n} \Phi_V(x'_{i,1}, x'_{i,2}).$$

Therefore there exists some j such that

$$\Phi_V(x'_{j,1}, x'_{j,2}) < \Phi_V(x_{j,1}, x_{j,2}).$$

By Corollary 10, $F_j(x'_{j,1}, x'_{j,2}) < F_j(x_{j,1}, x_{j,2})$, implying that the rebalancing strictly lowered at least one CFMM's liquidity. ◀

Theorems 3 and 11 immediately imply the main result:

▶ **Theorem 12.** *A configuration is arbitrage free if and only if it is Pareto efficient.*

4 Rebalancing as Convex Optimization

Parameters:	$n \in \mathbb{N}$ (Number of CFMMs) $x_{i,j} \quad i \in \{1,\dots,n\}, j \in \{1,2\}$ (Initial pool sizes) $\mathcal{T} = \{(i,j,k,\ell) \mid$ transfer allowed between pools $x_{i,j}, x_{k,\ell}\}$
Decision variables:	$x'_{i,j} \quad i \in \{1,\dots,n\}, j \in \{1,2\}$ (Final pool sizes) $\delta_{i,j,k,\ell}$ Amount (+/-) transferred along $(i,j,k,\ell) \in \mathcal{T}$
Objective:	max $L(x'_{1,1}, x'_{1,2}, \dots, x'_{n,1}, x'_{n,2}) := \sum_{i=1}^{n} \log F_i(x'_{i,1}, x'_{i,2})$
Subject to:	$x'_{i,j} \geq 0$ for $i \in \{1,\dots,n\}, j \in \{1,2\}$ (non-negativity) $\log F_i(x_{i,1}, x_{i,2}) \leq \log F_i(x'_{i,1}, x'_{i,2}), \quad i = 1,\dots,n$ (liquidity non-reduction) $x'_{i,j} = x_{i,j} + \sum_{\substack{k,\ell: \\ (k,\ell,i,j)\in\mathcal{T}}} \delta_{k,\ell,i,j} - \sum_{\substack{k,\ell: \\ (i,j,k,\ell)\in\mathcal{T}}} \delta_{i,j,k,\ell}$ (conservation)

Figure 1 Liquidity Rebalancing Optimization.

In this section, we show how to find a unique rebalancing to protect a configuration from arbitrage. We require the solution to be unique because mutually mistrustful LPs must independently verify the result. Any ambiguity would lead to missed rebalancing opportunities.

Specifically, we show that finding an arbitrage-free rebalancing can be cast as a convex optimization problem. Moreover, we can exploit Theorem 12 to simplify the optimization program's constraints by explicitly seeking a Pareto-efficient final configuration instead of a (possibly more complicated) arbitrage-free final configuration.

4.1 Parameters

Rebalancings can transfer tokens only between pools of the same type. We capture which transfers are permissible through the index set:

$$\mathcal{T} := \{(i,j,k,\ell) \mid \text{transfers allowed between pools } x_{i,j}, x_{k,\ell}\}.$$

The other parameters are the starting pool sizes:

$$(x_{1,1}, x_{1,2}), \dots, (x_{n,1}, x_{n,2}).$$

4.2 Decision Variables

There are transfer amounts

$$\delta_{i,j,k,\ell}, \qquad (i,j,k,\ell) \in \mathcal{T}.$$

If $\delta_{i,j,k,\ell} > 0$, then $|\delta_{i,j,k,\ell}|$ tokens are transferred from the pool corresponding to $x_{i,j}$ to the pool corresponding to $x_{k,\ell}$. If $\delta_{i,j,k,\ell} < 0$, the transfer is in the opposite direction. If $\delta_{i,j,k,\ell} = 0$, no transfer occurs.

The remaining decision variables are the new pool sizes:

$$(x'_{1,1}, x'_{1,2}), \ldots, (x'_{n,1}, x'_{n,2}).$$

4.3 Constraints

The *non-negativity* constraints require new pool sizes to be non-negative:

$$x'_{i,j} \geq 0 \quad \text{for } i \in \{1,\ldots,n\}, j \in \{1,2\}.$$

This system of linear inequalities defines a convex set.

The *conservation* constraints define the relationship between the pool sizes and transfers. For $i \in \{1,\ldots,n\}, j \in \{1,2\}$,

$$x'_{i,j} = x_{i,j} + \sum_{\substack{k,\ell: \\ (k,\ell,i,j)\in\mathcal{T}}} \delta_{k,\ell,i,j} - \sum_{\substack{k,\ell: \\ (i,j,k,\ell)\in\mathcal{T}}} \delta_{i,j,k,\ell}.$$

This system of linear equalities defines a convex set.

The *liquidity non-reduction* constraints require that rebalancing not decrease any liquidity levels:

$$F_i(x_{i,1}, x_{i,2}) \leq F_i(x'_{i,1}, x'_{i,2}), \quad i \in \{1,\ldots,n\}.$$

This constraint, however, is generally not convex, but it can be made convex through a log transform:

$$\log F_i(x_{i,1}, x_{i,2}) \leq \log F_i(x'_{i,1}, x'_{i,2}), \quad i \in \{1,\ldots,n\}.$$

Since the left-hand side is constant and the right-hand side is concave, the feasible region defined by this inequality is convex.

4.4 Objective Function

The objective function can express goals that span individual CFMMs, thereby promoting social welfare.

▶ **Theorem 13.** *Let $L : \mathbb{R}^n \to \mathbb{R}$ be strictly increasing in each argument. Any configuration that maximizes $L(F_1(x'_{1,1}, x'_{1,2}), \ldots, F_n(x'_{n,1}, x'_{n,2}))$ over feasible rebalancings is arbitrage-free.*

Proof. Suppose c maximizes $L(F_1(x_{1,1}, x_{1,2}), \ldots, F_n(x_{n,1}, x_{n,2}))$ but is not arbitrage-free. By Theorem 12, c is not Pareto efficient, so there is a rebalancing that takes each $(x_{i,1}, x_{i,2})$ in c to $(x'_{i,1}, x'_{i,2})$, where each $F_i(x_{i,1}, x_{i,2}) \leq F_i(x'_{i,1}, x'_{i,2})$, and at least one inequality is strict. It follows that $L(F_1(x_{1,1}, x_{1,2}), \ldots, F_n(x_{n,1}, x_{n,2})) < L(F_1(x'_{1,1}, x'_{1,2}), \ldots, F_n(x'_{n,1}, x'_{n,2}))$, contradicting the hypothesis. ◀

▶ **Corollary 14.** *Any configuration that maximizes a weighted sum or weighted product of liquidities*

$$\max \sum_{i=1}^{n} w_i F_i(x'_{i,1}, x'_{i,2}) \quad \textit{or} \quad \max \prod_{i=1}^{n} F_i(x'_{i,1}, x'_{i,2})^{w_i}$$

for $w_i > 0$ is arbitrage-free.

For the objective function, we choose to maximize the product of the CFMMs' liquidities. Maximizing this objective is not directly a convex optimization problem, but it can be made convex through a log transform:

$$\max \quad L(x'_{1,1}, x'_{1,2}, \ldots, x'_{n,1}, x'_{n,2}) := \sum_{i=1}^{n} \log F_i(x'_{i,1}, x'_{i,2}).$$

Because each $\log F_i(\cdot, \cdot)$ is concave on its domain by hypothesis, $L(\cdot, \cdot)$ is concave as the sum of concave functions. Figure 1 shows the complete convex optimization problem.

▶ **Theorem 15.** *The optimization program in Figure 1 is a convex optimization problem.*

Proof. The claim follows because we have shown that (1) the objective function to be maximized is concave, and (2) the feasible region defined by the constraints is convex. ◀

▶ **Theorem 16.** *The optimization program in Figure 1 has a unique solution.*

Proof. Because each $\log F_i$ is strictly concave, the objective function is strictly concave. A strictly concave function has at most one maximizer over a convex feasible set. ◀

As noted earlier, if CFMMs have additional convex constraints, they can be incorporated into the optimization program, though care must be taken that these additional constraints do not render the problem infeasible.

5 Mixed Populations

So far we have considered rebalancing among a population of CFMMs, all of whom are willing to rebalance with one another to mitigate a common arbitrage vulnerability.

Some LPs might be unwilling to allow a rebalancing agent to transfer tokens directly in and out of their CFMMs' pools. We distinguish between *active* CFMMs who are willing to allow direct pool-to-pool transfers, and *passive* CFMMs, who are not. We assume there is at least one active CFMM. Passive CFMMs cannot be ignored, because their spot prices might expose the others to arbitrage losses, but passive CFMMs can interact with a rebalancing agent only through standard trades, not direct pool-to-pool transfers.

A *price oracle* is a market maker that models an external market that trades at a fixed reference price. For example, consider a centralized exchange with (effectively) inexhaustible liquidity and no slippage that is willing to sell arbitrary amounts of a token at a constant price. This section shows how rebalancing can be extended to accommodate both passive CFMMs and price oracle market makers.

5.1 Active vs. Passive CFMMs

Recall the circular trade scenario where $\mathcal{A}, \mathcal{B}, \mathcal{C}$ have the constant-product trading function $F(x_{i,1}, x_{i,2}) := x_{i,1} \cdot x_{i,2} = 3$, in a configuration where $\mathcal{A}$ has €1 and \$3, $\mathcal{B}$ has £1 and €3, and $\mathcal{C}$ has \$1 and £3. Because this configuration is arbitrage-prone, the LPs for $\mathcal{A}$ decide to rebalance, but this time without cooperation from the LPs for $\mathcal{B}, \mathcal{C}$. The rebalancing agent does the following:

- Transfer \$0.5 out of $\mathcal{A}$,
- Trade \$0.5 to $\mathcal{C}$ for £1.0,
- Trade £1.0 to $\mathcal{B}$ for €1.5, and
- Transfer €1.5 into $\mathcal{A}$.

Note that each trade is compatible with that CFMM's trading function. The resulting configuration is arbitrage-free. Because $\mathcal{B}, \mathcal{C}$ performed only trades, their liquidity levels remain unchanged, but the liquidity for $\mathcal{A}$ has increased from 3 to 6.25. By actively rebalancing, $\mathcal{A}$ has effectively executed the threatened arbitrage and allocated the profit to itself. In general, that profit would be spread across the active CFMMs.

A rebalancing is *restricted* if it does not change passive CFMMs' liquidities. A restricted rebalancing can be implemented by executing both direct transfers between active CFMMs and trades with any CFMMs, while ensuring the process is self-funding by making sure that every token ends up in some CFMM's pool.

As before, consider CFMMs $\mathcal{A}_1, \ldots, \mathcal{A}_n$, with trading functions $F_1(x, y), \ldots, F_n(x, y)$. The *active* CFMMs $\mathcal{A}_1, \ldots, \mathcal{A}_p$ support arbitrary transfers in and out of their pools, for $p > 0$, while the *passive* CFMMs $\mathcal{A}_{p+1}, \ldots, \mathcal{A}_n$ support only standard trades, that is, transfers that leave their liquidities unchanged.

The network of CFMMs forms a *CFMM graph*, where the CFMMs are the vertices, and there is an edge linking each pair of CFMMs $\mathcal{A}, \mathcal{B}$ that trade the same token type. Without loss of generality, we assume this graph is *connected*, meaning that between any pair of vertices $\mathcal{A}, \mathcal{B}$, there is a sequence $\mathcal{A}_1, \ldots, \mathcal{A}_k$ called a *path*, where consecutive CFMMs are connected by an edge, $\mathcal{A} = \mathcal{A}_1$, and $\mathcal{B} = \mathcal{A}_k$.

▶ **Lemma 17.** *Let T be a token type traded by some CFMM $\mathcal{A}_1$. An agent in possession of $\epsilon > 0$ tokens of type T can increase the liquidity of some active CFMM while leaving all other liquidity levels unchanged.*

Proof. By induction on ℓ, the length of the shortest path $\mathcal{A}_1, \ldots, \mathcal{A}_\ell$ linking $\mathcal{A}_1$ to an active CFMM $\mathcal{A}_\ell$ in the CFMM graph.

For the base case, $\ell = 1$, so $\mathcal{A}_1$ is active. The agent simply transfers the ϵ tokens into $\mathcal{A}_1$'s pool for T, increasing $\mathcal{A}_1$'s liquidity while leaving the others unchanged.

For the induction step, assume the result for paths shorter than $\ell > 1$. The agent first trades at $\mathcal{A}_1$ the ϵ tokens of type T for $\epsilon' > 0$ tokens of type T', leaving $\mathcal{A}_1$'s liquidity unchanged. The agent now holds $\epsilon' > 0$ tokens of type T' traded by $\mathcal{A}_2$, a passive CFMM linked to an active CFMM by a path shorter than ℓ. The claim follows from the induction hypothesis. ◀

▶ **Theorem 18.** *Arbitrage-prone configurations are not Pareto efficient under restricted rebalancing.*

Proof. Let c be an arbitrage-prone configuration.

As in the proof of Theorem 3, an arbitrageur starting with a basket of tokens $(b_1, \ldots, b_m)$ can execute a sequence of trades starting from c, ending with the basket $(b'_1, \ldots, b'_m)$, where each $b'_j \geq b_j$, for $j \in \{1, \ldots, m\}$, and at least one inequality is strict.

For each token type T_k for which $b'_k - b_k > 0$, Lemma 17 implies that the agent can transfer those surplus tokens to some active CFMM trading T_k, increasing that CFMM's liquidity while leaving all other liquidity levels unchanged. Repeating this procedure for every profitable token type yields a rebalancing in which every CFMM's liquidity is at least its original value and at least one CFMM's liquidity is strictly greater, contradicting the hypothesis that c is Pareto efficient. ◀

▶ **Theorem 19.** *Any arbitrage-free configuration is Pareto efficient under restricted rebalancing.*

Proof. The claim follows immediately from Theorem 11. ◀

The *restricted rebalancing convex optimization program* requires only one change to the optimization program of Figure 1: instead of maximizing the product of *all* CFMM liquidities, the restricted program maximizes the product of the active CFMM liquidities *only* (replacing n with p):

$$\max \quad L(x'_{1,1}, x'_{1,2}, \ldots, x'_{p,1}, x'_{p,2}) := \sum_{i=1}^{p} \log F_i(x'_{i,1}, x'_{i,2}). \tag{2}$$

The passive CFMMs' liquidities are bound only by the liquidity non-reduction constraint:

$$F_i(x_{i,1}, x_{i,2}) \leq F_i(x'_{i,1}, x'_{i,2}), \quad i \in \{p+1, \ldots, n\}.$$

Let $c' = (x'_{1,1}, x'_{1,2}), \ldots, (x'_{n,1}, x'_{n,2})$ be the optimal configuration produced by solving the restricted rebalancing program. Note that because the program's objective function is strictly concave, and the feasible region is convex, the solution c' is unique.

We claim that in c', the passive CFMMs' non-reduction constraints hold with equality.

▶ **Lemma 20.**

$$F_i(x_{i,1}, x_{i,2}) = F_i(x'_{i,1}, x'_{i,2}), \quad i \in \{p+1, \ldots, n\}.$$

Proof. Assume by way of contradiction that in c', $F_i(x'_{i,1}, x'_{i,2}) > F_i(x_{i,1}, x_{i,2})$ for some passive $\mathcal{A}_i$. Because $F_i(\cdot, \cdot)$ is strictly increasing in both arguments, we can transfer $\epsilon > 0$ tokens out of pool $x'_{i,1}$ without violating the program's constraints: $F_i(x'_{i,1} - \epsilon, x'_{i,2}) \geq F_i(x_{i,1}, x_{i,2})$. By Lemma 17, we can use these tokens to increase the liquidity of some active CFMM while leaving all other liquidity levels unchanged. This action increases the objective function's value, contradicting the hypothesis that c' is optimal. ◀

▶ **Lemma 21.** *Configuration c' is arbitrage-free.*

Proof. By Theorem 12, it suffices to show that c' is Pareto efficient. Assume by way of contradiction that there is a rebalancing taking c' to a configuration $c^\dagger = (x^\dagger_{1,1}, x^\dagger_{1,2}), \ldots, (x^\dagger_{n,1}, x^\dagger_{n,2})$ where for $i \in \{1, \ldots, n\}$, $F_i(x^\dagger_{i,1}, x^\dagger_{i,2}) \geq F_i(x'_{i,1}, x'_{i,2})$, and at least one inequality is strict.

If $c^\dagger$ strictly increases an active CFMM's liquidity, then $c^\dagger$ has a higher objective function value than c', while continuing to satisfy the restricted program's other constraints, contradicting the hypothesis that c' is Pareto efficient.

If, on the other hand, no active CFMM's liquidity increases, then every strict increase must occur at a passive CFMM. By Lemma 20, however, every passive CFMM already satisfies its liquidity non-reduction constraint with equality in c'. Any strict increase in a passive CFMM's liquidity would continue to satisfy the restricted program's constraints, and by Lemma 17 the additional liquidity could be transferred to an active CFMM without decreasing any other liquidity levels, contradicting the optimality of c'. ◀

In summary, the optimization program of Figure 1 extends naturally to encompass passive CFMMs simply by omitting their liquidities from the objective function.

5.2 Price Oracles

A *price oracle* models a centralized exchange that sets the "market price" for a token by trading at that price. Recall the circular trade scenario with CFMMs $\mathcal{A}, \mathcal{B}, \mathcal{C}$. As before, $\mathcal{A}$ and $\mathcal{B}$ have constant-product trading functions $F_i(x_{i,1}, x_{i,2}) := x_{i,1} \cdot x_{i,2} = 3$. $\mathcal{C}$, however, is an oracle market maker with unlimited liquidity that trades pounds and dollars at par. Initially, $\mathcal{A}$ has €1 and \$3, $\mathcal{B}$ has £1 and €3, and $\mathcal{C}$ has inexhaustible amounts of dollars and pounds. This configuration is arbitrage-prone, so the LPs for $\mathcal{A}$ alone decide to rebalance. The rebalancing agent:

- Transfers $\$(\sqrt{3}-1) \approx \$0.732051$ from $\mathcal{A}$,
- Trades $\$(\sqrt{3}-1)$ for $£(\sqrt{3}-1)$ at $\mathcal{C}$,
- Trades $£(\sqrt{3}-1)$ for $€(3-\sqrt{3}) \approx €1.26795$ at $\mathcal{B}$, and
- Transfers $€(3-\sqrt{3})$ to $\mathcal{A}$.

In the end, $\mathcal{A}$ has pools $(4-\sqrt{3}, 4-\sqrt{3})$ and $\mathcal{B}$ has $(\sqrt{3}, \sqrt{3})$. $\mathcal{A}$'s liquidity has increased from 3 to $19-8\sqrt{3} \approx 5.14359$, while the others, being passive, remain unchanged. This final configuration is arbitrage-free, and hence Pareto-efficient.

To create a convex optimization program that encompasses one or more oracle AMMs, the key insight is that a price-setting oracle AMM is simply a passive CFMM with linear trading function $F_i(x_{i,1}, x_{i,2}) := ax_{i,1} + bx_{i,2}$, where $a, b > 0$, initialized with sufficient reserves to ensure no pool is exhausted by the rebalancing. (Alternatively, one could permit oracle AMMs to have negative reserve balances in the optimization program.)

6 Remarks

6.1 Alternatives to CFMMs

Although we have focused on CFMMs, the equivalence between arbitrage-free and Pareto-efficient configurations extends to any financial system prone to arbitrage where parties have distinct private utility functions.

For example, suppose Alice owns a basket containing one gold token and one silver token. She subjectively values gold at 1 dollar per token and silver at 2 dollars per token, and thus values her basket at 3 dollars. Bob also owns a basket containing one gold token and one silver token. He subjectively values gold at 2 dollars per token and silver at 1 dollar per token, and thus also values his basket at 3 dollars. Both parties will accept any trade that increases the perceived value of their baskets according to their own valuations.

Suppose an arbitrageur offers Alice $((1+\epsilon)/2)$ silver tokens in exchange for her gold token. She will accept, because she values the silver she receives at $(1+\epsilon)$ dollars, while she values the gold she gives up at only 1 dollar. Her new basket is worth $(3+\epsilon)$ dollars according to her valuation. Similarly, the arbitrageur can offer Bob $((1+\epsilon)/2)$ gold tokens in exchange for his silver token. Bob will also accept, and he too will value his new basket at $(3+\epsilon)$ dollars.

Collectively, the arbitrageur receives one gold token and one silver token while paying out $((1+\epsilon)/2)$ of each. The arbitrageur therefore earns a profit of $((1-\epsilon)/2)$ gold tokens and $((1-\epsilon)/2)$ silver tokens.

By comparison, a rebalancing might exchange Alice's gold token for Bob's silver token, leaving both parties in an arbitrage-free, Pareto-efficient configuration in which each values their basket at 4 dollars.

6.2 Alternatives to Rebalancing

As an alternative to rebalancing, LPs might instead execute ordinary trades among the CFMMs to mitigate their exposure to arbitrage. Here we use a simple example to illustrate why LPs might prefer defensive rebalancing (as introduced here) to trading. In general, however, defensive trading cannot be *self-funding*: the trading agent must either inject tokens into the system of CFMMs or withdraw them.

Consider the following example. Recall the circular trade scenario where $\mathcal{A}, \mathcal{B}, \mathcal{C}$ are in the arbitrage-prone configuration $\mathcal{A} : (€1, \$3)$, $\mathcal{B} : (£1, €3)$, and $\mathcal{C} : (\$1, £3)$. Figure 2 shows an optimization program that seeks an arbitrage-free configuration. It uses slack variables $\sigma_\$, \sigma_€, \sigma_£$ to represent the amount of each token the agent adds or removes from $\mathcal{A}, \mathcal{B}, \mathcal{C}$. The objective function is to minimize the sum of the squares of the slack variables. If a self-funding solution were possible, these variables would all be zero.

This optimization problem is not convex, but it is small enough to be readily solved by off-the-shelf solvers[3]. In this case, minimizing the sum of the squares of the slack variables requires transferring $\sigma_\$ = \sigma_€ = \sigma_£ = 4 - 2\sqrt{3} \approx 0.535898$ units of each currency from the three CFMMs to the agent, yielding a solution that is both less effective than rebalancing (because the CFMMs permanently surrender assets) and more complicated (because the agent must retain custody of the withdrawn assets).

Parameters:	$(x_{1,1}, x_{1,2}), \quad (x_{2,1}, x_{2,2}), \quad (x_{3,1}, x_{3,2})$
Decision variables:	$(x'_{1,1}, x'_{1,2}), \quad (x'_{2,1}, x'_{2,2}), \quad (x'_{3,1}, x'_{3,2})$ $\sigma_€, \quad \sigma_£, \quad \sigma_\$$
Objective:	$\min \quad \sigma_€^2 + \sigma_£^2 + \sigma_\$^2$
Subject to:	$x'_{1,1}x'_{1,2} = x_{1,1}x_{1,2} = 3 \quad$ (Invariant for $\mathcal{A}$) $x'_{2,1}x'_{2,2} = x_{2,1}x_{2,2} = 3 \quad$ (Invariant for $\mathcal{B}$) $x'_{3,1}x'_{3,2} = x_{3,1}x_{3,2} = 3 \quad$ (Invariant for $\mathcal{C}$) $x_{1,1} + x_{2,2} = x'_{1,1} + x'_{2,2} + \sigma_€$ $x_{2,1} + x_{3,2} = x'_{2,1} + x'_{3,2} + \sigma_£$ $x_{3,1} + x_{1,2} = x'_{3,1} + x'_{1,2} + \sigma_\$$ $x'_{1,1}x'_{2,1}x'_{3,1} = x'_{1,2}x'_{2,2}x'_{3,2} \quad$ (Arbitrage-free condition) $x'_{i,j} > 0 \quad$ for $i \in \{1,2,3\}, j \in \{1,2\}$

Figure 2 Arbitrage protection with trades alone.

7 Related Work

The notion of a CFMM is due to Angeris *et al.* [3], who define the notion of a CFMM, analyze how CFMMs can serve as decentralized price oracles, and formulate the *optimal arbitrage problem*, namely, how to exploit an arbitrage-prone configuration most profitably. Angeris *et al.* [4] consider how to use convex optimization to detect arbitrage-prone configurations, and how best to exploit them, a problem they call *optimal routing*. Our use of optimization complements theirs: rather than exploiting arbitrage opportunities, we compute optimal rebalancings that eliminate them. Angeris *et al.* [5] investigate what happens when a CFMM displaces an external market as a price setter.

[3] For this example, we used Mathematica's `NMinimize`.

Milionis *et al.* [17] propose *loss-versus-rebalancing* (LVR) as an improvement over divergence loss as a way of measuring LP costs in configurations with a central reference market. LVR measures the difference between the value of a hypothetical investment strategy that responds optimally to market price changes and the value of an LP's actual investment in a CFMM, which responds only through arbitrage. Defensive rebalancing promises to reduce LVR by allowing LPs to adjust their pool sizes dynamically in response to price changes at other markets.

Danos *et al.* [11] formulate various arbitrage problems in a network of AMMs as optimization programs. Angeris *et al.* [2] describe how a variety of problems associated with CFMMs can be formulated as optimization problems. Wang *et al.* [20] perform an empirical investigation of arbitrage transactions in Uniswap V2, reporting on the frequency of arbitrage opportunities, how often they are exploited, and their profitability. Bartoletti *et al.* [6] describe arbitrage between AMM pairs as a two-party game. Dimitri [12] analyzes how arbitrage can affect CFMM parameters such as pool sizes, prices, and liquidity. Milionis *et al.* [16] study how block time affects arbitrage profits.

As a way to avoid arbitrage altogether, Krishnamachari *et al.* [13] describe *dynamic curves*, which allow AMMs to adjust their trading curves to match oracle-reported prices.

In classical economics, the *First Welfare Theorem* states that, under suitable assumptions, every competitive equilibrium is Pareto efficient, while the *Second Welfare Theorem* states that, under additional convexity assumptions, every Pareto-efficient allocation can be supported as a competitive equilibrium after an appropriate redistribution of initial endowments [15, Chapter 16]. Theorems 3 and 11 bear a strong resemblance to the classical welfare theorems and might at first seem to be special cases. The analogy seems close: CFMMs play the role of consumers, tokens the role of commodities, liquidity the role of utility, rebalancing the role of feasible reallocation, and a common token valuation the role of a competitive price vector. Nevertheless, a direct reduction to the classical welfare theorems is not immediate. In particular, CFMMs do not explicitly optimize utility subject to exogenously specified budget constraints, and the feasible reallocations are constrained by the network of token pools. Thus, establishing Theorems 3 and 11 as formal consequences of those theorems would require a careful correspondence between the two models.

8 Conclusion

8.1 Transaction Costs and Fees

In practice, CFMMs charge *trading fees*. A CFMM $\mathcal{A}_i$ with trading function $F_i(x_1, x_2)$ retains a fraction $\gamma_i \in (0, 1]$ of each incoming trade. If a trader transfers $\delta_\$$ dollars to $\mathcal{A}_i$ and receives $\delta_€$ euros in return, the pool invariants update according to $F_i(x_{i,1} + \gamma_i\delta_\$, x_{i,2} - \delta_€) = k$. (The un-discounted fraction $(1 - \gamma_i)\delta_\$$ of dollars is typically retained by LPs as fee revenue.) For example, in Uniswap, γ_i depends on the pool's fee tier.

Adding fees to the model requires a few changes to Figure 1. There is a new parameter: each $\mathcal{A}_i$ has a fee parameter γ_i. The conservation constraint is split into two parts:

$$\text{for } i \le p: \quad x'_{i,j} = x_{i,j} + \sum_{\substack{k,\ell: \\ (k,\ell,i,j)\in\mathcal{T}}} \delta_{k,\ell,i,j} - \sum_{\substack{k,\ell: \\ (i,j,k,\ell)\in\mathcal{T}}} \delta_{i,j,k,\ell}, \quad \text{(active conservation)}$$

$$\text{for } i > p: \quad x'_{i,j} = x_{i,j} + \sum_{\substack{k,\ell: \\ (k,\ell,i,j)\in\mathcal{T}}} \gamma_i\delta_{k,\ell,i,j} - \sum_{\substack{k,\ell: \\ (i,j,k,\ell)\in\mathcal{T}}} \delta_{i,j,k,\ell}, \quad \text{(passive conservation)}$$

The conservation equations for active CFMMs are unchanged (we assume pool-to-pool transfers between active CFMMs do not incur trading fees), while the conservation constraints for passive CFMMs reflect their respective fees.

Agents also typically incur a small fixed cost with each trade. Angeris *et al.* [4] observe that incorporating such costs in the optimization program transforms a convex program into a mixed-integer convex program. Exact solutions for such programs may be computationally expensive, but efficient approximate solvers exist (see the cited reference for details).

The optimization program could be extended in a number of ways. It might impose additional constraints, such as enforcing upper or lower bounds on pool sizes. The objective function could be extended to take into account costs such as inventory risk or gas fees.

8.2 Practical Issues

Rebalancing requires active CFMMs to move tokens directly from one pool to another without changing LPs' ownership shares. Existing CFMMs (*e.g.*, [1, 14, 7, 10, 19]) provide mechanisms for LPs to add or remove liquidity, but none currently provide the exact functionality needed to support rebalancing.

Rebalancing also raises security concerns. When presented with a rebalancing proposal, it is trivial for a CFMM to check that its own liquidity does not decrease. However, verifying that the proposal maximizes the global objective function is expensive in gas costs. For this reason, the LPs might instead rely on a trusted third party to manage rebalancing, in the same way DEXes currently engage services like Chainlink [8] to provide oracles, or LayerZero [21] or Axelar [18] to provide cross-chain messaging. In the context of the Ethereum Virtual Machine, one could instead install a deterministic solver as a "precompiled contract" [9], thereby providing a low-cost way for a CFMM to verify that a proposed rebalancing is fair and optimal.

In practice, successful defensive rebalancing requires executing a rebalancing transaction before competing arbitrage transactions eliminate the opportunity. Mechanisms for achieving this goal, including transaction sequencing, private order flow, auctions, or proposer-builder separation, are beyond this paper's scope. One direction for future work is the design of decentralized mechanisms that detect arbitrage opportunities, compute Pareto-efficient rebalancings, and obtain sufficient execution priority to realize those rebalancings in adversarial transaction-ordering environments.

References

1 Hayden Adams, Noah Zinsmeister, Moody Salem, River Keefer, and Dan Robinson. Uniswap v3 Core, March 2021. URL: `https://uniswap.org/whitepaper-v3.pdf`.

2 Guillermo Angeris, Akshay Agrawal, Alex Evans, Tarun Chitra, and Stephen Boyd. Constant function market makers: Multi-asset trades via convex optimization, 2021. `arXiv:2107.12484`.

3 Guillermo Angeris and Tarun Chitra. Improved Price Oracles: Constant Function Market Makers. *SSRN Electronic Journal*, 2020. `doi:10.2139/ssrn.3636514`.

4 Guillermo Angeris, Tarun Chitra, Alex Evans, and Stephen Boyd. Optimal Routing for Constant Function Market Makers, 2022. Version Number: 1. `doi:10.48550/arXiv.2204.05238`.

5 Guillermo Angeris, Alex Evans, and Tarun Chitra. When does the tail wag the dog? Curvature and market making. *arXiv:2012.08040 [q-fin]*, December 2020. arXiv: 2012.08040. `arXiv:2012.08040`.

6 Massimo Bartoletti, James Hsin-yu Chiang, and Alberto Lluch-Lafuente. A theory of Automated Market Makers in DeFi. *arXiv:2102.11350 [cs]*, April 2021. arXiv: 2102.11350. arXiv:2102.11350.

7 Bullish. Solana | Digital Assets | Bullish. URL: https://www.bullish.com/us/digital-assets/solana.

8 {Chainlink, Inc.}. Chainlink 2.0 and the future of Decentralized Oracle Networks | Chainlink, December 2025. URL: https://chain.link/whitepaper.

9 EVM codes. Precompiled contracts, 2022. As of 25 January 2026. URL: https://www.evm.codes/precompiled.

10 CoinEx. CoinEx Automated Market Making (AMM) User Guide, September 2025. URL: https://support.coinex.com/hc/en-us/articles/4411281577369-CoinEx-Automated-Market-Making-AMM-User-Guide.

11 Vincent Danos, Julien Prat, and Hamza El Khalloufi. Global Order Routing on Exchange Networks | Financial Cryptography and Data Security. FC 2021 International Workshops, March 2021. URL: https://dl.acm.org/doi/10.1007/978-3-662-63958-0_19.

12 Nicola Dimitri. Arbitrage in automated market makers. *Frontiers in Blockchain*, Volume 7 - 2024, 2024. doi:10.3389/fbloc.2024.1455070.

13 Bhaskar Krishnamachari, Qi Feng, and Eugenio Grippo. Dynamic Curves for Decentralized Autonomous Cryptocurrency Exchanges. *arXiv:2101.02778 [q-fin]*, January 2021. arXiv: 2101.02778. arXiv:2101.02778.

14 Fernando Martinelli and Nikolai Mushegian. Balancer: A non-custodial portfolio man- ager, liquidity provider, and price sensor. https://balancer.finance/whitepaper/, 2019. As of 2 February 2021.

15 Andreu Mas-Colell, Michael D. Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

16 Jason Milionis, Ciamac C. Moallemi, and Tim Roughgarden. Extended abstract: The effect of trading fees on arbitrage profits in automated market makers. In Aleksander Essex, Shin'ichiro Matsuo, Oksana Kulyk, Lewis Gudgeon, Ariah Klages-Mundt, Daniel Perez, Sam Werner, Andrea Bracciali, and Geoff Goodell, editors, *Financial Cryptography and Data Security. FC 2023 International Workshops - Voting, CoDecFin, DeFi, WTSC, Bol, Brač, Croatia, May 5, 2023, Revised Selected Papers*, volume 13953 of *Lecture Notes in Computer Science*, pages 262–265. Springer, 2023. doi:10.1007/978-3-031-48806-1_17.

17 Jason Milionis, Ciamac C. Moallemi, Tim Roughgarden, and Anthony Lee Zhang. Automated Market Making and Loss-Versus-Rebalancing, September 2022. arXiv:2208.06046 [math, q-fin]. arXiv:2208.06046.

18 Axelar Research. Axelar network: Connecting applications with blockchain ecosystems, 2022. As of 8 May 2023. arXiv:2011.12783.

19 Skate. Skate AMM Fundamentals. URL: https://docs.skatechain.org/skate-amm/introduction.

20 Ye Wang, Yan Chen, Haotian Wu, Liyi Zhou, Shuiguang Deng, and Roger Wattenhofer. Cyclic Arbitrage in Decentralized Exchanges, January 2022. arXiv:2105.02784 [cs, q-fin]. arXiv:2105.02784.

21 Ryan Zarick, Bryan Pellegrino, and Caleb Banister. Layerzero: Trustless omnichain interoperability protocol, 2022. As of 8 May 2023. URL: https://layerzero.network/pdf/LayerZero_Whitepaper_Release.pdf.

A Appendix

▶ **Lemma 6.** *Any arbitrage-free configuration has a global valuation.*

Proof. Suppose the CFMM network trades token types $(T_1, \ldots, T_m)$, and let $p_{jk} > 0$ denote the spot price of T_j in units of T_k at any CFMM trading the pair (T_j, T_k). We first show that arbitrage freedom implies that the product of spot prices around every cycle is 1.

Consider a cycle

$$T_{i_1}, T_{i_2}, \ldots, T_{i_r}, T_{i_1}.$$

If

$$p_{i_1 i_2} p_{i_2 i_3} \cdots p_{i_r i_1} > 1,$$

then, by continuity of the CFMM trading functions, a sufficiently small quantity of T_{i_1} can be traded around the cycle, returning a strictly larger quantity of T_{i_1} and thereby yielding an arbitrage profit. Hence

$$p_{i_1 i_2} p_{i_2 i_3} \cdots p_{i_r i_1} \leq 1.$$

Applying the same argument to the cycle traversed in the opposite direction gives the reciprocal inequality, so

$$p_{i_1 i_2} p_{i_2 i_3} \cdots p_{i_r i_1} = 1.$$

Assume first that the token graph is connected. Choose T_1 as a numéraire and set $V_1 := 1$. For any other token T_j, choose a path

$$T_1 = T_{i_0}, T_{i_1}, \ldots, T_{i_r} = T_j$$

and define

$$V_j = \prod_{\ell=0}^{r-1} p_{i_{\ell+1} i_\ell}.$$

This definition is independent of the chosen path. Indeed, if two paths from T_1 to T_j produced different values, then traversing one path forward and the other backward would form a cycle whose product of spot prices was not 1, a contradiction.

Finally, suppose a CFMM trades T_j, T_k. Take any path from T_1 to T_k and append the edge from T_k to T_j. By the definition of V_j,

$$V_j = p_{jk} V_k,$$

and therefore

$$p_{jk} = \frac{V_j}{V_k}.$$

Thus $V = (V_1, \ldots, V_m)$ is a consistent valuation.

If the token graph has multiple connected components, apply the above construction independently to each component, choosing an arbitrary token in each component as a numéraire and assigning it value 1. ◀

Multi-Currency AMMs for Decentralized FOREX Markets: Feasibility & Optimal Design

Reina Ke Xin Li
University of Toronto, Canada

Andreas Park
University of Toronto, Canada

Andreas Veneris
University of Toronto, Canada

Srisht Fateh Singh
University of Toronto, Canada

Abstract

Most currency pairs lack a direct liquid market, so international foreign exchange relies on routing transactions through a dominant vehicle currency. Multi-currency automated market makers (AMMs) offer an alternative by sharing liquidity across many currency pairs, facilitating direct cross-currency trade while exploiting liquidity consolidation. This paper studies a multi-currency pool design that minimizes trading cost. Under a constant-mean AMM architecture, equilibrium trading costs reflect the trade-off between reduced price impact from consolidated liquidity and increased impermanent loss from joint return risk. This work derives closed-form costs, characterizes optimal pool weights, and shows that the optimized multi-currency pool dominates the status quo over a range of market parameters. It then formulates the system-level problem of partitioning currencies into multi-currency pools, which is solved using a hierarchical agglomerative clustering algorithm. Empirically, using exchange rate and trade data for 43 currencies over 2008-2023, the algorithm runs in 1.6 seconds and produces pools with geographic and economic structure. Notably, this reduces realized costs by ~13% relative to the status quo of vehicle-currency routing, with gains stable through episodes of global financial stress.

2012 ACM Subject Classification Applied computing → Economics

Keywords and phrases automated market makers, decentralized finance, currency foreign exchange

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.13

Related Version *Full Version*: https://arxiv.org/abs/2607.26405 [33]

Supplementary Material *Software (Source Code)*: https://github.com/reinali07/Multi-Currency-AMMs-FOREX, archived at swh:1:dir:b0a4c7db3553cfe7a4686b8ec2695047b5f2b0bb

1 Introduction

Daily trading volume in foreign exchange markets more than doubled from $4.0 trillion in 2010 to $9.6 trillion in 2025 [7, 9]. Despite this growth, foreign exchange transfers follow a peculiar arrangement: most currency pairs do not trade directly with one another but route through the US dollar (USD). For instance, a firm in Australia importing goods from Sweden cannot exchange Australian dollars (AUD) for Swedish kronor (SEK) directly at a "tight" spread as direct AUD/SEK markets are thin. Instead, it must first convert the AUD to USD and then the USD to SEK, thus paying the transaction costs for both trades. The work of [10] estimates that, in 2022, direct trades between the AUD and the SEK were virtually non-existent. More broadly, 13% of foreign exchange trading volume involving the USD is generated by routing trades between non-dollar currencies through the USD [47]. This is a significant volume given that USD-involved foreign exchange transactions as a whole account for over 85% of the global foreign exchange volume.

© Reina Ke Xin Li, Andreas Park, Andreas Veneris, and Srisht Fateh Singh;
licensed under Creative Commons License CC-BY 4.0

This dependence on the USD as a *vehicle currency* persists as an emergent feature of the international monetary system. After World War II, the stable US economy with its dollar attached to the gold standard made it an attractive peg codified through the 1944 Bretton Woods Agreement. This allowed the USD to play an important role as a vehicle currency for international foreign exchange. Even after the end of the Bretton Woods era in 1971 with the Nixon shock that canceled the convertibility of USD to gold, the USD's dominance as a vehicle currency has been sustained through various economic forces. Among other reasons, when bilateral trade between two currencies is moderate or low, neither side can sustain a deep enough market to offer competitive spreads, so traders gravitate toward an intermediate currency with high liquidity. Early work by [31], in fact, makes the case that concentrating trade in a vehicle currency is more efficient than trading with multiple pairs.

Yet, the political landscape is shifting in ways that make the architecture of international payments a pressing policy question. The use of dollar-based payments systems as a political instrument through sanctions and freezing of sovereign reserves has prompted countries to seek alternatives [22, 46, 39, 19, 51]. New currency swap agreements between central banks have been negotiated and payment systems outside the SWIFT network are under active development [48, 12, 15, 23, 43]. Most importantly, technological advances and innovations in blockchain-based trading technology raise the question of whether novel currency exchange arrangements can reduce reliance on a vehicle currency but also overall transaction costs.

This paper posits that Decentralized Finance (DeFi) offers a compelling alternative. Specifically, an *automated market maker* (AMM) is a decentralized exchange protocol that holds pooled reserves of assets that are traded against at prices governed by a mathematical *invariant*. The Balancer protocol, in particular, offers a *multi-asset AMM* [40]. Such a protocol could be the backbone for direct markets between various currency pairs, while still allowing intermediation through a vehicle currency when necessary. This practice has the potential to reduce overall trading costs relative to the status quo system of routing all trades through a vehicle currency by broadening support for *direct* cross pair trading.

The BIS's Project Mariana demonstrates the technical feasibility of AMM-based trading and settlement for foreign exchange trades [8]. However, whether a multi-currency AMM arrangement can achieve economic improvements on the status quo depends on a fundamental trade-off. Liquidity providers, *i.e.,* investors who deposit funds into an AMM pool, are exposed to *impermanent loss*: when exchange rates move, arbitrageurs profit against the pool. This is especially so for pools of heterogeneous assets, each with their own price movements and risk. An AMM pool planner must design fees, exposure weights, and currency inclusion to balance this risk against the liquidity benefits of consolidation. At the same time, pooling currencies with volatile or negatively correlated returns amplifies impermanent loss and raise the fee(s) required to sustain the pool, undermining the gains from deeper liquidity.

Resolving these trade-offs is the central contribution of this paper. *First*, we extend to multi-currency AMMs a mathematical framework in which equilibrium trading costs – the costs under optimally designed pool parameters assuming competitive liquidity provision – are a function of trading volumes, exchange rate covariances, and pool composition. *Second*, we show the market conditions under which an optimally designed multi-currency AMM dominates the status quo and study optimal pool weights in closed form for these conditions. We further develop a tractable approximation for the general case. *Finally*, we examine the system-level problem of which currencies to pool together using a *hierarchical agglomerative clustering* (HAC) algorithm on a correlation-based distance metric. Empirically, applying the framework to 43 non-pegged currencies over 2008–2023, the HAC algorithm runs in 1.6 seconds and produces pools that demonstrate meaningful *regional* and *economic* structure. The resulting architecture reduces realized aggregate costs by ~13% – USD ~2.5 billion – relative to the status quo, with gains remaining stable through episodes of financial stress.

The paper proceeds as follows. Section 2 introduces the general AMM framework. Section 3 contextualizes this setup. Section 4 studies benchmark exchange arrangements. Section 5 analyzes equilibrium costs of multi-currency AMM pools. Section 6 derives analytical weight optimization results for parametrized symmetric currencies. Section 7 studies the general weight optimization problem and develops a closed-form approximation. Section 8 formulates the currency partitioning problem, describes the HAC algorithm, and presents an empirical evaluation. Section 9 reviews the literature. Section 10 concludes and discusses further directions.

2 Preliminaries: General AMM Framework

Consider an AMM for $N+1$ currencies indexed by $i \in \{0, 1, \ldots, N\}$, funded by *liquidity providers* who deposit reserves $R_i \geq 0$ of each currency. The AMM is governed by an *invariant function* of the reserves that implicitly determines the prices at which the AMM trades. When a trader exchanges asset i for asset j by submitting $\Delta_i > 0$ units, the outgoing quantity Δ_j is uniquely determined by the requirement that the invariant is preserved.

Throughout, we use the *constant mean* invariant introduced by Balancer [40], $\prod_{i=0}^{N} R_i^{w_i} = k$, for a fixed weight vector $(w_0, \ldots, w_N)$ with $w_i \geq 0$ and $\sum_i w_i = 1$. The weight w_i governs pool exposure to currency i: at fundamental prices, currency i accounts for fraction w_i of total pool value. Thus, the 2-asset constant-product AMM ($R_0 R_1 = k$) is the special case used by Uniswap V2, among other popular AMMs [1].

2.1 Prices, Fees, and Trading Costs

The AMM-implied marginal price of asset i in terms of asset j is, for the constant-mean AMM, $p_{ij} = (w_i/w_j)(R_j/R_i)$ [40]. Competitive arbitrageurs continuously restore the pool's marginal prices to the exogenous exchange rates S_i (with numeraire $S_0 \equiv 1$), so that $p_{ij} = S_i/S_j$ in equilibrium. For the constant-mean AMM this requires $S_i R_i/w_i = S_j R_j/w_j$ for all pairs.

A finite trade of size Δ in numeraire moves the reserve vector along the invariant surface, causing the effective price $S^{\text{eff}}(\Delta)$ paid by the trader to deviate from the pre-trade spot price S. The *price impact* at spot price S is

$$\Pi(\Delta) = \frac{S^{\text{eff}}(\Delta) - S}{S}, \tag{1}$$

which is strictly positive and increasing in Δ for the constant-mean invariant. Traders additionally pay an explicit fee $f \geq 0$ proportional to transaction value,[1] so the total *unit trading cost* is $c(\Delta, f) = \Pi(\Delta) + f$. Denoting by c_{ij} the unit cost for pair i, j and Q_{ij} the total trading volume on pair (i, j) (assuming a fixed trade size Δ and balanced flows in either direction), the aggregate cost for pair (i, j) over the period is approximately $c_{ij}(\Delta, f) = \big(\Pi(\Delta) + f\big)Q_{ij}$.

2.2 Impermanent Loss

The pool's value in the numeraire is $V = \sum_i S_i R_i$, held collectively by liquidity providers. When external prices change, arbitrageurs trade against the pool to restore price alignment. Arbitrageurs are incentivized by the favourable terms offered by the AMM: the AMM sells the

[1] For simplicity, we assume that fees are paid to liquidity providers directly, not added back to the pool.

asset below its new fundamental value or buys above it. The profit earned by the arbitrageur is at the expense of the pool's liquidity providers. This gives rise to *impermanent loss* (IPL). Consider an initial deposit $V(0)$ and a period $[0, T]$ in which currency i's has gross return $x_i = \frac{S_i(T)}{S_i(0)}$ (with $x_0 \equiv 1$). After arbitrageurs restore price alignment, the pool value is $V(T)$, while the counterfactual hold value of the initial deposit is $V_{\text{hold}}(T) = \sum_i x_i S_i(0) R_i(0)$. The IPL is defined as[2]

$$\text{IPL} = \frac{V(T) - V_{\text{hold}}(T)}{V(0)} \leq 0. \tag{2}$$

The loss is described as "impermanent" in DeFi settings because it is unrealized if prices revert before the provider withdraws. In the context of this work, the loss is due to changes in the fundamental.

2.3 Equilibrium Liquidity Supply and Optimal Fees

We follow the framework of [36] and model equilibrium competitive liquidity provision: risk-neutral providers deposit at the start of a period and withdraw at the end, earning fee revenue while bearing impermanent loss. To simplify the exposition, we assume that there is sufficient exogenous, uninformed trading relative to the order flow of informed arbitrageurs so that fees accrue on noise volume while IPL stems from price changes driven by informed flow. Competitive liquidity providers contribute capital until they *break-even*:

$$E[|\text{IPL}|] \cdot V(0) = E[Q] \cdot f, \qquad Q = \sum_{i<j} Q_{ij}. \tag{3}$$

For a given fee rate f and return and volume distributions, (3) pins down the equilibrium pool depth $V(0)$ and hence the price impact and unit trading cost $c(\Delta, f)$. The pool designer chooses the fee that minimizes unit trading costs, $f^* = \arg\min_f c(\Delta, f)$; this fee maximizes trader welfare and ensures that no competing pool can charge better rates.

3 Overview

At a high level, our goal is to design an optimal foreign exchange mechanism and ask: *do multi-currency pools lower costs* or *does the additional risk from holding many currencies outweigh the benefits of pooling?* Our benchmarks are bilateral AMMs as proxies for the cost of vehicle routing. Although at first sight these AMMs look different to traditional markets, they are economically equivalent to a limit order book and as [36] show, bilateral AMMs are a low cost benchmark for existing markets. We use the framework in Section 2 to isolate the key forces determining costs. The AMM invariant determines how trades move pool prices, giving rise to price impact – a cost that diminishes with increased liquidity. Liquidity providers supply this liquidity, but exchange rate volatility exposes them to impermanent loss, so they require trading fees as compensation. A higher fee attracts more liquidity but is charged to the trader. A pool planner therefore sets a fee that balances these forces, yielding optimal equilibrium pool depth and minimized trading cost.

The following builds on this framework by studying how pools should be designed once these forces are in place, moving from simple benchmarks to richer design problems. Constant-mean pools introduce weights as an additional design parameter that control the pool's

[2] IPL is also sometimes defined in other literature as $(V(T) - V_{\text{hold}}(T))/V_{\text{hold}}(T)$.

exposure to each currency's risk and volume. At the system level, design also involves deciding which currencies to pool together. We ultimately study the pool weights and membership structures that minimize equilibrium trading costs across different market environments.

4 Benchmark Currency Exchange Architectures

We begin by specializing to a minimal three-currency environment consisting of currencies $\{0, 1, 2\}$, with currency 0 as the numeraire, to establish benchmarks, and then generalize the architectures to $N+1$ currencies. Following the work in [35], we consider three arrangements. In the *status quo*, trades between currencies 1 and 2 are routed through the vehicle currency 0 via two separate bilateral constant-product pools; each $(0, i)$ pool handles total volume $Q_i = \sum_{j \neq i} Q_{ij}$, and a *cross-pair* trader (trading currencies 1 and 2) bears costs in both. In the *dedicated bilateral pools* architecture, each pair has its own pool. In the *equal-weighted constant mean pool*, all three currencies are held in a constant mean pool with $w_0 = w_1 = w_2 = 1/3$. By [35], the fee-optimized equilibrium unit costs for trades between $i \leftrightarrow j$ of size Δ in numeraire under the three arrangements are, respectively,

$$c_{ij}^{\mathrm{SQ}} = \sqrt{\frac{\sigma_i^2\,\Delta}{E[Q_i]}} + \sqrt{\frac{\sigma_j^2\,\Delta}{E[Q_j]}}, \qquad c_{ij}^{\mathrm{BP}} = \sqrt{\frac{\sigma_{ij}^2\,\Delta}{E[Q_{ij}]}}, \qquad c_{ij}^{\mathrm{SYM}} = \sqrt{\frac{2H\Delta}{3E[Q]}}$$

where $\sigma_i^2 = \mathrm{Var}(x_i)$, $\sigma_{ij}^2 = \mathrm{Var}(x_i - x_j)$ (we call σ the relative volatility of i and j), $H = \sum_{i<j} \sigma_{ij}^2$ and $Q = \sum_{i<j} Q_{ij}$.[3]

In each case, costs scale with volatility and shrink with volume. The status quo charges two costs for a $i \leftrightarrow j$ trade, each discounted by its own currency's total volume. The bilateral pool charges a single cost but only gets liquidity from the i, j pair. In the equal-weighted AMM all pairs share the same cost discounted by aggregate volume $E[Q]$, so more actively traded pairs effectively subsidize thinner-traded ones.

The cost c^{BP} generalizes straightforwardly to $N+1$ currencies, with aggregate cost $c^{\mathrm{BP}} = \sum_{i<j} \sqrt{\sigma_{ij}^2 E[Q_{ij}]\Delta}$. We extend the results for c^{SQ} and c^{SYM} to $N+1$ currencies in the following propositions, both of which are proved in Appendix A.1.

▶ **Proposition 4.1** (Status quo trading cost). *The expected aggregate trading cost in the status quo for $N+1$ currencies routed through the vehicle currency 0 is*

$$c^{SQ} = \sum_i \sqrt{\sigma_i^2 E[Q_i]\Delta}$$

▶ **Proposition 4.2** (Equal weights trading cost). *The expected aggregate trading cost in an equal-weighted $N+1$ currency constant-mean pool is*

$$c^{SYM} = \sqrt{\frac{2\Delta H E[Q]}{N+1}}$$

In each case, the aggregate costs scale with exchange rate volatilities, but c^{SQ} charges volatility per currency, as each currency routes trades through the vehicle separately, while c^{BP} and c^{SYM} charge per pair, reflecting the direct cross-pair trading. The dominant pool depends on the specific volatility and volume structure, which is explored further in Section 6.

[3] [35] derives the costs in expectation using second-order approximations for IPL under the assumption of small, zero-mean excess returns $\epsilon_i = x_i - 1 \approx 0$. We use the same simplifying assumption throughout.

5 Equilibrium in Constant-Mean AMMs

Equal weights are not optimal in general: a currency with high individual volatility inflicts more impermanent loss per weight, while one with high trading volume generates more fee revenue per weight. Introducing non-uniform weights allows the pool designer to tilt exposure away from high-volatility, low-volume currencies, reducing equilibrium trading costs.

The break-even framework of Section 2.3, developed for two-asset constant-product pools in [36], extends to general constant-mean pools. The lemmas below derive the corresponding price impact and impermanent loss expressions; applied to the break-even condition (3) and cost function $c(\Delta, f)$, they yield the equilibrium trading cost in Proposition 5.3. All proofs for this section are found in Appendix A.2.

▶ **Lemma 5.1** (Price impact). *In the general constant-mean AMM, the price impact of a trade of size Δ in numeraire between currencies i and j is approximately*

$$\Pi(\Delta) \approx \frac{w_i + w_j}{w_i w_j} \cdot \frac{\Delta}{2V}$$

▶ **Lemma 5.2** (Impermanent loss). *For gross returns x_i with $x_i \approx 1$ and $E[x_i] = 1$, the expected impermanent loss of the general constant-mean pool satisfies*

$$E[|IPL|] \approx \frac{H_w}{2}, \qquad H_w = \sum_{i<j} w_i w_j \sigma_{ij}^2$$

Each currency pair (i, j) contributes to H_w in proportion to the product of the two currencies' weights and the variance of their relative returns. Concentrating weight on a pair with high relative volatility σ_{ij}^2 increases expected impermanent loss and therefore the fee needed to sustain liquidity provision in equilibrium. Together with Lemma 5.1, this gives us the following proposition.

▶ **Proposition 5.3** (Equilibrium trading cost). *Under the break-even condition (3) and Lemmas 5.1–5.2, the aggregate equilibrium trading cost is minimized at*

$$f^* = \sqrt{\frac{\Delta H_w}{4(E[Q])^2} \sum_i \frac{E[Q_i]}{w_i}}, \qquad c^* = \sqrt{\Delta H_w \sum_i \frac{E[Q_i]}{w_i}} \tag{4}$$

The cost factors cleanly into two terms: H_w, which is proportional to expected impermanent loss, and $\sum_i E[Q_i]/w_i$, which is proportional to the volume-weighted price impact per unit of volume. The pool designer's goal is to choose weights to minimize this cost.

6 Optimal Pools Under Symmetric Multi-Currency Parametrizations

We begin by deriving closed-form optimal weights and cost comparisons for special symmetric cases, where the general problem of Section 5 reduces to a scalar optimization. The three-currency case extends the analysis of [35] by characterizing optimal weights, while the $N + 1$-currency case is, to our knowledge, new.

6.1 Three Currency Case

We first study a parametrization that captures the structure of a three-currency system organized around a vehicle currency. Let currency 0 be the vehicle. Suppose that the $(0, 1)$ and $(0, 2)$ pairs have equal trading volumes and equal relative volatilities, while the

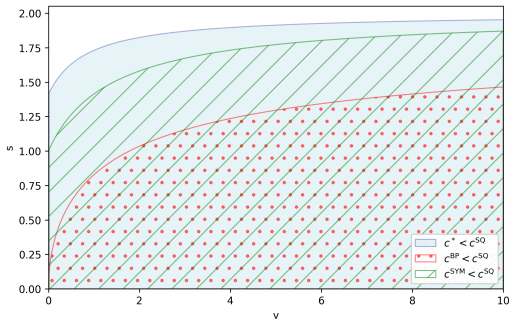


Figure 1 s vs. v regions for optimality over the status quo under the three-currency (v, s) parametrization.

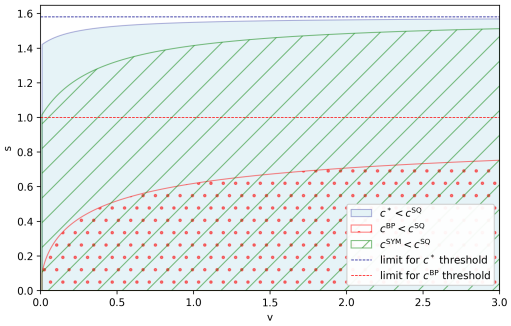


Figure 2 s vs. v regions for optimality over the status quo under the (v, s) parametrization for $N = 5$ currencies.

$(1, 2)$ cross pair trades at a smaller volume with higher relative volatility. That is, let $E[Q_{01}] = E[Q_{02}] = q$ and $\sigma_{01}^2 = \sigma_{02}^2 = \sigma^2$. Write $E[Q_{12}] = vq$ for some $v > 0$ and $\sigma_{12}^2 = s^2\sigma^2$ for $s \geq 0$.[4] Values $s > \sqrt{2}$ correspond to negatively correlated returns for currencies 1 and 2, while $s < \sqrt{2}$ corresponds to positively correlated returns. The case $v < 1$ represents the typical vehicle-currency scenario in which direct cross-currency trading is thin; $v > 1$ could represent two highly liquid currencies and one smaller currency trading against the two.

Under this parametrization, currencies 1 and 2 are symmetric, so the cost-minimizing weights satisfy $w_1 = w_2 = (1 - w_0)/2$, and the problem reduces to optimizing over the scalar $w_0 \in (0, 1)$. The result is summarized in the following proposition which is proved in Appendix A.3.

▶ **Proposition 6.1** (Optimal weight and cost for vehicle and 2 symmetric currencies)**.** *Under the symmetric (v, s) parametrization for 3 currencies, if $s^2 < 2(1 + 2v)/(1 + v)$, the expected aggregate trading cost of Equation (4) has unique interior minimizer at $w_1^* = w_2^* = \frac{1-w_0^*}{2}$ and*

$$w_0^* = \frac{s}{\sqrt{(1 + 2v)(4 - s^2)}}, \qquad c^* = \sqrt{q\sigma^2\Delta}\sqrt{s^2v + 2 + s\sqrt{(4 - s^2)(1 + 2v)}}$$

If $s^2 \geq 2(1 + 2v)/(1 + v)$, no interior minimizer exists; the infimum over $w_1, w_2 > 0$ is obtained as $w_0 \to 1$.

The optimal vehicle-currency weight w_0^* increases with s: when the $(1, 2)$ pair is volatile, the pool tilts toward the vehicle to lower IPL generated by the cross pair. Conversely, w_0^* decreases with v (the relative volume of the non-vehicle pair): when the $(1, 2)$ pair has high volume, the pool should provide more depth to the cross pair and reduce its price impact.

We plot the regions in which c^*, c^{SYM}, and c^{BP} are optimal over the status quo cost, c^{SQ} in Figure 1. The precise expressions for the thresholds are found in Appendix B.

6.2 Extension to More Currencies

The above can be generalized to $N \geq 2$ symmetric currencies organized around a vehicle currency. Again, let currency 0 be the vehicle and suppose that $(0, i)$ pairs have equal trading volumes and volatilities, q and σ, while each pair (i, j) with $i, j \neq 0$ trading at volume vq with

[4] Since $\sigma_{12}^2 = \mathrm{Var}(\epsilon_1 - \epsilon_2) = \sigma_{01}^2 + \sigma_{02}^2 - 2\,\mathrm{Cov}(\epsilon_1, \epsilon_2)$, the Cauchy–Schwarz inequality implies $0 \leq \sigma_{12}^2 \leq 4\sigma^2$ and $s \in [0, 2]$.

volatility of returns $s\sigma$. Under this parametrization, the weights satisfy $w_i = (1-w_0)/N$, and the problem reduces to optimizing over $w_0 \in [0,1]$, summarized in the following proposition and proved in Appendix A.3.

▶ **Proposition 6.2** (Optimum for vehicle and N symmetric currencies). *Under the symmetric (v,s) parametrization for $N+1$ currencies, if $s^2 < 2(1+Nv)/(1+(N-1)v)$, the expected aggregate trading cost from Equation (4) has unique interior minimizer at $w_i^* = (1-w_0^*)/N$ for all $i > 0$ and*

$$w_0^* = \frac{s}{\sqrt{(1+Nv)(2N-(N-1)s^2)}}$$

If $s^2 \geq 2(1+Nv)/(1+(N-1)v)$, no interior minimizer exists; the infimum over $w_1, w_2 > 0$ is obtained as $w_0 \to 1$.

The expression for the optimal aggregate expected cost c^* can be found as Equation (7) in Appendix A.3. The optimal weights of the three-currency case are recovered by setting $N = 2$, and similarly, we observe the increasing optimal vehicle weight with s.

We plot, for the $N+1$ currencies case, the regions for which c^*, c^{SYM}, and c^{BP} are optimal over the status quo cost, c^{SQ} in Figures 2. The precise expressions for the thresholds are found in the Appendix B.

7 General Constant Mean AMM Weight Optimization

For a general distribution of volumes and volatilities of a given pool of currencies, the pool designer solves for the weights $w_i > 0$ summing to 1 and minimizing $\mathcal{F}(w) = H_w \sum_i E[Q_i]/w_i$. Closed-form solutions are not available in general, but the system can be solved numerically via constrained optimization methods.

A tractable heuristic approximation is obtained by noting that, in the objective $\mathcal{F}(w) = H_w \cdot \sum_i E[Q_i]/w_i$, the first factor captures the impermanent loss exposure of the pool and the second captures the volume-weighted price impact. The two factors interact through the weights, with the first biasing towards currencies with small relative volatilities and the second biasing towards currencies with high trading volumes. A useful simplification captures these forces and decouples the weights: we approximate the minimization objective by the heuristic $\tilde{\mathcal{F}}(w) = \sum_i \frac{E[Q_i]}{w_i H_i}$ where $H_i = \sum_{j \neq i} \sigma_{ij}^2$. The heuristic-approximated optimum is presented in the following proposition, proved in Appendix A.4.

▶ **Proposition 7.1** (Approximate optimal weights). *When $H_i > 0$ for all i and $E[Q_i] > 0$ for some i, the minimizer of $\tilde{\mathcal{F}}(w) = \sum_i \frac{E[Q_i]}{w_i H_i}$ subject to $\sum_i w_i = 1$, $w_i > 0$ $\forall i$, is*

$$w_i^* = \frac{1}{D}\sqrt{\frac{E[Q_i]}{H_i}}, \qquad \forall\, i = 0, \ldots, N, \qquad \text{where } D = \sum_j \sqrt{E[Q_j]/H_j},$$

This heuristic rule assigns each currency a weight proportional to the square root of its expected aggregate trading volume $E[Q_i]$ and the inverse of the sum of its relative variances. The square-root scaling arises from the diminishing effects of concentrating weight on a single currency in reducing price impact while leaving the impermanent loss exposure unimproved. Hence, the optimum disperses weight broadly, with the degree of dispersion governed by the square-root of relative volumes and volatilities.

7.1 Numerical Validation

We evaluate the quality of the approximation $\min \tilde{\mathcal{F}}$ by comparing the equilibrium trading cost it achieves against the cost attained by the numerical minimizer of $\mathcal{F}$, across a wide range of pool sizes and randomly drawn market environments.

Methodology. For each pool of $N \in \{2, 3, \ldots, 20\}$ non-vehicle currencies (pooled together with the vehicle), we conduct 500 independently drawn trials, proceeding as follows. To generate covariance matrices, we use a synthetic 4-factor model [13], where returns are correlated through shared factors, with loadings $B \in \mathbb{R}^{N\times 4}$, a matrix of standard normal entries. The covariance matrix K is constructed as $K = BB^T + \Psi$, where Ψ is a diagonal matrix with entries uniformly chosen from $[0, 1]$, representing idiosyncratic risk. An additional column and row of zeros is added for the 0-return vehicle currency. The pairwise relative variance matrix $\sigma^2 \in \mathbb{R}^{(N+1)\times(N+1)}$ is then computed with entries $\sigma^2_{ij} = K_{ii} + K_{jj} - 2K_{ij}$. Expected trading volumes are drawn as $E[Q_{ij}] = q_i q_j$ with $q_i \sim \text{LogNormal}(0, 1)$, yielding a symmetric volume matrix with log-normally distributed marginals. $E[Q_i]$ for all i and $E[Q]$ are computed from this matrix. We compute three weight vectors: $w^{\mathcal{F}}$, the numerical minimizer of $\mathcal{F}(w) = H_w \sum_i E[Q_i]/w_i$ using sequential quadratic programming (SLSQP) [30]; $w^{\tilde{\mathcal{F}}}_i = D^{-1}\sqrt{E[Q_i]/H_i}$, the approximation from Proposition 7.1; and $w^{\text{EQ}}_i = 1/(N+1)$, the equal weights vector. The aggregate expected cost of Equation (4) is evaluated for each weight vector with trade size $\Delta = 1$. We also compute the costs of the status quo vehicle routing and the dedicated bilateral pools architectures for comparison.

Results. Figures 3 and 4 report the mean aggregate expected cost, expressed as a percentage of total volume, and its coefficient of variation over the trials as a function of pool size N. The two methods achieve nearly identical costs for all N, with the gap remaining small and roughly constant as N grows from 3 to 20. This implies that ignoring the interaction between H_w and $\sum_i E[Q_i]/w_i$ when optimizing weights is negligible in practice. The coefficient of variation tracks closely across both methods for all N, indicating that variability reflects heterogeneity of the market environments rather than any degradation of the approximation. A pool designer can thus use the closed-form rule $w^*_i \propto \sqrt{E[Q_i]/H_i}$ in place of a much more computationally demanding numerical optimizer. This is practical for AMM weight optimization in computationally restricted applications such as in blockchains, or for efficiently estimating costs in the optimization algorithm described in the next section.

8 Currency Partitioning for System-Level Pool Design

In this section, we address the question of *which* currencies to pool together. Evidently, not every currency benefits from co-pooling: if the returns of two currencies have high relative variance σ^2_{ij}, their impermanent loss raises the fee, which may outweigh liquidity gains. We therefore introduce the *currency partitioning problem*: given N currencies and their pairwise volumes and return covariance structure, partition the currencies into multi-currency pools and unpooled singletons to minimize aggregate trading costs.

We fix currency 0 as the vehicle and consider multi-currency pools of the form $I = \{0\} \cup G$ where $G \subseteq \{1, \ldots, N\}$. Let $M = \{I_1, \ldots, I_K, O\}$ be a collection of such pools (that are disjoint except for the vehicle) and the set O of unpooled currencies. A trade between a non-vehicle i in a pool I and $j \notin I$ is executed in two legs: $i \leftrightarrow 0$ in the pool, and $0 \leftrightarrow j$ in the pool containing j or a bilateral pool with the vehicle if $j \in O$. Trades between unpooled currencies are also routed bilaterally through the vehicle.

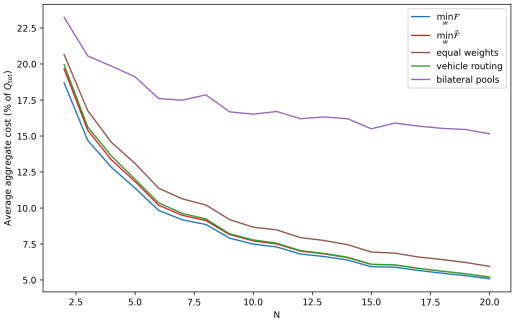


Figure 3 Mean equilibrium aggregate trading cost (as a percentage of total volume) achieved by the exact minimizer of $\mathcal{F}$ and the closed-form approximate minimizer of $\tilde{\mathcal{F}}$, averaged over 500 trials per pool size.

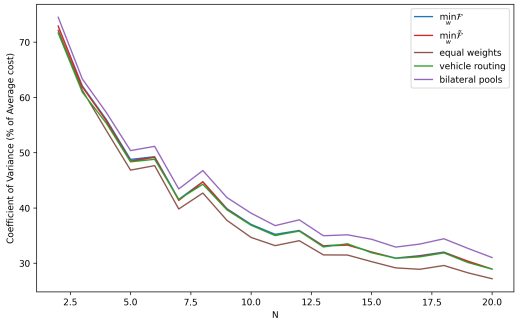


Figure 4 Coefficient of variation (standard deviation / mean) of the equilibrium aggregate cost across trials, as a function of pool size N, for the exact and approximate minimizers.

Let $c(M)$ denote the total expected system cost; the currency partitioning problem is to choose M minimizing $c(M)$. The expression for $c(M)$ is calculated in Appendix C.

8.1 Rolling Window Realized Cost

Our empirical evaluation separates the *ex-ante* and *ex-post* quantities. Liquidity providers set the pool depth V and fee f before observing the period's realized trading volume and price movements. They base these decisions on expectations formed from recent data.

We implement this separation as follows. For each quarter t, we form the *expected* covariance matrix using a 12-month rolling window of monthly returns ending at the start of quarter t, scaled to a quarterly frequency. We form the *expected* volume matrices using the same 12-month windows, scaled to quarterly averages. The pool planner uses these expected values to compute the break-even fee f^* via Proposition 5.3 and the optimal weights w^* numerically by minimizing $\mathcal{F}(w)$ at the beginning of each quarter t. Liquidity providers then decide $V = V^*$, the pool depth.

The *realized* cost in quarter t is then computed by applying the unit cost implied by the ex-ante pool parameters (w^*, f^*, V^*) to the *realized* trade volume for quarter t, which is then summed for every quarter over the evaluation period. The expression for the realized cost is also found in Appendix C. This specification captures an additional risk faced by liquidity providers: the fee, weights, and depth are fixed at the start of the quarter, but the realized cost depends on how much volume actually flows through the pool.

8.2 Hierarchical Agglomerative Clustering Algorithm

The number of partitions of N items grows super-exponentially, making exhaustive search infeasible. Instead, we exploit the cost function structure to inform heuristic methods. Since a currency inside a multi-asset pool executes all its trades inside the pool, the main determinant of pooling efficiency is the pairwise relative variance of currencies within a pool. Currencies with highly correlated returns have lower σ_{ij}^2 and thus generate lower impermanent loss. Hence, we use a *hierarchical agglomerative clustering* (HAC) algorithm on the correlations [37].

First, we transform correlations into distances $d_{ij} = \sqrt{0.5(1-\rho_{ij})} \in [0,1]$ so that highly correlated currencies are close together. We then construct a partition over the set of currencies as follows. Starting from singletons (equivalent to the status quo), we iteratively

merge the two clusters with smallest distance. Here, the distance between two clusters A and B is defined as $d_{AB} = \frac{1}{|A||B|}\sum_{i\in A, j\in B} d_{ij}$, the mean pairwise distance between the currencies across the two clusters. We require $d_{AB} \leq \tau$, a distance threshold. If all clusters are farther than τ from each other, the procedure ends and returns the current partition.

We evaluate the partition produced by a range of thresholds $\tau \in [0, 1]$ against the expected aggregate cost $c(M)$ (evaluated using approximate weights from Proposition 7.1 for the sake of efficiency) and select the threshold τ and resulting partition that minimizes this cost.

8.3 Empirical Data

Exchange rates come from the IMF Exchange Rates dataset (monthly period-average *US Dollar per domestic currency* indicator) [28], from which we compute monthly returns and relative variances. Trade volumes are proxied by the USD value (in millions) of bilateral goods exports between the countries issuing the currency. This approximates the real demand for cross-border currency exchange. This data is obtained from the IMF International Trade in Goods dataset (monthly *Export of Goods, FOB, US dollar* indicator) [29].[5] Currency partitioning with the HAC algorithm uses 2002–2007 monthly exchange rates and trade data to calculate correlations, relative variances, and trade volumes over the whole period (scaled to a quarterly basis); out-of-sample 12-month quarterly rolling window relative variances and trade volumes are calculated for the evaluation period, 2008–2023. We choose these windows for several reasons: first, 2002 is the starting year as this is when the Euro was officially adopted; second, we avoid using data from the 2008 financial crisis in the pool selection, but do not avoid evaluating on this crisis period; third, 2023 is the end year as it is the last year for which IMF AREAER exchange rate regime classifications are available.

The last criteria is important as we restrict to non-pegged currencies under the IMF AREAER [27] – currencies classified as independently floating, managed floating, crawling band, crawling peg (pre-2008 classifications), other managed, free floating, floating, crawl-like, crawling peg, or stabilized (post-2008 classifications) over 2002–2023. Pegged currencies are excluded because their near-zero volatility against the peg makes them misleading. For example, pooling USD/EUR/DKK/BGN, where the Danish krone (DKK) and the Bulgarian lev (BGN) are pegged to the Euro, results in a realized cost of $c(I) = 840$ USD millions, versus $c^{\text{SQ}}(I) = 1{,}400$ USD millions, a 40% cost saving over 2008–2023. However, this is a flawed analysis – central banks managing pegged currencies tend to provide guaranteed exchange rates against the peg, rather than using market supply and demand. We further exclude currencies with observed zero returns against the USD for three consecutive months, or that are described as pegged by the IMF AREAER,[6] as well as the Haitian gourde (co-circulates with USD[7]) and the Somali shilling (limited regime data). The resulting 43 currencies are listed in Table 1, with the USD as vehicle (currency 0). For the remainder of this section, we fix $\Delta = 1$ USD million throughout.

8.4 Empirical Results

The HAC algorithm, implemented in Python 3.13 with Numpy 2.2.1, Pandas 2.2.3, and SciPy 1.14.1 runs in 1.6 seconds on a 16-core Intel Core Ultra 7 155H processor and 16 GB RAM. Table 2 shows the pool architecture selected by the HAC algorithm, and Table 3

[5] We use the Euro Area (Eurozone) in lieu of a country for the Euro.
[6] Afghani afghani, Cambodian riel, Dominican peso, Gambian dalasi, Georgian lari, Kyrgyz som, Laotian kip, Liberian dollar, Malagasy ariary, Moldovan leu, Myanmar kyat, and Papua New Guinean kina.
[7] IBAN lists USD as a currency of Haiti.

Table 1 Currencies with non-pegged exchange rates for all years in 2002–2023 according to IMF AREAER classification (43 currencies).

Currency code
ALL, AMD, AUD, BIF, BRL, CAD, CDF, CHF, CLP, COP, CZK, DZD, EUR, GBP, GHS, GTQ, IDR, INR, ISK, JMD, JPY, KES, KRW, MUR, MXN, MZN, NIO, NOK, NZD, PEN, PHP, PLN, PYG, RSD, SEK, SGD, THB, TRY, TZS, UGX, UYU, ZAR, ZMW

Table 2 Pool architecture from HAC algorithm. USD included implicitly in all pools.

Pool	Currencies	Description
1	ALL, CHF, CZK, EUR, GBP, JPY, NOK, PLN, RSD, SEK	High liquidity + Europe
2	AUD, ISK, NZD	Carry trade targets
3	COP, TRY	Macro risk
4	KRW, SGD, THB	East & South-East Asian

reports out-of-sample realized aggregate costs over 2008–2023. The optimized architecture, with distance threshold $\tau = 0.46$, places 18 of the 43 currencies into four multi-asset pools, reducing realized aggregate system costs by approximately 13% (roughly USD 2.5 billion) relative to bilateral USD routing. Some quarters exhibit realized savings exceeding 15-20%, and performance persists and remains stable even through the 2008 financial crisis and the 2020 COVID-19 recession, indicating that the selected pool structures are persistent rather than artifacts of the calibration window.[8]

Patterns emerge from the resulting pools. Pool 1 includes currencies with high foreign exchange turnover often considered "safe-havens" (EUR, JPY, GBP, CHF) and other European currencies, reflecting the institutional and trade integration of the European economic area and its close partners [9, 45]. Pool 2 groups commodity-linked AUD and NZD with ISK, all three of which are carry-trade target currencies [14, 16]. Pool 3 pairs the Colombian peso and Turkish lira, both emerging markets exposed to high political risks [50]. Pool 4 clusters KRW, SGD, and THB, East and South-East Asian currencies of manufacturing-heavy economies and regional trade integration [49]. This confirms that the trade-off at the heart of this paper – liquidity consolidation versus impermanent loss – can be resolved in practice by grouping currencies that co-move, capturing the benefits of shared liquidity without amplifying exchange rate risk. The emergent geographical and economic structures are a direct consequence of this co-movement pooling. Figure 6 visualizes the resulting pools in a heatmap of pairwise correlations, ordered by pool assignment. The heatmap shows the three high-correlation blocks along the diagonal corresponding the Pools 1-4, with the large Pool 1 block most pronounced on the top-right. Overall, the structured results and 13% empirical cost reduction serve to validate that well-designed multi-currency pools can meaningfully lower foreign exchange trading costs relative to bilateral vehicle-currency routing.

[8] The HAC algorithm is not theoretically cost-optimizing, but we compare it with a simple greedy algorithm in Appendix D to benchmark the HAC algorithm's performance.

Table 3 Out-of-sample realized aggregate trading costs, 2008–2023 in millions USD.

Architecture	$c(M)$	c^{SQ}	Savings	Savings (%)
HAC algorithm pool selection	16,450	18,930	2,500	13%

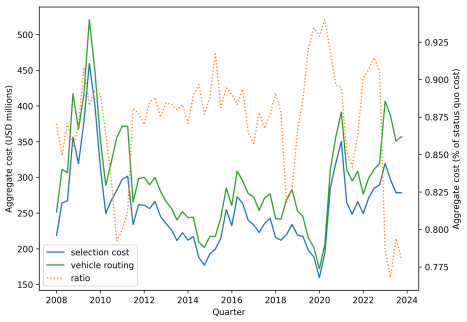


Figure 5 Quarterly realized cost of Table 2 architecture as a percentage of USD-routing cost.

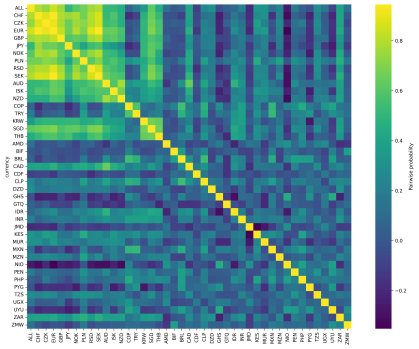


Figure 6 Heatmap of correlations with currencies ordered by HAC-optimal pool assignment.

9 Related Work

Developments on AMMs have accelerated since the constant-product invariant was introduced by Uniswap V2 [1]. Subsequent protocols explored alternative invariants and liquidity mechanisms: Balancer generalized to the constant mean invariant, Curve developed a hybrid invariant optimized for low-volatility assets, and Uniswap V3 introduced concentrated liquidity to improve capital efficiency [40, 20, 2]. Our work builds on the constant-mean framework and studies AMMs through the lens of pool formation and weight optimization.

Meanwhile, theoretical literature studies the economics of AMMs. The authors of [3, 4] show how arbitrage aligns prices with external markets and concave payoff functions can be replicated by suitable invariants, while [5] studies optimal routing across multi-asset AMMs. Another major focus is liquidity provider losses induced by arbitrage and price volatility, quantified by [44, 42]. Empirical and strategic analysis examine realized liquidity provider outcomes and adoption limits [26, 41, 11], optimal and equilibrium liquidity provision [52, 32, 6], and endogenous fee setting [21, 36] in 2-asset constant product pools. Closer to our setting, the work of [34, 24] compare FX efficiency in AMMs under different invariants and levels of liquidity fragmentation. We add to this literature by embedding closed-form expressions for AMM prices and impermanent loss of multi-asset constant-mean pools into the equilibrium model of competitive liquidity provision from [36] in a foreign exchange setting.

Our setting also relates to the literature on vehicle currencies in foreign exchange markets. Consistent with our framework, [31] show that vehicle currencies emerge from transaction cost minimization and economies of scale concentrating liquidity along vehicle currency pairs, while [25] emphasize the persistence of vehicles due to trade and industry structure.

Finally, our problem is related to portfolio optimization. Building on the mean–variance optimization framework of [38], subsequent work incorporates metaheuristic methods such as genetic and evolutionary algorithms to address nonlinearities [18, 17, 53]. Our problem, however, jointly determines pool membership and optimal weights with a highly nonlinear objective where weights also appear in the denominator. Moreover, while portfolio optimization seeks to diversify away covariance risk, our setting benefits from highly correlated assets.

10 Conclusion and Future Work

This paper studies the design of multi-currency AMM architectures for foreign exchange systems. In our framework, the equilibrium trading cost depends jointly on trading volumes, exchange-rate volatilities, and pool composition. We show that under this framework, pooling multiple currencies in a constant weighted mean AMM can reduce aggregate trading costs compared to the status quo – bilateral routing through a vehicle currency. The gains arise because multi-currency pools enable direct trades while still concentrating liquidity across many trading pairs. While multi-asset pools suffer from increased risk exposure, pooling assets with correlated exchange rates and appropriately weighing exposures limits the additional impermanent loss. We derive analytical results characterizing optimal weights in symmetric environments and a tractable approximation for general weight optimization, and formulate the system-level problem of partitioning currencies into pools.

To solve the partitioning problem, we employ a hierarchical agglomerative clustering heuristic algorithm that exploits the correlation structure of exchange-rate returns to efficiently group currencies into structurally similar pools. Empirically, using exchange-rate and international trade data for 43 non-pegged currencies over 2008–2023, we find that the resulting architecture reduces realized aggregate trading costs by ~13% relative to bilateral USD routing, with improvements remaining stable through major episodes of financial stress.

Several directions remain for future research. First, relaxing the assumptions we make by allowing persistent order imbalances, richer exchange-rate dynamics, stochastic trade-size distributions, or endogenous trading demand may materially affect the optimal architecture and equilibrium costs. Second, future work could seek analytical characterizations of the general weight optimization solution or derive approximation schemes with formal error bounds. Third, the present framework treats pool membership as fixed after the initial pool setup. The design of a dynamic multi-currency AMM that supports rebalancing mechanisms allowing currencies to migrate between pools as market conditions change could improve the flexibility and robustness of the multi-currency architecture, especially for currencies issued by countries with volatile economic policies. The current work uses historical correlations for pool formation, but the same theory extends to dynamic pools formed using time-varying volatilities. Finally, the robustness of the agglomerative clustering algorithm should be studied over a variety of training data regimes and sets of currencies before it can be proposed as a reliable method for currency pooling, especially if deployed in an algorithmically rebalancing multi-currency AMM. Indeed, there may exist algorithms that are more efficient or robust across different market environments. Alternative approaches based on graph partitioning, metaheuristics, convex relaxations, or machine learning may yield improved solutions or stronger theoretical guarantees. These methods may also be able to deal with more complex pooling architectures, such as non-disjoint pools or pools without a common vehicle currency.

References

1 Hayden Adams, Noah Zinsmeister, and Dan Robinson. Uniswap v2 Core. Technical report, Uniswap, March 2020. URL: `https://uniswap.org/whitepaper.pdf`.

2 Hayden Adams, Noah Zinsmeister, Moody Salem, River Keefer, and Dan Robinson. Uniswap v3 Core. Technical report, Uniswap, March 2021. URL: `https://uniswap.org/whitepaper-v3.pdf`.

3 Guillermo Angeris and Tarun Chitra. Improved price oracles: Constant function market makers. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, AFT '20, pages 80–91, New York, NY, USA, 2020. Association for Computing Machinery. `doi:10.1145/3419614.3423251`.

4 Guillermo Angeris, Alex Evans, and Tarun Chitra. Replicating market makers. *Digital Finance*, 5(2):367–387, June 2023. doi:10.1007/s42521-023-00082-0.

5 Guillermo Angeris, Alex Evans, Tarun Chitra, and Stephen Boyd. Optimal routing for constant function market makers. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, EC '22, pages 115–128, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3490486.3538336.

6 Jun Aoyagi. Liquidity provision by automated market makers, October 2020. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3674178.

7 Bank for International Settlements. Triennial central bank survey: Report on global foreign exchange market activity in 2010. Technical report, Bank for International Settlements, Basel, Switzerland, December 2010. URL: https://bis.org.

8 Bank for International Settlements. Project mariana: Cross-border exchange of wholesale cbdcs using automated market-makers. Technical report, BIS Innovation Hub, September 2023. URL: https://www.bis.org/publ/othp75.pdf.

9 Bank for International Settlements. Triennial central bank survey: Otc foreign exchange turnover in april 2025. Technical report, Bank for International Settlements, Basel, Switzerland, September 2025. URL: https://bis.org.

10 Cara Bordier, Lukas Frei, and Simon Stalder. Dollar dominance: A source of dollar volatility?, May 2026. URL: https://ssrn.com/abstract=6757018.

11 Agostino Capponi and Ruizhe Jia. The adoption of blockchain-based decentralized exchanges. Papers 2103.08842, arXiv.org, March 2021. URL: https://ideas.repec.org/p/arx/papers/2103.08842.html.

12 Joe Cash. China, Saudi Arabia sign currency swap agreement. *Reuters*, November 2023. URL: https://www.reuters.com/markets/currencies/china-saudi-arabia-central-banks-sign-local-currency-swap-agreement-2023-11-20/.

13 Gary Chamberlain and Michael Rothschild. Arbitrage, factor structure, and mean-variance analysis on large asset markets. *Econometrica*, 51(5):1281–1304, 1983. URL: http://www.jstor.org/stable/1912275.

14 Yu chin Chen and Kenneth Rogoff. Commodity currencies. *Journal of International Economics*, 60(1):133–160, 2003. Emperical Exchange Rate Models. doi:10.1016/S0022-1996(02)00072-7.

15 Jana Choukeir. UAE, Ethiopia sign currency swap agreement with value of up to $817 million. *Reuters*, July 2024. URL: https://www.reuters.com/business/finance/uae-ethiopia-sign-currency-swap-agreement-with-value-up-817-million-2024-07-16/.

16 Virginie Coudert, Cyriac Guillaumin, and Hélène Raymond. Looking at the other side of carry trades: Are there any safe haven currencies? Working Papers 2014-03, CEPII research center, February 2014. URL: https://ideas.repec.org/p/cii/cepidt/2014-03.html.

17 Y. Crama and M. Schyns. Simulated annealing for complex portfolio selection problems. *European Journal of Operational Research*, 150(3):546–571, 2003. Financial Modelling. doi:10.1016/S0377-2217(02)00784-1.

18 Giacomo Di Tollo and Andrea Roli. Metaheuristics for the portfolio selection problem. *International Journal of Operations Research*, 5(1):13–35, 2008. URL: https://www.orstw.org.tw/ijor/vol5no1/2_IJOR_Vol5_No1_pp13_35.pdf.

19 Kimberly Donovan and Maia Nikoladze. The axis of evasion: Behind China's oil trade with Iran and Russia. *Atlantic Council*, March 2024. URL: https://www.atlanticcouncil.org/blogs/new-atlanticist/the-axis-of-evasion-behind-chinas-oil-trade-with-iran-and-russia/.

20 Michael Egorov. StableSwap—Efficient Mechanism for Stablecoin Liquidity. Technical report, Curve Finance, 2019. URL: https://curve.fi/files/stableswap-paper.pdf.

21 Alex Evans, Guillermo Angeris, and Tarun Chitra. Optimal fees for geometric mean market makers, 2021. arXiv:2104.00446.

22 Elena Fabrichnaya and Guy Faulconbridge. What and where are russia's $300 billion in reserves frozen in the west?, December 2023. URL: https://www.reuters.com/world/europe/what-where-are-russias-300-billion-reserves-frozen-west-2023-12-28/.

23 Evan Freidin. Brics pay as a challenge to swift network, December 2024. URL: https://www.lowyinstitute.org/the-interpreter/brics-pay-challenge-swift-network.

24 Krzysztof Gogol, Johnnatan Messias, Malte Schlosser, Benjamin Kraner, and Claudio Tessone. Cross-border exchange of cbdcs using layer-2 blockchain, 2024. doi:10.48550/arXiv.2312.16193.

25 Linda S. Goldberg and Cédric Tille. Vehicle currency use in international trade. *Journal of International Economics*, 76(2):177–192, 2008. doi:10.1016/j.jinteco.2008.07.001.

26 Lioba Heimbach, Ye Wang, and Roger Wattenhofer. Behavior of liquidity providers in decentralized exchanges. Papers 2105.13822, arXiv.org, May 2021. URL: https://ideas.repec.org/p/arx/papers/2105.13822.html.

27 International Monetary Fund. Annual report on exchange arrangements and exchange restrictions (areaer), 2024. URL: https://www.elibrary-areaer.imf.org/.

28 International Monetary Fund. Exchange rates (er). https://data.imf.org/en/datasets/IMF.STA:ER, 2026. Accessed: 2026-05-18.

29 International Monetary Fund. International trade in goods (by partner country) (imts). https://data.imf.org/en/datasets/IMF.STA:IMTS, 2026. Accessed: 2026-05-18.

30 D. Kraft. *A Software Package for Sequential Quadratic Programming*. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht. Wiss. Berichtswesen d. DFVLR, 1988. URL: https://books.google.ca/books?id=4rKaGwAACAAJ.

31 Paul Krugman. The international role of the dollar: Theory and policy implications. In John F. O. Bilson and Richard C. Marston, editors, *Exchange Rate Theory and Practice*, pages 261–278. University of Chicago Press, 1984.

32 Alfred Lehar and Christine Parlour. Decentralized exchange: The uniswap automated market maker. *Journal of Finance*, 80(1):321–374, February 2025. doi:10.1111/jofi.13405.

33 Reina Ke Xin Li, Andreas Park, Andreas Veneris, and Srisht Fateh Singh. Multi-currency amms for decentralized forex markets: Feasibility & optimal design, 2026. arXiv:2607.26405.

34 Alex Lipton and Artur Sepp. Automated market-making for fiat currencies, 2021. arXiv:2109.12196.

35 Katya Malinova and Andreas Park. Dethroning the dollar? multi-currency automated market making for fx. Unpublished.

36 Katya Malinova and Andreas Park. Learning from defi: Would automated market makers improve equity trading?, August 2023. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4531670.

37 Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, USA, 2008.

38 Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952. URL: http://www.jstor.org/stable/2975974.

39 Alexander Marrow. Russia, China find payments workaround as US sanctions net widens, sources say. *Reuters*, June 2024. URL: https://www.reuters.com/world/russia-china-find-payments-workaround-us-sanctions-net-widens-sources-say-2024-06-20/.

40 Fernando Martinelli and Nikolai Mushegian. Balancer: A Non-Custodial Portfolio Manager, Liquidity Provider, and Price Sensor. Technical report, Balancer Labs, September 2019. URL: https://balancer.fi/whitepaper.pdf.

41 Jason Milionis, Ciamac C. Moallemi, and Tim Roughgarden. Automated market making and arbitrage profits in the presence of fees. Papers 2305.14604, arXiv.org, May 2023. URL: https://ideas.repec.org/p/arx/papers/2305.14604.html.

42 Jason Milionis, Ciamac C. Moallemi, Tim Roughgarden, and Anthony Lee Zhang. Automated market making and loss-versus-rebalancing. Papers 2208.06046, arXiv.org, August 2022. URL: https://ideas.repec.org/p/arx/papers/2208.06046.html.

43 Government of India Ministry of External Affairs. Chair's Statement and Outcome Document at BRICS Foreign Ministers' Meeting, May 2026. URL: https://mea.gov.in/bilateral-documents.htm?dtl/41144.

44 Pintail. Uniswap: A good deal for liquidity providers?, January 2019. URL: https://pintail.medium.com/uniswap-a-good-deal-for-liquidity-providers-104c0b6816f2.

45 Ayano Sato, Hayato Nakata, and Jay Percy. Time-variant safe haven currencies. *International Review of Economics & Finance*, 93:316–328, 2024. doi:10.1016/j.iref.2024.04.015.

46 Ritu Sharma. Loaded With Massive Reserves Of Rupees In Indian Banks, Russia Buys "Sensitive" Goods From New Delhi Despite Embargo - Reports, September 2024. URL: https://www.eurasiantimes.com/ne-loaded-with-massive-reserves-of-rupees/.

47 Fabricius Somogyi. Dollar dominance in fx trading. *Management Science*, April 2026. doi:10.1287/mnsc.2024.07656.

48 Benn Steil, Yuma Schuster, and Samuel Zucker. Central bank currency swaps tracker, April 2026. URL: https://www.cfr.org/articles/central-bank-currency-swaps-tracker.

49 United Nations Statistics Division. UN Comtrade Database, 2025. Accessed: 2026-05-19. URL: https://comtradeplus.un.org/.

50 Maria Elena Vizcaino, Zijia Song, and Nicolle Yapur. Resurgent Political Risk Derails Rallies Across Emerging Markets. *Bloomberg Markets*, May 2026. URL: https://www.bloomberg.com/news/articles/2026-05-24/resurgent-political-risk-derails-rallies-across-emerging-markets.

51 Kandy Wong. Saudi Arabia 'open' to petroyuan, closer China ties, minister says. *South China Morning Post*, September 2024. URL: https://www.scmp.com/economy/global-economy/article/3277788/saudi-arabia-open-petroyuan-closer-china-ties-minister-says.

52 Álvaro Cartea, Fayçal Drissi, and Marcello Monga. Decentralised finance and automated market making: Predictable loss and optimal liquidity provision, 2024. arXiv:2309.08431.

53 Álvaro Rubio-García, Juan José García-Ripoll, and Diego Porras. Portfolio optimization with discrete simulated annealing, 2022. arXiv:2210.00807.

A Proofs

A.1 Proofs of benchmark arrangement trading costs

▶ **Proposition 4.1** (Status quo trading cost). *The expected aggregate trading cost in the status quo for $N+1$ currencies routed through the vehicle currency 0 is*

$$c^{SQ} = \sum_i \sqrt{\sigma_i^2 E[Q_i]\Delta}$$

Proof. We sum the costs over the volume of the bilateral trades:

$$\sum_{i<j} Q_{ij}\left(\sqrt{\frac{\Delta\sigma_i^2}{E[Q_i]}} + \sqrt{\frac{\Delta\sigma_j^2}{E[Q_j]}}\right) = \sum_{i<j} Q_{ij}\sqrt{\frac{\Delta\sigma_i^2}{E[Q_i]}} + \sum_{i<j} Q_{ij}\sqrt{\frac{\Delta\sigma_j^2}{E[Q_j]}}.$$

Swapping $i \leftrightarrow j$ in the second sum and using $Q_{ij} = Q_{ji}$, the two combine to give $\sum_i \sum_{j\neq i} \frac{\sigma_i Q_{ij}\sqrt{\Delta}}{\sqrt{E[Q_i]}}$. Hence, in expectation,

$$c^{\mathrm{SQ}} = \sum_i \sqrt{\sigma_i^2 E[Q_i]\Delta}$$ ◀

▶ **Proposition 4.2** (Equal weights trading cost). *The expected aggregate trading cost in an equal-weighted $N+1$ currency constant-mean pool is*

$$c^{SYM} = \sqrt{\frac{2\Delta H E[Q]}{N+1}}$$

Proof. With equal weights, we can drop the weights and recover a constant product invariant. For a trade of size Δ in numeraire from currency i to j, this invariant function requires

$$\left(R_i + \frac{\Delta}{S_i}\right)\left(R_j - \frac{\Delta}{S^{\text{eff}}}\right) = R_i R_j,$$

$$\iff \frac{\Delta}{S^{\text{eff}}} = R_j - \frac{R_i R_j}{R_i + \Delta/S_i} = R_j\left(\frac{\Delta/S_i}{R_i + \Delta/S_i}\right) = \frac{R_j \Delta}{S_i R_i + \Delta} \iff S^{\text{eff}} = \frac{S_i R_i + \Delta}{R_j}$$

The spot price of currency i in terms of currency j implied by the pool is $S_{ij}^{\text{AMM}} = R_j/R_i$. At fundamental prices, $S_{ij}^{\text{AMM}} = S_i/S_j$ implies $S_i R_i = S_j R_j$ for all pairs, so each currency contributes equal value to the pool: $V = (N+1)S_i R_i$ for all i.

Hence, $S^{\text{eff}} = \frac{S_i R_i}{R_j} + \frac{\Delta}{R_j} = S_j + \frac{\Delta(N+1)S_j}{V}$. The price impact $\Pi(\Delta) = \frac{(N+1)\Delta}{V}$ follows from Equation (1) using $S = S_j$.

The pool value $V(T)$ is also governed by the invariant: $\prod_i R_i(0) = \prod_i R_i(T)$, implying

$$\prod_i \frac{V(0)}{S_i(0)(N+1)} = \prod_i \frac{V(T)}{S_i(T)(N+1)} \iff \frac{V(T)}{V(0)} = \prod_i \left(\frac{S_i(T)}{S_i(0)}\right)^{1/(N+1)} = \prod_i x_i^{1/(N+1)}.$$

Meanwhile, the counterfactual hold value is

$$V_{\text{hold}}(T) = \sum_i x_i S_i(0) R_i(0) = \frac{V(0)}{N+1}\sum_i x_i$$

By Equation (2), the IPL is then

$$\text{IPL} = \prod_i x_i^{1/(N+1)} - \frac{1}{N+1}\sum_i x_i$$

For $x_i = 1 + \epsilon_i$ and $\epsilon_i \approx 0$, the second-order Taylor approximation around $\epsilon_i = 0$ is

$$\text{IPL} \approx -\frac{1}{2(N+1)^2}\sum_{i<j}(\epsilon_i - \epsilon_j)^2, \qquad E[|\text{IPL}|] \approx \frac{H}{2(N+1)^2}, \qquad H = \sum_{i<j}\sigma_{ij}^2$$

where the expectation is taken assuming ϵ_i has zero-mean. This results in the break-even pool value $V = \frac{2(N+1)^2 f E[Q]}{H}$ (from Equation (3)) and unit trading cost

$$c(\Delta, f) = \frac{H\Delta}{2(N+1)fE[Q]} + f$$

Minimizing over f (the AM-GM inequality implies that $A/f + Bf \geq 2\sqrt{AB}$ for $f > 0$, and equality here holds when $f = \sqrt{A/B}$) yields

$$f^* = \sqrt{\frac{H\Delta}{2(N+1)E[Q]}}, \qquad c^* = \sqrt{\frac{2H\Delta}{(N+1)E[Q]}}$$

Aggregating over the total expected trade volume $E[Q]$ yields the expected aggregate cost

$$c^{\text{SYM}} = \sqrt{\frac{2HE[Q]\Delta}{(N+1)}}$$

◀

A.2 Proofs of costs in constant-mean AMMs

▶ **Lemma 5.1** (Price impact). *In the general constant-mean AMM, the price impact of a trade of size Δ in numeraire between currencies i and j is approximately*

$$\Pi(\Delta) \approx \frac{w_i + w_j}{w_i w_j} \cdot \frac{\Delta}{2V}$$

Proof. Consider a trade between currencies i and j of notional size Δ in the numeraire (so the trader pays Δ/S_i units of currency i). As only the reserves R_i and R_j change, the invariance function requires

$$\left(R_i + \frac{\Delta}{S_i}\right)^{w_i} \left(R_j - \frac{\Delta}{S^{\text{eff}}}\right)^{w_j} = R_i^{w_i} R_j^{w_j},$$

where Δ/S^{eff} is the number of units of j received. Isolating the output quantity:

$$\frac{\Delta}{S^{\text{eff}}} = R_j \left(1 - \left(1 + \frac{\Delta}{S_i R_i}\right)^{-w_i/w_j}\right)$$

Assuming $\Delta \ll S_i R_i = w_i V$, a second-order Taylor expansion gives

$$\left(1 + \frac{\Delta}{S_i R_i}\right)^{-w_i/w_j} \approx 1 - \frac{w_i}{w_j}\frac{\Delta}{S_i R_i}\left(1 - \frac{1}{2}\frac{\Delta}{S_i R_i}\frac{w_i + w_j}{w_j}\right)$$

The spot price of currency i in terms of currency j is $S_{ij}^{\text{AMM}} = (R_j/w_j)/(R_i/w_i)$. At fundamental prices, $S_{ij}^{\text{AMM}} = S_i/S_j$ implies $S_i R_i/w_i = S_j R_j/w_j$ for all pairs, so each currency i accounts for w_i of total pool value, i.e., $w_i V = S_i R_i$. Then, using $R_j w_i/w_j = (S_i/S_j)R_i$ and $S_i R_i = w_i V$:

$$\frac{\Delta}{S^{\text{eff}}} \approx \frac{\Delta}{S_j}\left(1 - \frac{\Delta}{2V}\frac{w_i + w_j}{w_i w_j}\right)$$

Inverting and applying a first-order expansion (valid for $\Delta \ll 2w_i w_j V/(w_i + w_j)$):

$$S^{\text{eff}} \approx S_j \left(1 + \frac{w_i + w_j}{w_i w_j}\frac{\Delta}{2V}\right)$$

Substituting into (1) yields the result. ◀

▶ **Lemma 5.2** (Impermanent loss). *For gross returns x_i with $x_i \approx 1$ and $E[x_i] = 1$, the expected impermanent loss of the general constant-mean pool satisfies*

$$E[|IPL|] \approx \frac{H_w}{2}, \qquad H_w = \sum_{i<j} w_i w_j \sigma_{ij}^2$$

Proof. We first derive the exact expression for $V(T)$. As derived in the proof of Proposition 5.1, at fundamental prices, $R_i(t) = w_i V(t)/S_i(t)$ for all i. Substituting into the constant-mean invariant:

$$\prod_i \left(\frac{w_i V(T)}{S_i(T)}\right)^{w_i} = \prod_i \left(\frac{w_i V(0)}{S_i(0)}\right)^{w_i}$$

Since $\sum_i w_i = 1$, the V terms factor out as $V(T)^{\sum_i w_i} = V(T)$ and $V(0)^{\sum_i w_i} = V(0)$, giving

$$V(T) = V(0) \prod_i \left(\frac{S_i(T)}{S_i(0)}\right)^{w_i} = V(0) \prod_i x_i^{w_i}$$

The counterfactual hold value is

$$V_{\text{hold}}(T) = \sum_i S_i(T) R_i(0) = \sum_i S_i(T) \cdot \frac{w_i V(0)}{S_i(0)} = V(0) \sum_i w_i x_i$$

Substituting into (2) gives the exact expression $\text{IPL} = \prod_i x_i^{w_i} - \sum_i w_i x_i$.

For $x_i = 1 + \epsilon_i$ and $\epsilon_i \approx 0$, the second-order Taylor expansion around $\epsilon_i = 0$ is

$$\begin{aligned}\text{IPL} &\approx \left(1 + \sum_i w_i \epsilon_i + \frac{1}{2}\left[\left(\sum_i w_i \epsilon_i\right)^2 - \sum_i w_i \epsilon_i^2\right]\right) - \left(1 + \sum_i w_i \epsilon_i\right) \\ &= -\frac{1}{2}\left(\sum_i w_i \epsilon_i^2 - \left(\sum_i w_i \epsilon_i\right)^2\right)\end{aligned}$$

Expanding $(\sum_i w_i \epsilon_i)^2 = \sum_i w_i^2 \epsilon_i^2 + 2\sum_{i<j} w_i w_j \epsilon_i \epsilon_j$, grouping the sums of ϵ_i^2, and using $1 - w_i = \sum_{j \neq i} w_j$ to write $\sum_i w_i(1 - w_i)\epsilon_i^2 = \sum_{i<j} w_i w_j (\epsilon_i^2 + \epsilon_j^2)$, we get

$$\text{IPL} \approx -\frac{1}{2} \sum_{i<j} w_i w_j (\epsilon_i^2 + \epsilon_j^2 - 2\epsilon_i \epsilon_j) = -\frac{1}{2} \sum_{i<j} w_i w_j (\epsilon_i - \epsilon_j)^2$$

The expectation under zero-mean returns is $E[|\text{IPL}|] \approx H_w/2$ where $H_w = \sum_{i<j} w_i w_j \sigma_{ij}^2$. ◀

▶ **Proposition 5.3** (Equilibrium trading cost)**.** *Under the break-even condition* (3) *and Lemmas 5.1–5.2, the aggregate equilibrium trading cost is minimized at*

$$f^* = \sqrt{\frac{\Delta H_w}{4(E[Q])^2} \sum_i \frac{E[Q_i]}{w_i}}, \qquad c^* = \sqrt{\Delta H_w \sum_i \frac{E[Q_i]}{w_i}} \tag{4}$$

Proof. By Lemma 5.2, $E[|\text{IPL}|] \approx H_w/2$. The break-even condition (3) therefore gives equilibrium pool depth

$$V = \frac{2fE[Q]}{H_w}.$$

By Lemma 5.1, the unit trading cost for a pair (i, j) trade of size Δ is

$$c_{ij}(\Delta, f) = \frac{w_i + w_j}{w_i w_j} \frac{\Delta}{2V} + f = \frac{w_i + w_j}{w_i w_j} \frac{H_w \Delta}{4fE[Q]} + f.$$

Aggregating over all pairs, weighted by their respective volumes Q_{ij}:

$$c(\Delta, f) = \frac{H_w \Delta}{4fE[Q]} \sum_{i<j} Q_{ij} \frac{w_i + w_j}{w_i w_j} + fQ.$$

We simplify $\sum_{i<j} Q_{ij}(w_i + w_j)/(w_i w_j) = \sum_{i<j} Q_{ij}/w_j + \sum_{i<j} Q_{ij}/w_i$. Swapping $i \leftrightarrow j$ in the second sum and using $Q_{ij} = Q_{ji}$, both sums combine to give $\sum_i Q_i/w_i$, where $Q_i = \sum_{j \neq i} Q_{ij}$ is the total volume involving currency i. Hence

$$c(\Delta, f) = \frac{H_w \Delta}{4fE[Q]} \sum_i \frac{Q_i}{w_i} + fQ. \tag{5}$$

In expectation, minimizing over f (the AM-GM inequality implies that $A/f + Bf \geq 2\sqrt{AB}$ for $f > 0$, and equality here holds when $f = \sqrt{A/B}$) yields

$$f^* = \sqrt{\frac{H_w \Delta}{4(E[Q])^2} \sum_i \frac{E[Q_i]}{w_i}}, \qquad c^* = \sqrt{\Delta H_w \sum_i \frac{E[Q_i]}{w_i}}$$ ◀

A.3 Proofs of parametrized optimums

▶ **Proposition 6.1** (Optimal weight and cost for vehicle and 2 symmetric currencies). *Under the symmetric (v, s) parametrization for 3 currencies, if $s^2 < 2(1+2v)/(1+v)$, the expected aggregate trading cost of Equation (4) has unique interior minimizer at $w_1^* = w_2^* = \frac{1-w_0^*}{2}$ and*

$$w_0^* = \frac{s}{\sqrt{(1+2v)(4-s^2)}}, \qquad c^* = \sqrt{q\sigma^2\Delta}\sqrt{s^2v + 2 + s\sqrt{(4-s^2)(1+2v)}}$$

If $s^2 \geq 2(1+2v)/(1+v)$, no interior minimizer exists; the infimum over $w_1, w_2 > 0$ is obtained as $w_0 \to 1$.

Proof. With $w_1 = w_2 = (1-w_0)/2$:

$$H_w = w_1w_2\sigma_{12}^2 + w_0w_1\sigma_{01}^2 + w_0w_2\sigma_{02}^2 = \tfrac{1}{4}(1-w_0)^2s^2\sigma^2 + (1-w_0)w_0\sigma^2.$$

The volume-weighted sum is $\sum_i E[Q_i]/w_i = 4q(1+v)/(1-w_0) + 2q/w_0$. After substituting into Equation (4) and simplifying, the expected aggregate cost $c(w_0; s, v)$ satisfies

$$\frac{c(w_0;s,v)^2}{2q\sigma^2\Delta} = \left(\frac{s^2v}{2} + 1\right) + P(w_0;s,v),$$

where

$$P(w_0;s,v) = w_0\left(1 - \tfrac{s^2}{4}\right)(1+2v) + \frac{s^2}{4w_0}.$$

Here, c is monotonically increasing in P, so we solve the equivalent problem of minimizing P. Since $d^2P/dw_0^2 = s^2/(2w_0^3) > 0$ for all $w_0 > 0$, P is strictly convex in w_0 for that region. We have the form $A/w_0 + Bw_0$, which has minimum $2\sqrt{AB}$ attained at $w_0 = \sqrt{A/B}$, yielding

$$w_0^* = \frac{s}{\sqrt{(1+2v)(4-s^2)}}, \qquad P(w_0^*;s,v) = \frac{s}{2}\sqrt{(4-s^2)(1+2v)},$$

When $s^2 < 2(1+2v)/(1+v)$, $w_0^* < 1$ is a valid interior solution with

$$c^* = \sqrt{q\sigma^2\Delta}\sqrt{s^2v + 2 + s\sqrt{(4-s^2)(1+2v)}}.$$

For $s \in [0, 2]$ and $v > 0$, these quantities are well-defined.

When $s^2 \geq 2(1+2v)/(1+v)$, the unconstrained minimizer is $w_0^* \geq 1$, and the convexity of $P(w_0)$ implies that the constrained infimum is attained as $w_0 \to 1$. ◀

▶ **Proposition 6.2** (Optimum for vehicle and N symmetric currencies). *Under the symmetric (v, s) parametrization for $N+1$ currencies, if $s^2 < 2(1+Nv)/(1+(N-1)v)$, the expected aggregate trading cost from Equation (4) has unique interior minimizer at $w_i^* = (1-w_0^*)/N$ for all $i > 0$ and*

$$w_0^* = \frac{s}{\sqrt{(1+Nv)(2N-(N-1)s^2)}}$$

If $s^2 \geq 2(1+Nv)/(1+(N-1)v)$, no interior minimizer exists; the infimum over $w_1, w_2 > 0$ is obtained as $w_0 \to 1$.

To prove the proposition, we first establish the following lemma on the admissible range of s.

▶ **Lemma A.1.** *Under the above parametrization, $s \in \left[0, \sqrt{\frac{2N}{N-1}}\right]$.*

Proof. Let C be the $N \times N$ covariance matrix of the returns of the N symmetric currencies, which has σ^2 on the diagonal and $\sigma^2 - \frac{s^2\sigma^2}{2}$ elsewhere. The variance of any linear combination of the currencies must be non-negative, so R must be positive semi-definite (PSD). For simplicity, we will work with the correlation matrix R which satisfies $\sigma^2 R = C$ and must also be PSD.

We can write $R = (1-\rho)I + \rho u u^T$, where I is the $N \times N$ identity matrix and $u^T = (1, \ldots, 1)$. Then, the eigenvectors and eigenvalues of R satisfy

$$Rv = (1-\rho)v + \rho\langle u, v\rangle u = \lambda v \iff (\lambda + \rho - 1)v = \rho\langle u, v\rangle u.$$

which implies that, either both sides are 0, which corresponds to eigenvalue $\lambda_1 = 1 - \rho$, or $v = au$ for some $a \in \mathbb{R}\setminus\{0\}$ and $\lambda + \rho - 1 = \rho N$, for which we have eigenvalue $\lambda_2 = \rho(N-1)+1$. Thus R is positive semidefinite if and only if

$$1 - \rho \geq 0 \quad \text{and} \quad 1 + (N-1)\rho \geq 0 \iff -\frac{1}{N-1} \leq 1 - \frac{s^2}{2} \leq 1.$$

Rearranging, we have $s \leq \sqrt{\frac{2N}{N-1}}$. ◀

Now, assuming $s \in [0, \sqrt{\frac{2N}{N-1}}]$, we prove the proposition.

Proof. Now, in the N symmetric currency AMM, we have $w_i = (1 - w_0)/N$, implying

$$H_w = \sum_{i>0} \frac{w_0(1-w_0)}{N}\sigma^2 + \sum_{0<i<j} \frac{(1-w_0)^2}{N^2}s^2\sigma^2 = N\frac{w_0(1-w_0)}{N}\sigma^2 + \binom{N}{2}\frac{(1-w_0)^2}{N^2}s^2\sigma^2.$$

This simplifies to

$$H_w = w_0(1-w_0)\sigma^2 + \frac{N-1}{2N}(1-w_0)^2 s^2\sigma^2$$

The volume-weighted sum is

$$\sum_i \frac{E[Q_i]}{w_i} = \frac{Nq}{w_0} + \frac{N(q + vq(N-1))}{(1-w_0)/N} = Nq\left(\frac{(1-w_0) + w_0 \cdot N(1 + v(N-1))}{w_0(1-w_0)}\right)$$

Substituting, the expected aggregate cost is

$$c(\Delta, f) = \sqrt{q\sigma^2 N\Delta}\sqrt{\left(w_0 + \frac{N-1}{2N}(1-w_0)s^2\right)\left(\frac{(1-w_0) + w_0 \cdot N(1+v(N-1))}{w_0}\right)} \quad (6)$$

and, in optimum, satisfies

$$\begin{aligned}\frac{(c^*)^2}{q\sigma^2 N\Delta} &= (1-w_0) + w_0 N(1+v(N-1)) + \frac{N-1}{2Nw_0}(1-w_0)^2 s^2 + \frac{N-1}{2}(1-w_0)s^2(1+v(N-1)) \\ &= 1 + \frac{s^2}{2N}(N-1)(N(1+v(N-1)) - 2) + P(w_0; s, v)\end{aligned}$$

where

$$P(w_0; s, v) = w_0(N-1)(1+Nv)\frac{1}{2N}(2N - (N-1)s^2) + \frac{(N-1)s^2}{2Nw_0}$$

Again, c is strictly monotonically increasing in P, we minimize P instead. Since $d^2P/dw_0^2 = (N-1)s^2/w_0^3 > 0$ for all $w_0 > 0$, P is strictly convex in w_0 for that region. Once again, we have the form $A/w_0 + Bw_0$, which attains its minimum, $2\sqrt{AB}$ at $w_0 = \sqrt{A/B}$, yielding

$$w_0^* = \frac{s}{\sqrt{(1+Nv)(2N-(N-1)s^2)}}, \quad P(w_0^*; s, v) = \frac{s(N-1)}{N}\sqrt{(1+Nv)(2N-(N-1)s^2)},$$

When $s^2 < 2(1+2v)/(1+v)$, $w_0^* < 1$ is a valid interior solution with

$$c^* = \sqrt{q\sigma^2\Delta}\sqrt{N + \frac{s^2}{2}(N-1)(N(1+v(N-1))-2) + s(N-1)\sqrt{(1+Nv)(2N-(N-1)s^2)}} \tag{7}$$

These quantities are well-defined for $v > 0$ and the range of s derived in Lemma A.1.

When $s^2 \geq 2(1+Nv)/(1+(N-1)v)$, the unconstrained minimizer is $w_0^* \geq 1$, and the convexity of $P(w_0)$ implies that the constrained infimum is attained as $w_0 \to 1$. ◀

A.4 Proof of tractable weight optimization

▶ **Proposition 7.1** (Approximate optimal weights). *When $H_i > 0$ for all i and $E[Q_i] > 0$ for some i, the minimizer of $\tilde{\mathcal{F}}(w) = \sum_i \frac{E[Q_i]}{w_i H_i}$ subject to $\sum_i w_i = 1$, $w_i > 0$ $\forall i$, is*

$$w_i^* = \frac{1}{D}\sqrt{\frac{E[Q_i]}{H_i}}, \qquad \forall\, i = 0, \ldots, N, \qquad \text{where } D = \sum_j \sqrt{E[Q_j]/H_j},$$

Proof. We minimize $\tilde{\mathcal{F}}(w) = \sum_i \frac{E[Q_i]}{w_i H_i}$ subject to $\sum_i w_i = 1$, $w_i > 0$, where $H_i = \sum_{j\neq i}\sigma_{ij}^2$, via the Lagrangian

$$\mathcal{L}(w, \lambda) = \sum_i \frac{E[Q_i]}{w_i H_i} + \lambda\left(\sum_i w_i - 1\right).$$

The first-order conditions $\nabla_w \mathcal{L}(w, \lambda) = 0$ gives $\frac{E[Q_i]}{w_i^2 H_i} = \lambda$, so $w_i = \sqrt{\frac{E[Q_i]}{\lambda H_i}}$ for all i. Imposing $\sum_i w_i = 1$ through the condition $\nabla_\lambda \mathcal{L}(w, \lambda) = 0$:

$$\frac{1}{\sqrt{\lambda}}\sum_j \sqrt{E[Q_j]/H_j} = 1 \implies \sqrt{\lambda} = \sum_j \sqrt{E[Q_j]/H_j} \equiv D, \implies w_i^* = D^{-1}\sqrt{E[Q_i]/H_i}$$

The hessian $\nabla^2_{ww}\mathcal{L}(w, \lambda)$ has $\frac{\partial^2 \mathcal{L}}{\partial w_i^2} = \frac{2E[Q_i]}{w_i^3 H_i}$ on the diagonal and $\frac{\partial^2 \mathcal{L}}{\partial w_i \partial w_j} = 0$ elsewhere, so the second-order sufficient condition, $v^T \nabla^2_{ww}\mathcal{L}(w, \lambda)v > 0$ for all $v \neq 0$, becomes

$$\sum_i v_i^2 \frac{\partial^2 \mathcal{L}}{\partial w_i^2} = \sum_i v_i^2 \frac{2E[Q_i]}{w_i^3 H_i} > 0,$$

which holds when all $w_i > 0$, so this a global minimum for $w_i > 0$. ◀

B–D Full Version Material

Appendices B, C, and D are omitted here due to space limitations. They can be found in their respective Appendix sections in the full version of this paper [33].

Uniform-Loss Automated Market Making for Prediction Markets

Ciamac C. Moallemi
Decision, Risk, and Operations Division, Graduate School of Business, Columbia University, New York, NY, USA
Paradigm, New York, NY, USA
Uniswap Labs, New York, NY, USA

Dan Robinson
Paradigm, New York, NY, USA

Brian Zhu
Department of Industrial Engineering and Operations Research, Columbia University, New York, NY, USA

Abstract

Automated market makers (AMMs) for prediction markets descend from market scoring rules, where a mechanism operator subsidizes a market to aggregate beliefs about uncertain events. The existing literature has focused on bounding the total worst-case loss to the subsidizer, but has not addressed how that loss is distributed across price states or over time. We use the framework of loss-versus-rebalancing (LVR) to study this distribution and introduce *uniform AMMs*, defined by the property that instantaneous LVR is proportional to pool value and independent of the current token price. In a static setting, we show that for a broad class of *win-martingales* – processes that converge to 0 or 1 at a fixed resolution time – there exists a pricing function that achieves uniform LVR under that process, and conversely, that any sufficiently regular pricing function induces a win-martingale under which it is uniform. We then extend the framework to dynamic liquidity management, showing that liquidity levels can be adjusted over time to implement a prescribed target expected cumulative loss schedule. This theory is illustrated with canonical examples of win-martingales and pricing functions. Our results can inform AMM designers and liquidity providers on how the inevitable cost of subsidizing price discovery can be shaped and controlled across both price and time.

2012 ACM Subject Classification Applied computing → Economics

Keywords and phrases Prediction markets, automated market makers, loss-versus-rebalancing, win-martingales, liquidity provision

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.14

Category Extended Abstract

Related Version *Full Version*: `https://arxiv.org/abs/2607.17428`

Funding *Ciamac C. Moallemi*: CCM is supported by the Briger Family Digital Finance Lab at Columbia University. CCM is a research advisor for Paradigm and Uniswap Labs.

Binary prediction markets trade YES and NO claims whose prices sum to one and which resolve to $(1, 0)$ or $(0, 1)$. Market scoring rules and their equivalent invariant-based AMMs make these markets liquid by committing capital that informed traders can earn as they move stale quotes toward fair value. Classical bounded-loss guarantees control *how much* this subsidy can cost, but not *where* in the price space or *when* before resolution it is spent.

LVR measures the predictable transfer from liquidity providers (LPs) in an AMM to arbitrageurs Milionis et al. [3], and provides the framework to quantify where and when losses occur, and can also be interpreted as an LP's subsidy for information discovery. We


© Ciamac C. Moallemi, Dan Robinson, and Brian Zhu;
licensed under Creative Commons License CC-BY 4.0



propose *uniform LVR* as a desiderata of AMM design for prediction markets (or binary option markets in general): after accounting for the common rate of information arrival, every probability state incurs the same instantaneous LVR per dollar of pool value. Besides resulting in state-independent losses and thus removing price-based timing incentives for LPs, we show that uniform LVR minimizes a worst-case relative loss criterion across price states.

We assume that the fair YES price $P_t \in [0,1]$ is a continuous *win-martingale*, i.e., P is a martingale and $P_T \in \{0,1\}$ almost surely at resolution time T. To balance breadth with tractability, we study a class of separable dynamics:

$$dP_t = \frac{G(P_t)}{h(t)}\,dW_t, \qquad \int_0^t h(s)^{-2}ds < \infty \ (t<T), \quad \int_0^T h(s)^{-2}ds = \infty. \tag{1}$$

Here G determines where beliefs are most responsive to information, while h determines when information arrives. The divergent information clock guarantees binary resolution under standard regularity and absorbing-boundary conditions on G.

We represent an AMM by the value $V(p)$ of its arbitraged reserves at prices $(p, 1-p)$. The AMM's YES and NO reserves are $X(p) = V(p) + (1-p)\cdot V'(p)$ and $Y(p) = V(p) - p\cdot V'(p)$, respectively. Itô's formula yields pool-value dynamics and instantaneous LVR rate:

$$dV(P_t) = V'(P_t)\cdot\frac{G(P_t)}{h(t)}\,dW_t - \mathrm{LVR}_t dt, \qquad \mathrm{LVR}_t = -\frac{1}{2}\cdot V''(P_t)\cdot\frac{G(P_t)^2}{h(t)^2}\ . \tag{2}$$

Uniformity requires $\mathrm{LVR}_t/V(P_t)$, after normalizing by the information clock $h(t)^{-2}$, to be constant in P_t. Equivalently, the pool-value function solves the boundary value problem

$$G(p)^2\cdot V''(p) + \beta\cdot V(p) = 0\ , \qquad V(0) = V(1) = 0, \qquad V(p) > 0 \quad (0<p<1). \tag{3}$$

Under mild endpoint regularity, this singular Sturm–Liouville problem has a positive, concave, and unimodal solution. Conversely, any sufficiently regular strictly concave V induces a win-martingale under which V has uniform LVR through the volatility profile

$$G(p) = \sqrt{\beta\cdot\frac{V(p)}{-V''(p)}}. \tag{4}$$

This is the paper's central bidirectional correspondence: underlying belief dynamics determine the AMM geometry that equalizes losses, while an existing invariant reveals the belief process under which it is uniform. Canonical examples illustrate this correspondence; for example:

- Logistic volatility $G(p) = p(1-p)$ yields $V(p) = 2\sqrt{p(1-p)}$, i.e., the constant-product invariant $xy = L^2$.
- Wright–Fisher volatility $G(p) = \sqrt{p(1-p)}$ yields $V(p) = p(1-p)$, i.e., the constant-elasticity invariant $\sqrt{x} + \sqrt{y} = \sqrt{L}$.
- Gaussian score dynamics Archak and Ipeirotis [1], $G(p) = \phi(\Phi^{-1}(p))$ yield $V(p) = \phi(\Phi^{-1}(p))$, the pm-AMM in [4].
- The "most exciting game" Backhoff-Veraguas and Beiglböck [2] has volatilify profile $G(p) = \sin(\pi p)/\pi$ and yields a family of solutions for V based on Ferrers functions.
- In the converse direction, the LMSR value $V(p) = L[-p\log p - (1-p)\log(1-p)]$ is uniform for $G(p) = \sqrt{p(1-p)\cdot H(p)}$, where H is the binary entropy function.

Finally, the invariant controls the subsidy across *states*, while a deterministic liquidity scale L_t controls it across *time*. For any admissible differentiable, nondecreasing target expected cumulative loss fraction $D(t)$ and initial wealth w_0, the schedule

$$L_t = \frac{2w_0}{\beta V(P_0)}h(t)^2\exp\left(\int_0^t\frac{\beta}{2h(s)^2}\,ds\right)D'(t) \tag{5}$$

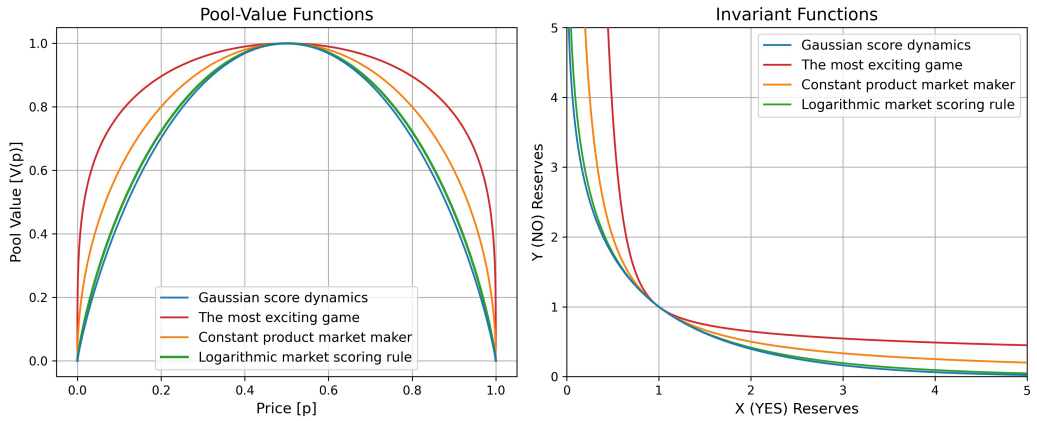


Figure 1 Normalized pool-value functions (left) and their reserve invariants (right). The same value profile can be translated into reserve demands.

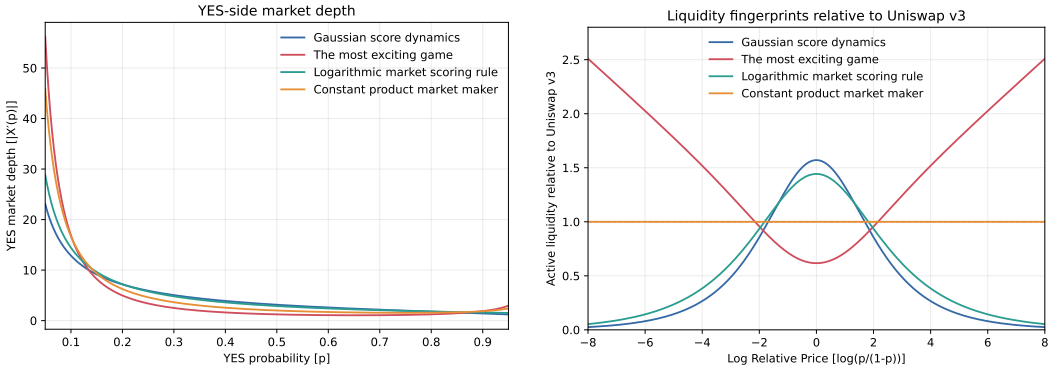


Figure 2 Left: YES-side market depth $|X'(p)|$, the reserve quantity available per unit change in the implied YES probability. Right: Uniswap-v3-equivalent *liquidity fingerprints*. For relative price $q = p/(1-p)$, the fingerprint translates local reserve response into the active liquidity $\ell(q) = -2q^{3/2}dX/dq$ that would reproduce the same infinitesimal price impact in Uniswap v3, normalized by the constant-product benchmark. It exposes where an invariant concentrates effective liquidity even when it is not implemented in Uniswap v3.

implements $\mathbb{E}[W_t] = w_0(1 - D(t))$ while preserving $\mathrm{LVR}_t = \beta L_t V(P_t)/[2h(t)^2]$, subject to initial cash feasibility. Hence the same uniform invariant can support front-loaded or back-loaded price-discovery programs without changing which probability states are subsidized. Read the full paper at `https://arxiv.org/abs/2607.17428`.

References

1 Nikolay Archak and Panagiotis G. Ipeirotis. Modeling volatility in prediction markets. In *Proceedings of the 10th ACM Conference on Electronic Commerce*, EC '09, pages 231–238, New York, NY, USA, 2009. Association for Computing Machinery. `doi:10.1145/1566374.1566414`.

2 Julio Backhoff-Veraguas and Mathias Beiglböck. The most exciting game. *arXiv preprint arXiv:2305.14037*, 2023. `arXiv:2305.14037`.

3 Jason Milionis, Ciamac C. Moallemi, Tim Roughgarden, and Anthony Lee Zhang. Automated market making and loss-versus-rebalancing. *arXiv preprint arXiv:2208.06046*, 2022. `arXiv:2208.06046`.

4 Ciamac Moallemi and Dan Robinson. pm-amm: A uniform amm for prediction markets. *Paradigm Research Blog Post*, 2024.

ParlayMarket: Automated Market Making for Parlay-Style Joint Contracts

Ranvir Rana
Kaleidoscope Blockchain Inc., Raleigh, NC, USA

Viraj Nadkarni
Princeton University, NJ, USA

Niusha Moshrefi
Princeton University, NJ, USA

Pramod Viswanath
Princeton University, NJ, USA

Abstract

Prediction markets are powerful mechanisms for information aggregation, but existing designs are optimized for single-event contracts. Traders frequently express beliefs about joint outcomes – sports parlays, conditional forecasts, multi-scenario financial bets. Current platforms either prohibit such trades or rely on ad hoc mechanisms that ignore correlation structure, resulting in inefficient prices and fragmented liquidity. We introduce ***ParlayMarket***, an automated market-maker for parlay-style joint contracts. The mechanism maintains a shared pairwise exponential-family belief state, so all base and parlay prices are marginals of one coherent distribution. This compresses the 2^M outcome space into $O(M^2)$ sufficient statistics and allows one liquidity pool to support an exponentially large family. Our main result characterizes the resulting learning and loss dynamics. Under repeated trading, prices converge to the best pairwise approximation of the true joint distribution. The induced expected market-maker loss grows at most quadratically in the number of base events, rather than exponentially in the number of listed parlays; moreover, this quadratic dependence is worst-case optimal for dense pairwise dependence, since there are $O(M^2)$ independent correlation directions to learn. Parlay trades are essential to this guarantee: they provide direct constraints on joint outcomes and reduce steady-state error relative to learning from marginal trades alone. Experiments on synthetic correlated markets and historical Kalshi combo data confirm the predicted scaling and show that the mechanism remains effective in realistic market-making settings. Our results demonstrate that combinatorial expressiveness does not require combinatorial capital.

2012 ACM Subject Classification Theory of computation → Market equilibria; Theory of computation → Online learning theory; Applied computing → Economics

Keywords and phrases Prediction markets, automated market makers, parlays, market scoring rules, online learning

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.15

Related Version *Full Version*: https://arxiv.org/abs/2603.22596

1 Introduction

Two uses of knowledge in a society. Knowledge in an economy can be used in two ways – *factually* and *counterfactually*. Factual knowledge concerns statements such as "the price of oil will be greater than $100 on March 29" or "the election will be won by party P." Counterfactual knowledge, by contrast, concerns statements such as "if the price of oil is greater than $100 on March 29, then the price of a banana will be greater than $2 on April 30" or "if the election is won by party P, then the probability of a recession will increase." Prediction markets have proven to be a powerful mechanism for aggregating

© Ranvir Rana, Viraj Nadkarni, Niusha Moshrefi, and Pramod Viswanath;
licensed under Creative Commons License CC-BY 4.0

and eliciting factual knowledge. The prices quoted by a prediction market can serve as a public good, allowing uninformed members of the economy to peer into the future through a market-implied best guess and plan accordingly.

The liquidity problem. Despite this promise, prediction-market liquidity is highly uneven. Existing markets tend to have substantial liquidity in sensational or gambling-oriented domains such as sports, while economically useful markets, including public policy, local politics, and macroeconomic indicators, are often woefully illiquid [9]. Automated market makers (AMMs) are particularly well suited to such illiquid markets because they can provide continuous liquidity without requiring a counterparty [11, 12]. However, conventional AMMs incur losses on their locked liquidity in exchange for eliciting information. In this sense, subsidies are required to induce traders to reveal factual knowledge.

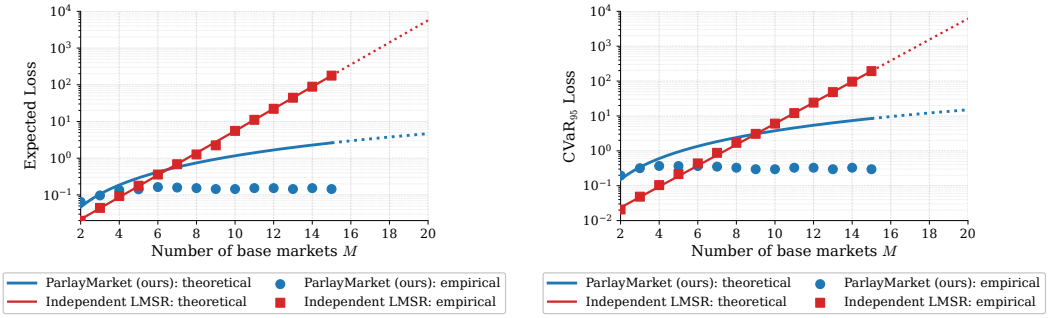
The joint events problem. A further challenge arises when the information of interest is not merely factual, but counterfactual. Prediction markets aggregate dispersed information into prices, but practical automated market makers are primarily designed for single-event contracts. In particular, bounded-loss pricing for single markets is typically achieved through Hanson's logarithmic market scoring rule (LMSR) [11, 12]. In many applications, however, traders reason about joint outcomes: sports parlays, conditional forecasts across related events, or multi-scenario financial bets. The probability of such a contract depends not only on the marginal probabilities of its constituent events, but also on the dependence structure among them.

Computational complexity. Supporting such joint contracts introduces a fundamental challenge. With M binary events, the joint outcome space contains 2^M possible realizations. A naive combinatorial market maker must maintain prices and risk exposure across all of these outcomes, leading to worst-case loss and computational requirements that scale exponentially in M. This renders direct approaches intractable beyond very small systems. The challenge is well known in combinatorial prediction markets: allowing direct trading over joint outcome spaces provides expressive contracts, but typically incurs exponential state and inference costs [12, 8]. Existing practical systems therefore either avoid parlay-style contracts or handle them through ad hoc mechanisms such as request-for-quote workflows, which fragment liquidity and need not maintain a coherent joint distribution.

Our contributions. This paper shows that *combinatorial expressiveness does not require combinatorial capital*. We introduce *ParlayMarket*, an automated market maker that maintains a shared probabilistic belief over all outcomes and derives every base and parlay price as a marginal of this belief. The key design choice is to restrict the belief state to a pairwise exponential family. Rather than tracking one state variable per joint outcome, the mechanism maintains only

$$M + \binom{M}{2} = O(M^2)$$

sufficient statistics: one marginal parameter per event and one interaction parameter per pair. This is the smallest generic exponential-family representation capable of encoding arbitrary pairwise dependence: independence cannot represent correlations, while an unrestricted joint model is exponential. Statistically, this model is also the maximum-entropy distribution matching the learned first and second moments, and therefore does not introduce higher-order structure unsupported by the observed trade flow [25].



(a) Expected Loss. **(b)** CVaR_{95} Loss.

Figure 1 ParlayMarket enables 2^M markets with $O(M^2)$ capital. Theoretical curves show the asymptotic loss scaling implied by the analysis in Section 5. Empirical curves are obtained from controlled simulation of correlated binary markets under the Gaussian score model described later in Sections 3 and 6, and confirm the same polynomial-vs.-exponential separation in practice.

Our approach is related in spirit to inference-based views of prediction markets, where prices are treated as marginals of an underlying probabilistic model [10, 8, 17]. The key difference is that ParlayMarket learns the dependence structure online from endogenous order flow rather than imposing a fixed graphical structure ex ante. Trades in both base markets and parlay markets are interpreted as observations about the underlying joint distribution. After each trade, the mechanism updates its shared parameters, allowing information from one contract to propagate immediately to related contracts. This couples liquidity and learning across the entire contract family rather than maintaining separate books for each parlay.

We show that this design satisfies three central objectives. First, it pools liquidity across markets by pricing all contracts from a shared state, avoiding fragmentation. Second, it achieves bounded loss through an LMSR-style cost function defined over this shared representation, with loss that grows at most quadratically in M, rather than exponentially in the number of joint outcomes. Moreover, this quadratic dependence is worst-case optimal for dense pairwise dependence. Third, it aggregates information efficiently: as trades accumulate, the model converges to the best pairwise approximation of the true joint distribution under the observed data. This does not claim that higher-order dependence is absent; rather, it identifies the pairwise family as the practical frontier at which global coherence, online learnability, and non-combinatorial loss can all be achieved simultaneously.

Conceptually, ParlayMarket transforms the market into an online inference system. Empirically, controlled simulations confirm the predicted polynomial-versus-exponential loss separation, as shown in Figure 1, and a replay on historical Kalshi combo data shows that the mechanism remains effective as a practical correlation-aware quoting engine. Together, these results show that parlay-style prediction markets can be supported by an automated market maker without full joint enumeration or separate capitalization for every combination.

Paper Overview. The remainder of the paper is organized as follows. In Section 2, we review background on prediction markets, combinatorial and parlay markets, and probabilistic models of dependence, and position our work relative to prior approaches. In Section 3, we formalize the problem and define the design objectives for parlay-capable market makers. Section 4 analyzes existing approaches and shows that they fail to simultaneously satisfy these objectives due to fragmented liquidity and combinatorial loss. Section 5 introduces ParlayMarket and

describes its pricing and update mechanisms. We then present the theoretical results on convergence, loss bounds, and information aggregation. Finally, Section 6 evaluates the mechanism empirically, and Section 7 discusses implications and future directions. Additional proofs, experimental results, and implementation details are provided in the full version of the paper.

2 Background and Related Work

Cost-function market makers. Prediction markets are commonly implemented using cost-function market makers, most notably Hanson's logarithmic market scoring rule (LMSR) [12, 13]. LMSR provides continuous liquidity, proper-scoring-rule prices, and bounded worst-case loss. Chen and Pennock [7] interpret LMSR as a utility-based market maker, while Abernethy et al. [1] connect market scoring rules to convex optimization, online learning, and Bregman divergence minimization. Related work on exponential-family markets [2] shows how bounded-loss market makers can be understood as maintaining beliefs in a parametric family. CFMMs provide a parallel AMM tradition in decentralized finance [3, 19], but these mechanisms do not by themselves enforce probabilistic consistency across related event contracts.

Combinatorial and Parlay Markets. Combinatorial prediction markets allow trading over joint outcome spaces [12], but exact implementations require reasoning over 2^M outcomes and become intractable for large M. Prior work therefore restricts the security space or market geometry to obtain tractable loss bounds [15], or studies settings where simple trader interfaces fail to recover coherent joint beliefs [20]. In practice, platforms such as Polymarket and Kalshi [21, 16] rarely implement full combinatorial AMMs; parlay-like products are typically handled through RFQ workflows or separate markets, which fragment liquidity and need not maintain a single coherent joint distribution.

Graphical models and inference-based markets. Graphical models offer a natural way to represent dependence compactly. Pairwise Markov random fields, including Ising models, represent binary joint distributions using only first-order and pairwise interaction terms and correspond to maximum-entropy distributions matching specified marginals and pairwise moments [25]. Learning such models has been studied extensively [22, 5, 24], and online variants connect naturally to stochastic optimization and online convex programming [26, 14]. Prediction markets have also been interpreted as probabilistic inference systems [10, 6]; Bayesian-network and graphical-model market makers make this connection explicit [8, 17]. Unlike those mechanisms, *ParlayMarket* learns the dependence structure endogenously from singleton and parlay order flow while using one shared state to quote all contracts coherently.

3 Model and Design Objectives

We formalize automated market making for correlated prediction markets as a joint mechanism design and statistical inference problem. The market maker interacts with a stochastic stream of trades that reveal partial information about an underlying joint distribution, and must simultaneously provide liquidity, control risk, and learn the dependence structure.

3.1 Market Setting

We consider a collection of M binary events $X_1, \ldots, X_M \in \{0,1\}$, and denote a joint outcome by $x \in \{0,1\}^M$. The uncertainty over outcomes is governed by an unknown ground-truth distribution P^* over $\{0,1\}^M$, which captures both marginal probabilities and dependencies among events.

Contracts. The market supports trading in a family of payoff functions indexed by subsets $S \subseteq \{1, \ldots, M\}$. Each contract corresponds to a Boolean payoff

$$f_S(x) = \prod_{i \in S} x_i \tag{1}$$

which pays 1 if all events in S occur and 0 otherwise. In particular, singleton sets $S = \{i\}$ recover base contracts, while larger subsets correspond to parlay contracts. Let $\mathcal{F}$ denote the set of all supported contracts.

Pricing and coherence. At any time, the market maker maintains a parametric distribution $P_{\boldsymbol{\varphi}}$ over $\{0,1\}^M$, which induces prices for all contracts via expectations:

$$p_S(\boldsymbol{\varphi}) = \mathbb{E}_{x \sim P_{\boldsymbol{\varphi}}}[f_S(x)] \tag{2}$$

This ensures probabilistic coherence, in the sense that all prices are consistent with a single joint distribution. In particular, marginal and joint prices satisfy the usual consistency constraints implied by $P_{\boldsymbol{\varphi}}$.

3.2 Trader/Information Model

We model the market as an online information aggregation process in which traders arrive sequentially and trade based on private signals about the underlying outcome distribution.

Underlying process. The underlying uncertainty in each base market is driven by a correlated Gaussian score process. Specifically, there are M base markets where market i resolves YES if an underlying score $S_i(T)$ exceeds threshold K_i at expiry T. The scores follow correlated arithmetic Brownian motion

$$dS_i(t) = \sigma_i \, dW_i(t), \qquad \mathrm{Cov}(dW_i, dW_j) = \rho_{ij} \, dt, \tag{3}$$

so that $S_T \mid S_t \sim \mathcal{N}(S_t, \Sigma)$ with $\Sigma_{ij} = \rho_{ij}\sigma_i\sigma_j(T-t)$. The binary outcome of market i is $X_i = \mathbf{1}[S_i(T) > K_i]$.

The Gaussian score model is a natural choice for several reasons. First, it provides a tractable closed-form expression for the conditional probability of each outcome given the current score, enabling informed traders to compute exact signals. Second, the pairwise correlation structure ρ_{ij} maps cleanly onto the interaction parameters W_{ij} of the Ising approximation used by ParlayMarket, making the theoretical analysis transparent. Third, it is a standard model for continuous-time prediction markets and sports scoring processes, and admits straightforward emulation from real-world data by calibrating $(\sigma_i, \rho_{ij}, K_i)$ to observed outcome frequencies and co-occurrences [23]. In particular, it captures how the volatility in the likelihood of an event increases as we move closer to when the outcome is going to be revealed.

Trader types. At each time step, a trader arrives and interacts with a single contract. *Informed traders* observe the true underlying score process S_t and target the exact conditional probability. For base market i they submit $\hat{p}_i = \Phi\left(\frac{S_t^i - K_i}{\sigma_i\sqrt{T-t}}\right)$, and for parlay $\mathcal{S}$ they submit the exact multivariate normal orthant probability $P\big(\bigcap_{i\in\mathcal{S}}\{S_i(T) > K_i\} \mid S_t\big)$. *Noise traders* observe a noisy signal $\tilde{S}_t^i = S_t^i + \varepsilon^i$ with $\varepsilon \sim \mathcal{N}(0, \mathrm{diag}(\sigma_i^2(T-t)))$ independent of the path, and target the resulting noisy probability estimate. The noise-trader fraction is $\alpha \in [0, 1]$.

Information Structure. Each trader has access only to a local signal about a bounded number of events – there is no global information. Informed traders observe the score process $S_t \in \mathbb{R}^M$ for all M base markets, but this is a signal about the current conditional distribution only; no trader has access to the full future path or to the joint distribution P^* directly. PM itself must learn P^* online from the sequence of incoming trades.

3.3 Market Interface

The market maker exposes a pricing and trading interface over the contract set $\mathcal{F}$. Trades are executed via a cost function $C_S(\cdot;\theta)$, which determines the payment required to adjust the position in contract S. Following each trade, the market maker updates its internal parameters θ, which in turn updates prices across all contracts.

Interaction model. Traders arrive sequentially. At each round t, one market m_t is selected uniformly at random from the $M + P$ markets, and a single trader arrives. With probability $1-\alpha$ the arriving trader is informed; with probability α they are a noise trader. Competition is fair at each step: every arriving trader observes the current PM price and decides whether and how much to trade before the next trader arrives.

3.4 Design Objectives

We formalize the requirements for a market maker supporting correlated contracts.

Property 1: Liquidity Consistency. The market maker should provide comparable liquidity across all contracts without requiring independent capital allocation.

Let $I_m(x;\phi)$ denote the price impact of a trade x in market m. Then there should exist constants $0 < c_1 \le c_2 < \infty$ such that for all sufficiently small admissible trade sizes x, such that there is non-zero price impact across all markets,

$$c_1 \le \frac{I_m(x;\phi)}{I_{m'}(x;\phi)} \le c_2 \qquad \forall m, m' \in \mathcal{M} \tag{4}$$

That is, price impact should remain comparable across all markets, including both base and parlay contracts, so that parlays are not disproportionately illiquid.

Property 2: Controlled loss. The market maker should control expected aggregate loss and tail risk under the induced joint process of trades and outcomes.

Let L_T denote cumulative realized AMM loss over all executed trades up to horizon T. A satisfactory mechanism should control both expected aggregate loss and tail risk under the induced joint process of trades and outcomes.

First, expected loss should scale subexponentially in the number of base markets, ideally linearly:

$$\mathbb{E}[L_T] = o(2^M), \qquad \text{ideally } \mathbb{E}[L_T] = O(M) \tag{5}$$

Second, tail loss should also remain controlled. For a confidence level α such as 0.95, define

$$\mathrm{VaR}_\alpha(L_T) := \inf\{\ell : \Pr(L_T \le \ell) \ge \alpha\} \tag{6}$$

Then the mechanism should satisfy

$$\mathrm{VaR}_{0.95}(L_T) = o(2^M), \qquad \text{ideally } \mathrm{VaR}_{0.95}(L_T) = O(M) \tag{7}$$

Optionally, the same requirement can be imposed on $\mathrm{CVaR}_{0.95}(L_T)$.

Property 3: Information aggregation. The market maker should aggregate information from trades so that the implied distribution converges to the true distribution P^*.

Let P_{ϕ_T} denote the distribution represented by the AMM after T trades. A satisfactory mechanism should use singleton and parlay order flow to recover the latent joint law with small statistical error.

In particular, after a number of trades linear in the effective model dimension, the represented distribution should be close to the true distribution in KL divergence:

$$D_{\mathrm{KL}}(P^\star \,\|\, P_{\phi_T}) \le \varepsilon \qquad \text{for } T = O(n), \tag{8}$$

where n denotes the relevant model dimension or number of informative parameters. Equivalently, the KL error should decrease with the number of informative trades:

$$\mathbb{E}[D_{\mathrm{KL}}(P^\star \,\|\, P_{\phi_T})] \to 0 \quad \text{as } T \text{ grows.} \tag{9}$$

Implication for market design. Taken together, Properties 1–3 indicate that neither pure contract-by-contract execution nor pure contract-by-contract pricing is sufficient. Treating each exposed parlay as an isolated market tends to fragment liquidity and scale risk with the number of listed contracts, while quoting combinations purely as functions of current base-market prices can improve accessibility but does not, by itself, recover the latent joint law. The design problem is therefore to support executable parlay trading while pooling risk and information strongly enough to preserve liquidity, control aggregate loss, and use order flow to learn dependence. The next section examines two natural baseline approaches along these lines and shows where each falls short.

4 Baseline Designs for Parlay-style Markets

The properties defined in Section 3 impose stringent requirements on any viable market design. We examine two natural baseline approaches – request-for-quote systems and independence-based automated market makers – and show that each fails to satisfy these requirements for fundamental, structural reasons. Additional empirical details, robustness checks, and full raw results are provided in the full version.

4.1 Request-for-Quote Protocol

In many existing platforms, parlay trades are handled via a request-for-quote (RFQ) mechanism: a trader specifies a bundle of outcomes, and a market maker returns a price based on proprietary models. This architecture is flexible, but it externalizes both pricing and risk management to separate dealers. As a result, quotes are not necessarily derived from a common joint distribution, and liquidity is fragmented across requested combinations.

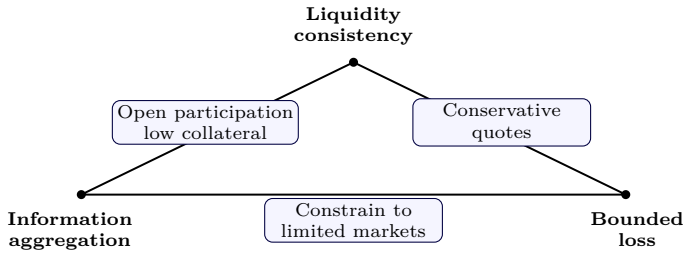


Figure 2 The RFQ trilemma.

Table 1 Average spreads in base and combo markets on Kalshi, restricting to markets with open interest between 1,000 and 2,000. Combo markets exhibit substantially wider spreads than base markets.

Market type	Number of markets	Average spread (cents)
Base markets	999	5.12
Combo markets	549	11.89

Shortcomings of Using RFQ Protocol for Parlays. This creates two structural failures. First, liquidity deteriorates for combinations: in our Kalshi combo sample, average spreads are substantially larger in combo markets than in base markets (Table 1). Second, RFQ quotes need not be probabilistically coherent. Since quotes are not marginals of a shared joint law, they can violate basic constraints such as the Fréchet bound $P(A \cap B) \leq \min\{P(A), P(B)$; detailed violations are reported in Appendix B.1 of the full version. Aggregating quotes from multiple market makers yields only marginal improvements in spread, suggesting that fragmentation persists because competition among quote providers remains limited. A detailed breakdown by leg count and market is provided in Appendix B.2 of the full version.

The RFQ trilemma. The RFQ architecture faces a fundamental trilemma among the three properties introduced in Section 3. Opening maker participation can improve quote availability and price discovery, but weakens loss control because risk is dispersed across less tightly capitalized makers. Conservative quoting protects makers from adverse selection, but the resulting quotes become poor probability estimates and therefore carry little information. Restricting the set of quoted combinations makes risk easier to manage, but sacrifices broad parlay liquidity. Thus, over a broad parlay family, RFQ systems can trade off among liquidity consistency, coherent information aggregation, and bounded loss, but cannot achieve all three simultaneously. We call this structural constraint the *RFQ trilemma* (Figure 2).

4.2 Independence-Based Parlay AMM

A second natural baseline is to keep independent AMMs for the base markets and quote a parlay as the product of its marginal prices:

$$q_{\text{ind}}(E_1 \cap \cdots \cap E_k) = \prod_{i=1}^{k} \hat{p}_i. \tag{10}$$

This gives immediate executable quotes, but it hard-codes independence and therefore fails whenever outcomes are correlated.

The failure is structural. Consider two positively correlated events with common marginal probability p. The independence quote for the joint contract is p^2, while the true joint probability can be close to p. An informed trader can repeatedly buy the parlay at p^2 against expected payoff p. Because parlay trades do not update the base-market prices or any shared correlation state, the mispricing does not disappear through trading. The AMM therefore fails both bounded loss and information aggregation: it cannot learn the dependence structure no matter how much parlay order flow it observes.

4.3 Towards a Joint Distribution Market Maker

These observations suggest that any viable design must satisfy three structural requirements: (i) maintain a shared representation of the joint distribution, (ii) update this representation from both base and composite trades, and (iii) derive all prices coherently from this shared state. The next section introduces ParlayMarket, an AMM that satisfies these requirements by maintaining a tractable approximation to the joint distribution and updating it via online learning from trade flow.

5 ParlayMarket

The limitations of baseline designs in Section 4 arise from the absence of a shared global state and the inability to propagate information across markets. We now construct a market maker that resolves these limitations by maintaining and updating a coherent joint distribution over all outcomes.

At a high level, ParlayMarket interprets each trade – whether in a base market or a parlay – as a noisy observation about the underlying joint distribution $P^\star$. The mechanism maintains a parametric approximation P_ϕ to this distribution, derives all prices as marginals of P_ϕ, and updates ϕ after every trade. This transforms the market into an online inference system, where prices reflect a globally consistent and continuously learned probabilistic model.

We show that this design satisfies all three properties from Section 3, and that the complete-parlay variant enjoys an additional structural bonus: per-market loss halves with each new base asset added.

5.1 Architecture

The central challenge for a multi-market maker is keeping prices across all $N_M = 2^M - 1$ markets mutually consistent as information arrives through trade flow. A trade on any single market reveals something about the joint distribution, and a rational AMM should propagate this information to related markets immediately – otherwise prices become stale and the AMM is exposed to arbitrage. The ParlayMarket solves this through three interlocking mechanisms: *shadow trades* that broadcast information across markets, an *Ising model* belief state that compactly represents the joint distribution, and a *composite likelihood SGD* update that continuously calibrates the model from observed trade flow.

Shadow trades and price propagation. Prices in the ParlayMarket are posted marginals of a shared internal belief distribution P_φ over all M outcomes. When a trader bets on market S, the observed trade direction reveals information not just about event S but about all correlated events. Since we assume all incoming traders are informed (Section 3), the AMM should treat each trade as a signal about the entire joint distribution and update prices everywhere, not only in market S.

To implement this, the AMM records a set of internal *shadow trades* on related markets after each real trade arrives. Shadow trades are not real bets – they are bookkeeping entries that encode the information the AMM infers about other markets from the observed trade. Critically, they ensure that no market's price becomes stale simply because it receives little direct trading volume; even an esoteric parlay market that rarely attracts traders will see its posted price updated whenever a correlated market is active.

Ising model as the joint belief state. To propagate information through shadow trades, the AMM needs a tractable model of the joint distribution that can be updated incrementally. Parlay trades are especially informative here: a bet on a k-way parlay directly reveals a joint probability $P^*(\bigcap_{i\in S}\{X_i = 1\})$, which constrains not only the individual event probabilities but also their pairwise and higher-order correlations.

The Ising model is the natural parametric choice. Given a target set of marginal probabilities $\{p_i^*\}$ and pairwise correlations $\{p_{ij}^*\}$, the Ising model is the *maximum-entropy* distribution consistent with those first and second-order moments:

$$P_{\boldsymbol{\varphi}}(\mathbf{x}) \;\propto\; \exp\left(\sum_{i=1}^{M}\theta_i x_i + \sum_{i<j} W_{ij}\, x_i x_j\right), \tag{11}$$

where $\boldsymbol{\varphi} = (\boldsymbol{\theta}, \mathbf{W})$ collects bias parameters $\theta_i \in \mathbb{R}$ (controlling individual event probabilities) and interaction weights $W_{ij} \in \mathbb{R}$ (controlling pairwise correlations). Choosing the maximum-entropy distribution is equivalent to making the weakest possible assumptions beyond the moment constraints: no spurious higher-order structure is imposed that is not supported by the data. This also corresponds to a moment-matching approach to Bayesian updating – rather than tracking the full 2^M-dimensional joint distribution, the AMM only needs to maintain the $\binom{M}{2} + M$ sufficient statistics $(\mathbb{E}[X_i], \mathbb{E}[X_i X_j])$ of the Ising exponential family.

Posted prices are the Ising marginals computed via belief propagation: $p_i^{\boldsymbol{\varphi}} = P_{\boldsymbol{\varphi}}(X_i = 1)$ for base markets and $p_S^{\boldsymbol{\varphi}} = P_{\boldsymbol{\varphi}}\big(\bigcap_{i\in S}\{X_i = 1\}\big)$ for a parlay $S \subseteq [M]$. All N_M prices are generated from the single shared parameter vector $\boldsymbol{\varphi}$, so they are guaranteed to be globally consistent with some joint distribution at every point in time.

Gradient updates from trade errors. Each trade on market m reveals a discrepancy between the AMM's current posted price $p_m^{\boldsymbol{\varphi}}$ and the true probability $p_m^* = P^*(\mathcal{E}_m)$ implied by the trader's signal. The AMM treats this discrepancy as a cross-entropy error and takes a stochastic gradient step to reduce it. Formally, the AMM implicitly minimises the *composite pseudo-likelihood* [4, 18]:

$$\mathcal{L}(\boldsymbol{\varphi}) \;=\; \sum_{i=1}^{M}\lambda_i\,\mathrm{CE}(p_i^*, p_i^{\boldsymbol{\varphi}}) \;+\; \sum_{i<j}\lambda_{ij}\,\mathrm{CE}(p_{ij}^*, p_{ij}^{\boldsymbol{\varphi}}), \tag{12}$$

where $\lambda_i, \lambda_{ij} \geq 0$ are normalised trade arrival rates and $\mathrm{CE}(p, q) = -p\log q - (1-p)\log(1-q)$. Upon each trade on market m, the AMM applies the SGD update

$$\boldsymbol{\varphi}_{t+1} \;=\; \boldsymbol{\varphi}_t - \eta\,\nabla_{\boldsymbol{\varphi}}\,\mathrm{CE}(p_m^*, p_m^{\boldsymbol{\varphi}_t}), \tag{13}$$

with step size $\eta > 0$. Because $\boldsymbol{\varphi}$ is shared across all markets, a gradient step triggered by a trade on market m immediately shifts the posted prices of every other market through the Ising model's interaction terms. Shadow trades are then computed using the updated $\boldsymbol{\varphi}_{t+1}$, so the propagated price changes are themselves grounded in the newly calibrated belief state. This cycle – real trade → gradient update → shadow trades – is the core feedback loop of the ParlayMarket.

Relation to Properties 1–3. The three mechanisms above address our desiderata from Section 3 in turn. We now give an informal preview of the reasons behind this, and then prove formal results and empirical justification in the succeeding sections.

Property 1 (Liquidity consistency) is satisfied because all N_M posted prices flow from a single shared $\boldsymbol{\varphi}$, there is no separate model or order book per market. More importantly, shadow trades mean the AMM does not need direct trading activity in a market to keep its price current: every real trade on any correlated market triggers updates that ripple through $\boldsymbol{\varphi}$ and shift all posted prices simultaneously. A rarely-traded parlay market is not forgotten; it benefits from every trade on its correlated counterparts. Complete proof is deferred to appendix of the full version.

Property 2 (Bounded loss) follows from the balance that shadow trades create between gradient signal and gradient noise. Each real trade adds one update to $\boldsymbol{\varphi}$, but the shadow trades it spawns spread that update's effect across all related markets. As the number of markets grows, both the number of updates per round and the noise in each update grow together and cancel, keeping the steady-state parameter error, and therefore the AMM's total monetary loss per round, bounded independently of M. Without shadow trades this cancellation breaks, and loss grows exponentially with M.

Property 3 (Information aggregation) follows from the structure of the cross-entropy objective. The composite pseudo-likelihood $\mathcal{L}(\boldsymbol{\varphi})$ has a unique minimum at the parameters $\boldsymbol{\varphi}^*$ whose implied moments match those of the true distribution P^*. Because each trade gradient pushes $\boldsymbol{\varphi}$ toward this minimum, the AMM's posted prices converge to those of the best Ising approximation to P^* as trade flow accumulates, and with it, the posted joint probabilities approach the true joint distribution.

5.2 Convergence to the True Distribution

We first show how Property 3 is satisfied by focusing on how well the Ising model learns the joint distribution via the incoming trades. Our main result concerns the posted prices of the ParlayMarket – they converge geometrically to the best Ising approximation of P^*. After $O(1/(\eta\mu))$ informative trades the KL divergence between the true joint distribution and the AMM's belief shrinks to the model-misspecification floor, and all N_M posted prices stabilize at their moment-matched values.

▶ **Theorem 1** (Geometric convergence). *In the pure arbitrageur environment, starting from* $\boldsymbol{\varphi}_0 = 0$ *with step size* $\eta \le 1/(2L)$*:*

$$\mathbb{E}\|\boldsymbol{\varphi}_t - \boldsymbol{\varphi}^*\|^2 \;\le\; (1-2\eta\mu)^t \,\|\boldsymbol{\varphi}^*\|^2 \;+\; \frac{\eta\sigma^2}{2\mu},$$

where $\sigma^2 = \mathbb{E}\|\mathbf{g}_t(\boldsymbol{\varphi}^*)\|^2$ *is the gradient variance at the optimum. With* $\eta = 0.2$ *and* $\mu \approx \frac{1}{4}$*, the contraction factor per round is* 0.9*; roughly* 22 *rounds suffice to reduce the initial error by* $10\times$*.*

Three results underpin this guarantee.

Moment-matched target. The fixed point $\boldsymbol{\varphi}^*$ is the unique parameter vector satisfying $p_i^{\boldsymbol{\varphi}^*} = p_i^*$ for all i and $p_{ij}^{\boldsymbol{\varphi}^*} = p_{ij}^*$ for all $i < j$. This follows directly from the update rule: at $\boldsymbol{\varphi}^*$ the per-trade scalar $\lambda_m = (p_m^{\boldsymbol{\varphi}^*} - p_m^*)/(1 - p_m^{\boldsymbol{\varphi}^*})$ vanishes for every market m, so all gradient steps are zero. Equivalently, $\boldsymbol{\varphi}^*$ is the I-projection of P^* onto the Ising family – the maximum-entropy distribution whose first and second moments match those of P^* [25]. Once $\boldsymbol{\varphi}_t \to \boldsymbol{\varphi}^*$, the posted joint prices $p_S^{\boldsymbol{\varphi}} = P_{\boldsymbol{\varphi}}(\bigcap_{i \in S}\{X_i = 1\})$ converge to the best Ising-family approximation of P^*, giving $D_{\mathrm{KL}}(P^* \| P_{\boldsymbol{\varphi}_t}) \to 0$ up to the misspecification gap from higher-order dependence not captured by the pairwise family.

Loss curvature. The contraction rate $(1 - 2\eta\mu)$ is determined by the strong-convexity constant μ of the composite pseudo-likelihood $\mathcal{L}$.

▶ **Proposition 2** (Strong convexity). *On the compact ball $\Phi_B = \{\boldsymbol{\varphi} : \|\boldsymbol{\varphi}\| \le B\}$, $\mathcal{L}$ is μ-strongly convex. At the uninformative initialisation $\boldsymbol{\varphi} = 0$, the Hessian is diagonal with $\mu = \frac{3}{16}$ (interaction directions) and $\Lambda = \frac{1}{4}$ (bias directions), giving condition number $\kappa = \Lambda/\mu = \frac{4}{3}$. All eigenvalues are $O(1)$, independent of M.*

Each CE term in $\mathcal{L}$ contributes a positive-semidefinite rank-one block

$$H_m = \frac{(\nabla_{\boldsymbol{\varphi}} p_m^{\boldsymbol{\varphi}})(\nabla_{\boldsymbol{\varphi}} p_m^{\boldsymbol{\varphi}})^\top}{p_m^*(1 - p_m^*)}$$

to the Hessian via the covariance identity. At $\boldsymbol{\varphi} = 0$ the Ising model is the uniform product measure, so cross-terms vanish and the Fisher matrix is diagonal with entries $\frac{1}{4}$ (bias parameters) and $\frac{3}{16}$ (interaction parameters) – all $O(1)$ regardless of M. Adding more base markets adds more CE terms and therefore more curvature, but the per-parameter curvature stays bounded, which is why the convergence rate does not degrade as M grows.

Parlay acceleration. Higher-order parlay trades strictly improve the convergence rate by adding curvature in the interaction directions that base-market trades cannot supply alone. When the AMM accepts k-way parlay bets for $k \ge 3$, each parlay-S trade contributes a further PSD block H_S to the Hessian, raising the effective strong-convexity constant to

$$\mu_{\mathrm{eff}} \;\ge\; \mu + \sum_{k \ge 3} \sum_{S \in \mathcal{S}_k} \lambda_S \, \mu_S, \qquad \mu_S = \lambda_{\min}(H_S) \ge 0, \tag{14}$$

with strict inequality whenever parlay S is informative about $\boldsymbol{\varphi}$. The contraction factor improves from $(1 - 2\eta\mu)$ to $(1 - 2\eta\mu_{\mathrm{eff}})$, and the cumulative regret tightens to $G^2/(2\mu_{\mathrm{eff}})(1 + \log T) = O(\log T)$. The gain is largest for the interaction weights W_{kl}: a two-way parlay $\{k, l\}$ contributes only $O(\rho^2)$ to the (W_{kl}, W_{kl}) curvature entry via base-price cross-effects, whereas a higher-order parlay $S \ni k, l$ contributes $O(1)$ directly – a factor of $O(\rho^{-2})$ improvement at small correlations ρ. Proofs are provided in Appendix C of the full version.

5.3 Loss Bounds

The previous subsection established Property 3 by showing that the shared parameter vector $\boldsymbol{\varphi}_t$ converges to the moment-matched target $\boldsymbol{\varphi}^*$. We now turn to Property 2. The question here is not whether the AMM learns the correct joint distribution, but how the remaining parameter error translates into monetary loss. The bridge is a quadratic sensitivity bound: near the fixed point, LMSR loss is second-order in pricing error, and pricing error is first-order in parameter error. In the complete-parlay regime, shadow trades then pool information across all

$$N_M = 2^M - 1$$

active contracts, keeping the *aggregate* steady-state loss per round bounded even though the number of quoted markets grows exponentially.

▶ **Proposition 3** (Complete-parlay loss scaling). *In the complete-parlay regime, where every non-empty subset $S \subseteq [M]$ is traded at equal rate and the SGD step size is η, the aggregate expected LMSR loss admits the decomposition*

$$\mathbb{E}[L_T] \;\le\; L_{\text{transient}} + T\, L_{\text{ss,round}},$$

with

$$L_{\text{transient}} = O\left(\frac{b\rho^2 M^2}{\eta}\right), \qquad L_{\text{ss,round}} = O(b\eta).$$

In particular, the dependence on M is polynomial rather than exponential, so the complete-parlay Parlay AMM satisfies the bounded-loss requirement of Property 2.

The proposition should be read as a two-phase decomposition. The first term is a one-time learning cost incurred while the AMM moves from the uninformative initialization toward $\boldsymbol{\varphi}^*$. The second is the steady-state cost of continuing to learn from a stochastic stream of informative trades once the model has essentially converged.

Loss–parameter link. We first convert the parameter-space error from Theorem 1 into a bound on monetary loss.

▶ **Lemma 4** (Quadratic sensitivity). *For a market S with true probability p_S^* and AMM parameters $\boldsymbol{\varphi}$, the expected LMSR monetary loss from one informed trade satisfies*

$$b\,\mathrm{KL}(p_S^* \| p_S^{\boldsymbol{\varphi}}) \;\le\; \frac{b\,\Lambda_S}{2}\,\|\boldsymbol{\varphi} - \boldsymbol{\varphi}^*\|^2, \qquad \Lambda_S := \sup_{\boldsymbol{\varphi}} \frac{\|\nabla_{\boldsymbol{\varphi}} p_S^{\boldsymbol{\varphi}}\|^2}{p_S^*(1-p_S^*)}.$$

The lemma is the composition of two local facts. First, for p near p^*,

$$\mathrm{KL}(p^* \| p) = \frac{(p-p^*)^2}{2p^*(1-p^*)} + O(|p-p^*|^3).$$

Second, the price error is linear in the parameter error:

$$p_S^{\boldsymbol{\varphi}} - p_S^* = \nabla_{\boldsymbol{\varphi}} p_S^{\boldsymbol{\varphi}^*} \cdot (\boldsymbol{\varphi} - \boldsymbol{\varphi}^*) + O(\|\boldsymbol{\varphi} - \boldsymbol{\varphi}^*\|^2).$$

Combining the two shows that monetary loss is quadratic in parameter error. This is the key bridge from the convergence theory to Property 2.

Signal–noise balance in the complete-parlay regime. In the formal interaction model, one market is selected uniformly at random each round. Over a block of N_M rounds, each of the $N_M = 2^M - 1$ markets is therefore visited once *in expectation*. The cumulative local information gathered over such a block is summarized by the aggregate sensitivity matrix

$$\Sigma_{\text{cycle}} := \sum_{S \neq \emptyset} F_S(\boldsymbol{\varphi}^*), \qquad F_S(\boldsymbol{\varphi}^*) := \frac{(\nabla_{\boldsymbol{\varphi}} p_S^*)(\nabla_{\boldsymbol{\varphi}} p_S^*)^\top}{p_S^*(1-p_S^*)}.$$

Each term $F_S(\boldsymbol{\varphi}^*)$ measures how informative market S is about the shared parameter vector near the truth: markets whose prices respond strongly to perturbations of $\boldsymbol{\varphi}$ contribute more to learning. Because all markets update the same shared state $\boldsymbol{\varphi}$, both the mean

restoring drift toward φ^* and the stochastic gradient noise accumulate through this same pooled matrix Σ_{cycle}. Put differently, adding more markets increases signal and noise at the same rate rather than letting noise dominate. As a result, the steady-state parameter error remains $O(\eta)$ rather than growing with N_M. Applying Lemma 4 then gives

$$L_{\text{ss,round}} = O(b\eta).$$

Why shadow trades matter. This favorable scaling is not automatic: it relies on the fact that a trade in one market updates the shared parameter vector and therefore refreshes all related quotes. That cross-market propagation is exactly what shadow trades provide. A trade in market S does not only improve the quote for S; it also carries information about overlapping base and parlay contracts through the common state φ. Without this mechanism, each additional market would still add its own stochastic noise, but the corresponding information would remain largely confined to the market that was directly traded. The system would then behave more like a collection of weakly connected books than a single pooled market maker, and the uniform aggregate bound in Proposition 3 would no longer be expected to hold.

As an immediate corollary, once the transient learning phase has passed,

$$\bar{\ell}_M \;\approx\; \frac{L_{\text{ss,round}}}{N_M} = O\!\left(\frac{b\eta}{2^M}\right),$$

so the *mean loss per listed market* decays exponentially with M. We state this as a corollary rather than part of Proposition 3 because the design objective in Section 3 is bounded *aggregate* loss; the halving pattern in Figure 3 is the complete-parlay consequence of that aggregate bound.

5.4 Information-theoretic optimality

The quadratic loss scaling in Proposition 3 is worst-case optimal. To see why, consider the pairwise family

$$P_s(x) \propto \exp\!\Big(\gamma \sum_{i<j} s_{ij} x_i x_j\Big), \qquad s_{ij} \in \{-1, +1\}.$$

This family contains one independent hidden correlation sign for every pair of events, hence $\binom{M}{2} = \Theta(M^2)$ independent pairwise facts. Any AMM that does not know these signs in advance must learn them from order flow in order to price the corresponding two-leg parlays correctly.

For a fixed pair (i, j), the two possible signs $s_{ij} = +1$ and $s_{ij} = -1$ imply different fair prices for the parlay $X_i X_j$. Until the sign is learned, any AMM quoting this parlay with liquidity scale b must be wrong on one of the two cases, and an informed trader can extract loss from that mispricing. For constant-strength correlations this loss is $\Omega(b)$ per unresolved pair; for weak correlations of size γ it is $\Omega(b\gamma^2)$.

Since there are $\Theta(M^2)$ independent pairwise signs, even the best AMM in this class has worst-case information-acquisition loss

$$\inf_A \sup_s \mathbb{E}_{P_s}[L_T] \;\geq\; \Omega\!\left(b\binom{M}{2}\right) = \Omega(bM^2),$$

for constant-strength correlations, and $\Omega(b\gamma^2 M^2)$ in the weak-correlation regime. Thus the quadratic loss rate is not an artifact of the Ising update rule: it is the unavoidable cost of learning dense pairwise correlation structure from trades. *ParlayMarket* matches this rate up to constants while avoiding the exponential loss of treating each parlay as an independent market.

5.5 Parlay Trades as essential for learning

The previous subsection used Theorem 1 only as an input to the loss bound. We now return to the learning dynamics themselves. Higher-order parlays are not merely extra contracts to quote; they also sharpen the learning problem by adding curvature to the shared objective. This strengthens Property 3 directly, and through Lemma 4 it improves the constants in Property 2 as well.

▶ **Proposition 5** (Parlay-accelerated convergence). *Suppose, in addition to base and pairwise markets, the AMM receives trades on higher-order parlays $S \subseteq [M]$ with $|S| \geq 3$ at rates $\{\lambda_S\}$. Let $\mathcal{L}^+$ denote the augmented composite loss including these parlay terms, and write*

$$\mathcal{S}_k := \{S \subseteq [M] : |S| = k\}.$$

Then:

1. ***Improved local curvature.*** *In a neighborhood of $\boldsymbol{\varphi}^*$, the augmented loss has local strong-convexity constant μ_{eff} satisfying*

$$\mu_{\text{eff}} \;\geq\; \mu + \sum_{k\geq 3} \sum_{S\in\mathcal{S}_k} \lambda_S \, \mu_S, \qquad \mu_S \geq 0,$$

with strict inequality whenever at least one traded higher-order parlay is informative about $\boldsymbol{\varphi}$.

2. ***Faster parameter convergence.*** *Under the same SGD dynamics* (13) *with step size $\eta \leq 1/(2L)$,*

$$\mathbb{E}\|\boldsymbol{\varphi}_t - \boldsymbol{\varphi}^*\|^2 \;\leq\; (1 - 2\eta\mu_{\text{eff}})^t \, \|\boldsymbol{\varphi}^*\|^2 \;+\; \frac{\eta\sigma^2}{2\mu_{\text{eff}}},$$

yielding strictly faster geometric contraction than under the base-and-pairwise objective alone.

3. ***Improved average error and loss.*** *Let*

$$\bar{\varepsilon}_T^2 := \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\|\boldsymbol{\varphi}_t - \boldsymbol{\varphi}^*\|^2.$$

Then

$$\bar{\varepsilon}_T^2 \;\leq\; O\!\left(\frac{\|\boldsymbol{\varphi}^*\|^2}{\eta\mu_{\text{eff}} T} + \eta\right).$$

Consequently, the average LMSR loss per round is $O(b\,\bar{\varepsilon}_T^2)$ and the cumulative excess loss over T rounds is $O(bT\,\bar{\varepsilon}_T^2)$, both with strictly improved constants.

Intuition: curvature as information aggregation. Each trade provides a stochastic gradient whose strength depends on how informative the observed event is about the parameters. Base-market trades mainly constrain first moments, and pairwise trades constrain interactions only one edge at a time. By contrast, a k-way parlay directly reveals a joint probability and therefore aligns with several interaction directions simultaneously. At the level of the Hessian, each informative parlay adds a positive semidefinite block, strengthening the restoring force toward $\boldsymbol{\varphi}^*$.

Interaction-level acceleration. The gain is largest for interaction weights W_{ij}. In weak-correlation regimes, these directions are the hardest to learn from singleton flow alone, because the relevant gradients appear only through small cross-effects. Higher-order parlays that include both i and j constrain several interaction directions at once and can therefore dominate the curvature in the weakly identified subspace; heuristically, the improvement can be of order $O(\rho^{-2})$ when $\rho \ll 1$.

From faster learning to lower loss. Proposition 5 should be read as the mechanism behind the favorable loss scaling in Proposition 3. By Lemma 4, monetary loss is quadratic in parameter error, so any increase in curvature reduces both the transient learning cost and the steady-state error floor. Parlays are therefore not only additional products to quote; they are an additional source of statistical signal that sharpens the learning dynamics of the market maker itself.

6 Evaluation

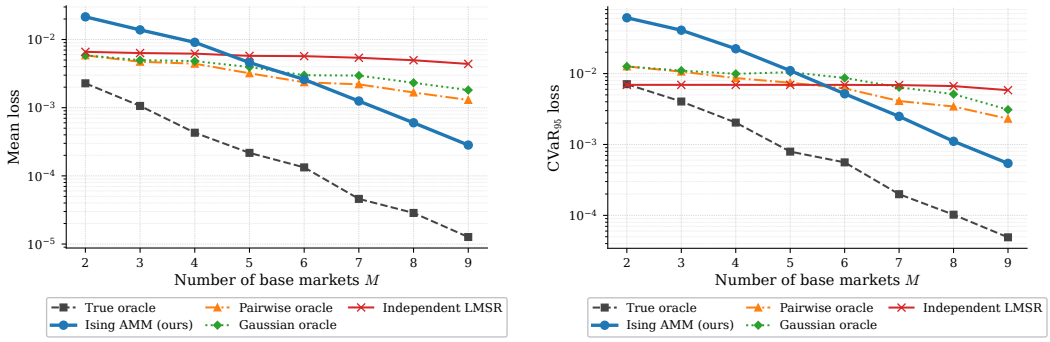
We evaluate ParlayMarket in both controlled and market-based settings. The synthetic experiments test the theoretical predictions of Section 5: bounded aggregate loss, geometric convergence, and the role of parlay order flow in accelerating learning. We then study a historical-market replay on Kalshi data to evaluate whether the same advantages persist under observed prices, RFQ flow, and realistic implementation constraints.

6.1 Complete-Parlay Loss Scaling

Setup. We simulate $M \in \{4, 5, 6, 7, 8, 9\}$ correlated binary assets drawn from a jointly Gaussian scoring model with pairwise correlation $\rho = 0.3$. At each round, one of the $N_M = 2^M - 1$ non-empty parlay markets is selected uniformly at random and an informed trader submits a bet of size b. The AMM updates its Ising parameters via SGD with step sizes $\eta_h = \eta_J = 0.2$. All results are averaged over 100 independent simulation runs.

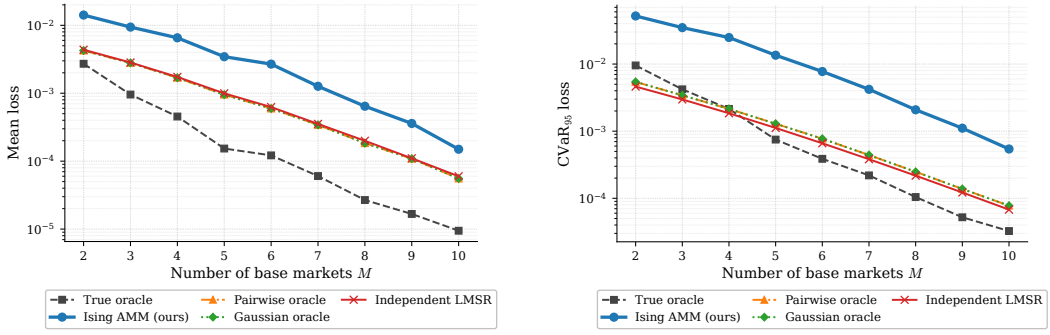
Baselines. We compare the Correlation AMM against four baselines. The *true oracle* knows P^* exactly and serves as a lower bound on achievable loss. The *independent LMSR* maintains a separate LMSR per market, ignoring correlations entirely. The *pairwise oracle* knows the true pairwise marginals p^*_{ij} analytically and fits an Ising MRF to these moments, but does not learn from trade flow. The *Gaussian oracle* uses the true Gaussian copula parameters to price joint events via multivariate normal CDF; it captures higher-order structure exactly under the Gaussian model but cannot update from trades. All oracle baselines are given their parameters for free and serve to bracket the loss achievable with varying degrees of distributional knowledge.

Per-market loss decay. Figure 3 shows the mean per-market LMSR monetary loss ℓ_M as a function of M. Consistent with Proposition 3, ℓ_M decays exponentially as $C \cdot 2^{-M}$, halving with each additional base asset. The product $\ell_M \cdot 2^M \approx 0.15$ is approximately constant for $M = 4, \ldots, 9$ (See Table 3 in Appendix A of the full version), confirming that the total per-round loss $N_M \cdot \ell_M$ is essentially flat – consistent with the steady-state dominance predicted by the two-phase decomposition.



(a) Mean LMSR loss per round per market $\bar{\ell}_M$. **(b)** CVaR_{95} loss per round per market.

Figure 3 The ParlayMarket is the only non-oracle model whose loss decays exponentially with the number of base markets M, in the presence of parlays; all other non-oracle models plateau or grow.



(a) Mean LMSR loss per round per market $\bar{\ell}_M$. **(b)** CVaR_{95} loss per round per market.

Figure 4 The ParlayMarket fails to perform better than even independent AMMs in the absence of parlay markets.

Price convergence. Figure 6 in Appendix A of the full version shows the mean absolute error (MAE) between the AMM's posted prices and the true probabilities over time, averaged across all N_M markets and all simulation runs, with $\pm 1\sigma$ shading. The top panel shows the LMSR market-price MAE; the bottom panel shows the Ising marginal belief error for the Ising model. Both converge monotonically; the Ising belief error converges faster because belief propagation directly optimises the Ising objective, while the market-price error includes the additional LMSR rounding effect.

Parlay acceleration. Consistent with Proposition 5, the advantage of the ParlayMarket is entirely contingent on the presence of parlay order flow. Figure 3 shows that with complete parlays the Ising AMM achieves the lowest non-oracle loss, decaying exponentially with M. Figure 4 shows the complementary picture: when informed traders are restricted to base markets only, the model ranking completely reverses.

6.2 Importance of Parlays for learning

Setup. We run an ablation in which informed traders are restricted to the M base markets only – no parlay bets are placed – while all AMMs still maintain and price all $N_M = 2^M - 1$ markets. Figures 3 and 4 show mean loss per round per market vs. M for both conditions.

The model ranking reverses completely. With complete parlays (Figure 3), the ParlayMarket is the *best-performing* non-oracle model: it achieves the lowest per-round loss of all non-oracle AMMs, sitting just above the true oracle and well below every other model. The independent LMSR, pairwise oracle, and Gaussian oracle all incur substantially higher and exponentially growing losses as M increases, reaching ≈ 2.24, ≈ 0.66, and ≈ 0.93 per round respectively at $M = 9$.

Without parlays (Figure 4), the ranking *completely reverses*: the ParlayMarket now has the *highest* loss of all non-oracle models, while the independent LMSR, pairwise oracle, and Gaussian oracle cluster together at much lower values (≈ 0.02–0.06 per round across all tested M). The y-axis scale tells the story: the no-parlay plot peaks at ≈ 0.18, while the with-parlay plot requires a scale of 2.3 – a 13× difference – just to accommodate the losses of the non-Ising models.

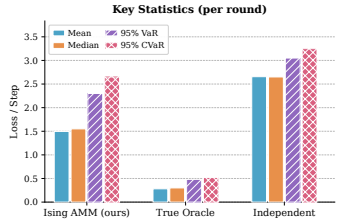
Why the ranking reverses. The reversal reveals precisely what parlays provide: the *gradient signal needed to calibrate the Ising joint belief model*. Without parlays, the ParlayMarket receives gradient updates only from the M base markets. This is insufficient to pin down the $\binom{M}{2}$ interaction parameters W_{ij} in any reasonable simulation length: the base-market signal is under-determined relative to the full Ising parameter space, so the model misprices joint events and incurs high loss. Simpler models – the pairwise oracle (which knows p^*_{ij} analytically) and the independent LMSR (which only needs to price M uncorrelated events) – do not face this calibration problem and therefore achieve low loss without parlays. With parlays, the situation reverses. Each parlay bet supplies a direct gradient signal on a specific joint event, giving the Ising model the rich information it needs to calibrate $(\boldsymbol{\theta}, \mathbf{W})$ from all N_M markets simultaneously (Proposition 5 and Figure 3). Once calibrated, belief propagation ensures that all N_M posted prices are globally consistent with the learned joint distribution, so adversaries cannot exploit mispriced correlations. The independent LMSR cannot use this information at all (it ignores correlations by design), and the pairwise and Gaussian oracles only capture lower-order structure – leaving them vulnerable to adversaries who trade higher-order parlays against their mispriced joint events.

Implication. Parlays are not merely an efficiency accelerant: they are the *primary information channel* through which the ParlayMarket learns the joint distribution. Without them, the Ising model is the worst non-oracle performer; with them, it is the best. No other model tested achieves this reversal, confirming that the ParlayMarket and parlay markets are jointly necessary – neither is sufficient alone.

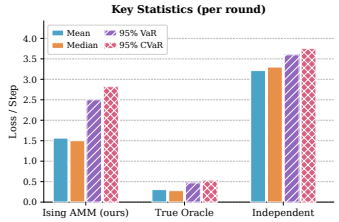
6.3 Robustness Under Noisy Informed Traders

Setup. We repeat the complete-parlay simulation with the same parameters but vary the noise-trader fraction $\alpha \in [0, 1]$: with probability α the arriving trader submits a uniformly random price signal, and with probability $1 - \alpha$ they are fully informed. Results at $M = 9$ are extrapolated from direct simulations at $M \in \{3, 4, 5\}$ using the clean $M = 5 \to 9$ loss scaling.

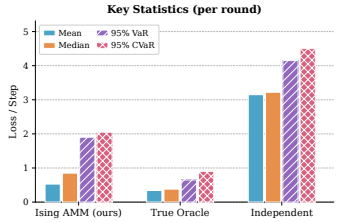
Results. Figure 7 in Appendix A of the full version shows the per-market LMSR loss as a function of noise-trader fraction α for $M = 9$. The ParlayMarket's loss grows gracefully with α – it remains the best-performing non-oracle model across all noise levels tested – because the noisy gradient signal inflates the steady-state error while leaving the convergence rate largely unchanged. All models cross the break-even line only at high noise fractions, demonstrating robustness to realistic noise levels.



(a) Five markets with $[2, 3, 4, 4, 5]$ outcomes.



(b) Eight markets with $[2, 3, 3, 4, 4, 5, 6, 7]$ outcomes.



(c) Eight markets with $[2, 3, 3, 4, 4, 5, 6, 7]$ outcomes, and random jumps.

Figure 5 Key per-round statistics (Loss / Step) for three multinomial market configurations. In all cases the Ising AMM (ours) sits between the True Oracle lower bound and the Independent baseline, with the largest relative gain visible in the tail-risk measures (95% VaR and CVaR).

6.4 Extension to Multinomial Markets

Setup. All experiments above assume binary outcomes per base asset. We now evaluate whether the Ising AMM generalises to *multinomial* markets, where each base asset can resolve to one of $K > 2$ outcomes. We run three configurations that vary the outcome structure and the nature of price jumps (e.g. smooth updates vs. discontinuous jumps between outcome clusters). In each case, we compare the Ising AMM against a *True Oracle* baseline (knows the true joint distribution exactly) and an *Independent* baseline (maintains a separate market maker per outcome, ignoring all correlations). The underlying trader model for these results is a straightforward generalization of the Gaussian scoring model. Details of the multinomial model are provided in Appendix D of the full version.

Results. Figure 5 reports the key per-round statistics (mean, median, 95% VaR, and 95% CVaR of Loss / Step) across the three configurations. In every scenario the Ising AMM sits strictly between the True Oracle lower bound and the Independent baseline: it captures a large fraction of the correlation structure available to the oracle while outperforming the disjoint-market-maker strategy on all four statistics. The gap between Ising AMM and Independent is particularly pronounced in the tail-risk metrics (95% VaR and CVaR), indicating that joint modelling not only reduces average loss but also curtails the worst-case exposure that arises when correlated outcomes are priced independently. These results demonstrate that the Ising parameterisation and the parlay-driven gradient signal are not artefacts of the binary setting. The mechanism extends naturally to multinomial markets and retains its loss advantage over disjoint market makers even as the outcome space grows.

6.5 Empirical Market Performance

To evaluate whether these advantages survive outside the synthetic setting, we replay the mechanism on historical Kalshi data. The empirical study focuses on three implementation details that are central to the practicality of the design.

Two execution settings. We consider *two execution settings*. In a *full-market emulation*, base markets are modeled as LMSRs under the AMM, and the informative signal extracted from each trade is its *target price*: an informed trader is assumed to move the market to the price implied by her information. In a *standalone market-maker emulation*, base markets remain external and the AMM is used only as a shared quoting and underwriting engine for

parlays; in that case the relevant signal is *order flow* – trade direction and share quantity – rather than the final market price. These two settings correspond to two distinct practical roles of the mechanism: a complete market design and a drop-in correlation-aware parlay market maker layered on top of an existing RFQ-style platform.

Real-market emulation. The replay is performed on a *real historical universe*. We use Kalshi's NBA slate for 2026-03-07, comprising 732 base markets across six games and 5,918 listed combo tickers with legs entirely inside this universe, spanning 2- through 10-leg parlays. To scale inference to clusters of this size, we train one Ising model per game cluster and use *loopy belief propagation* rather than exact enumeration. Each model is initialized from the first 30 candle mid-prices of its constituent legs, with fields initialized from median log-odds and couplings initialized at zero, after which the stream of candle updates and parlay trades is replayed chronologically. Candle updates train local fields, while parlay trades update both fields and interactions through the trade-implied target price. The replay therefore evaluates performance on historically accepted RFQ flow rather than on the full distribution of submitted RFQ requests.

Preprocessing. The empirical setup explicitly addresses a key market imperfection: *fee bias*. Observed Kalshi combo execution prices are ask quotes rather than direct probability assessments, so they are systematically upward biased by platform markup. Training directly on these prices would therefore bias the learned joint model upward. To correct for this, we preprocess each observed trade into a canonical all-YES representation using the complement rule and inclusion–exclusion; for mixed YES/NO parlays, the additive fee cancels in the alternating expansion, yielding an approximately unbiased target in the model's native parameterization. This correction is essential in practice: without it, parlay loss remains flatter and revenue is lower.

The full replay pipeline and implementation details are provided in Appendix E of the full version. Table 2 shows that standalone ParlayMarket market maker achieves the strongest risk-adjusted liquidity-provider portfolio, with the highest Sharpe ratio among the non-native strategies while still accepting a large fraction of historical flow. The full AMM accepts all trades but exhibits lower ROI, likely because the observed RFQ flow contains substantial noise that a fully internalized LMSR-style implementation must absorb directly. Taken together, these results show that the empirical value of ParlayMarket pricing is strongest in the *standalone market-maker* regime: the model can be inserted into an existing platform, improve LP portfolio quality, and remain competitive on accepted quotes.

6.6 Summary

Across both synthetic and historical evaluations, the results support the same conclusion. A shared, learned representation of the joint distribution allows ParlayMarket to quote a combinatorially large family of contracts while controlling aggregate loss, using parlay order flow as a direct signal for dependence learning. In synthetic settings this produces the predicted scaling and convergence behavior; in historical replay it yields a practically useful correlation-aware quoting engine with strong risk-adjusted performance that can be readily deployed.

Table 2 Emulation on the March 7, 2026 NBA slate under the market-competitive filter.

Strategy	Accepted Trades	MM Win Rate	Net PnL	Sharpe
Kalshi (execution price)	5436	84.5%	$244,190.77	0.7151
Standalone MM (θ_{corr})	3973	86.8%	$113,984.58	1.0049
Full AMM	5436	69.0%	$158,577.66	0.7062

7 Conclusion and limitations

In this paper, we showed that parlay trading can be incorporated into an automated market maker without requiring a separate pricing and risk-management process for every combination. The mechanism supports 2^M base and joint contracts using only an $O(M^2)$ shared representation, and correspondingly achieves $O(M^2)$ aggregate information-retrieval loss rather than loss that scales with the full combinatorial market space. This shared structure also allows parlay order flow to serve as a direct signal for learning dependence structure. The empirical results further suggest that the mechanism is not only theoretically well-founded but close to deployable in practice: historical-market replay on Kalshi data shows that, in the standalone market-maker setting, correlation-aware quoting remains competitive under observed prices, RFQ flow, and realistic execution constraints.

An important scope condition of the current framework is that the traded contracts are assumed to already be expressed in a structurally coherent binary form. In many practical settings, this representation must itself be constructed before correlation-aware pricing becomes meaningful. For example, families of mutually exclusive outcomes may satisfy constraints such as $\sum_{i=1}^{k} X_i = 1$, even though the underlying latent process remains stochastic. In such cases, a separate preprocessing or structural-learning layer is needed to map raw market data into variables on which the assumptions of the AMM hold. We illustrate one such preprocessing step in the evaluation section. This suggests a broader architecture for automated parlay markets: a structural layer that identifies and normalizes the relevant event representation, followed by a pricing layer such as ParlayMarket that provides automated execution and, as a secondary benefit, elicits dependence from the resulting trade flow.

References

1 Jacob Abernethy, Yiling Chen, and Jennifer Wortman Vaughan. Efficient market making via convex optimization. In *ACM EC*, 2013.

2 Jacob Abernethy, Sindhu Kutty, Sébastien Lahaie, and Rahul Sami. Information aggregation in exponential family markets. In *Proceedings of the 15th ACM Conference on Economics and Computation (EC)*, 2014.

3 Guillermo Angeris, Tarun Chitra, Alex Evans, and Stephen Boyd. Improved price oracles: Constant function market makers. *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, 2020. `arXiv:2003.10001`.

4 Julian Besag. Statistical analysis of non-lattice data. *The Statistician*, 24(3):179–195, 1975.

5 Guy Bresler. Efficiently learning ising models on arbitrary graphs. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, 2015. `doi:10.1145/2746539.2746631`.

6 Yiling Chen, Nicolas Lambert, David M. Pennock, and Jennifer Wortman Vaughan. Eliciting predictions and recommendations for decision making. *ACM Transactions on Embedded Computing Systems*, 9(4):39:1–39:21, 2010. URL: `https://www.microsoft.com/en-us/research/wp-content/uploads/2016/04/TEAC-final1.pdf`.

7 Yiling Chen and David M. Pennock. A utility framework for bounded-loss market makers. *Journal of Artificial Intelligence Research*, 2007.

8 Yiling Chen, David M. Pennock, Michael H. Bowling, and Nicolas Lambert. Price updating in combinatorial prediction markets with bayesian networks. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 2008. arXiv:1202.3756.

9 Free Systems. Building the truth machine. https://open.substack.com/pub/freesystems/p/building-the-truth-machine, 2024. Substack.

10 Rafael Frongillo, Ian Kash, and Mark D. Reid. Interpreting prediction markets: A stochastic approach. In *Advances in Neural Information Processing Systems*, 2012. URL: https://papers.nips.cc/paper/4529-interpreting-prediction-markets-a-stochastic-approach.

11 Robin Hanson. Logarithmic market scoring rules for modular combinatorial information aggregation. Technical report, George Mason University, 2002.

12 Robin Hanson. Combinatorial information market design. *Information Systems Frontiers*, 5(1):107–119, 2003. doi:10.1023/A:1022058209073.

13 Robin Hanson. Logarithmic market scoring rules for modular combinatorial information aggregation. *Journal of Prediction Markets*, 2007.

14 Elad Hazan. *Introduction to Online Convex Optimization*. Now Publishers, 2019. arXiv:1909.05207.

15 Sharif Hossain, Jenn Wortman Vaughan, Yuan Wang, and Yiling Chen. Designing automated market makers for combinatorial securities: A geometric viewpoint. In *Proceedings of the 13th ACM Conference on Economics and Computation (EC)*, 2012.

16 Kalshi. Kalshi: Event contracts exchange. https://kalshi.com, 2024. Accessed: 2026-03-19.

17 Kathryn Blackmond Laskey, Tai Sing Lee, Robin Hanson, Charles Twardy, Shun-Yang Matsumoto, and Brandon Goldfedder. Graphical model market maker for combinatorial prediction markets. In *Proceedings of the AAAI Workshop on Trading Agent Design and Analysis*, 2012.

18 Bruce G. Lindsay. Composite likelihood methods. In N. U. Prabhu, editor, *Statistical Inference from Stochastic Processes*, volume 80 of *Contemporary Mathematics*, pages 221–239. American Mathematical Society, Providence, RI, 1988.

19 Jason Milionis, Ciamac C. Moallemi, Tim Roughgarden, and Anthony Lee Zhang. Automated market making and loss-versus-rebalancing. *arXiv preprint*, 2022. arXiv:2208.06046.

20 Abraham Othman and Tuomas Sandholm. When do markets with simple traders fail? In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2010. URL: https://www.ifaamas.org/Proceedings/aamas2010/pdf/01%20Full%20Papers/18_02_FP_0266.pdf.

21 Polymarket. Polymarket: A decentralized information market platform. https://polymarket.com, 2024. Accessed: 2026-03-19.

22 Pradeep Ravikumar, Martin J. Wainwright, and John D. Lafferty. High-dimensional ising model selection using ℓ_1-regularized logistic regression. *The Annals of Statistics*, 38(3):1287–1319, 2010. doi:10.1214/09-AOS691.

23 Dan Robinson and Ciamac C. Moallemi. pm-amm. https://www.paradigm.xyz/2024/11/pm-amm, 2024.

24 Marc Vuffray, Sidhant Misra, Andrey Y. Lokhov, and Michael Chertkov. Interaction screening: Efficient and sample-optimal learning of ising models. In *Advances in Neural Information Processing Systems*, 2016. arXiv:1605.07252.

25 Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008. doi:10.1561/2200000001.

26 Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, 2003. URL: https://www.cs.cmu.edu/~maz/publications/techconvex.pdf.

On Sybil-Proof Mechanisms

Minghao Pan
California Institute of Technology, Pasadena, CA, USA
Flashbots, Pasadena, CA, USA

Bruno Mazorra
Flashbots, Barcelona, Spain

Christoph Schlegel
Flashbots, Zurich, Switzerland

Akaki Mamageishvili
Offchain Labs, Zurich, Switzerland

Abstract

We show that in the single-parameter mechanism design environment, the only non-wasteful, symmetric, incentive compatible and Sybil-proof direct mechanism is a second price auction with symmetric tie-breaking. Thus, if there is private information, lotteries or other mechanisms that do not always allocate to a highest-value bidder are not Sybil-proof or not incentive compatible. Moreover, we show that our main (im)possibility result extends beyond linear valuations, but not to multi-unit object allocation with unit-demand bidders. We also provide examples of mechanisms (with higher interim payoff for the bidders than a second price auction) that satisfy all of the other axioms and a weaker, Bayesian notion of Sybil-proofness. Thus, our (im)possibility result does not generalise to the Bayesian setting and we have a larger design space: With Sybil constraints, equivalence between dominant strategy and Bayesian implementation (that holds in classical single-parameter mechanism design without Sybils) no longer holds.

2012 ACM Subject Classification Theory of computation → Algorithmic game theory and mechanism design

Keywords and phrases Sybils, Mechanism Design, Blockchain

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.16

Category Extended Abstract

Related Version *Full Version*: `https://arxiv.org/abs/2407.14485`

1 Introduction

Where creating new identities is cheap and verifying them is hard, mechanism design must consider that a participant may enter a mechanism under several identities. Such a manipulation is called a Sybil attack, and a mechanism immune to it *Sybil-proof*. Blockchain systems, where participation is permissionless, are both rich in deployed mechanisms and susceptible to Sybil attacks.

When allocating a private good, a natural Sybil-proof mechanism is an auction that assigns the good to a highest-value bidder. Auctions may, however, be undesirable if fairness, wider participation, or other distributional goals matter. Event tickets, for example, are often sold below the market-clearing price, making the primary sale a target for middlemen who, if quantities per buyer are capped, buy through Sybils [1]. In blockchain systems, randomness in allocating block proposal rights is desirable to avoid power concentration, and proportional selection has been characterised there by Sybil-proofness, symmetry, non-wastefulness and collusion-proofness [2, 4].

It is therefore natural to ask whether there are Sybil-proof mechanisms for private goods other than auctions. Theorem 1 gives a strong negative answer: in the classical Myersonian setting, Sybil-proofness collapses a large design space of monotonic rules to a single one.


© Minghao Pan, Bruno Mazorra, Christoph Schlegel, and Akaki Mamageishvili;
licensed under Creative Commons License CC-BY 4.0

This can be read as a trilemma: a non-wasteful mechanism in the presence of Sybils is either not incentive compatible, or not Sybil-proof, or centralising. Most related to our work, [6] prove there is no Pareto-efficient, false-name-proof combinatorial auction. We do not require Pareto efficiency, as we ask whether non-trivial distributional goals can be achieved at all; but, as a consequence of our main theorem, our axioms imply Pareto efficiency.

One way out is to require only that no bidder gains from introducing Sybils in equilibrium. Under this Bayesian notion there are mechanisms which satisfy all the remaining axioms yet do not always allocate to a highest-value bidder (Theorem 3). Thus, in contrast to the classical setting [5], Bayesian implementation is strictly weaker than dominant strategy implementation under Sybil constraints.

2 Model and main result

A **(direct) mechanism** specifies, for each finite set $\mathcal{N} \subset \mathbb{N}$ of **bidders** (some real agents, some possibly Sybils), an **allocation rule** $x^{\mathcal{N}} : \mathbb{R}_+^{\mathcal{N}} \to \Delta(\mathcal{N})$, where $\Delta(\mathcal{N}) := \{x \in \mathbb{R}_+^{\mathcal{N}} : \sum_{i\in\mathcal{N}} x_i \leq 1\}$, and a **payment rule** $p^{\mathcal{N}} : \mathbb{R}_+^{\mathcal{N}} \to \mathbb{R}^{\mathcal{N}}$. The share $x_i^{\mathcal{N}}(v)$ for reported **values** $v \in \mathbb{R}_+^{\mathcal{N}}$ is a probability of obtaining an indivisible good or a share of a divisible good, and bidder i has utility $U_i = v_i x_i - p_i$.

We impose **payment normalisation**, $p_i^{\mathcal{N}}(0, v_{-i}) = 0$; **non-wastefulness**, $\sum_{j\in\mathcal{N}} x_j^{\mathcal{N}}(v) = 1$; **symmetry**, $x_j^{\mathcal{N}}(v) = x_{\pi(j)}^{\mathcal{N}}(\{v_{\pi(i)}\}_{i\in\mathcal{N}})$ for every permutation π of $\mathcal{N}$; and **incentive compatibility** in dominant strategies, which, given payment normalisation, is equivalent to monotonicity of $x^{\mathcal{N}}$ with the Myerson payments $p_j(v) = v_j x_j(v) - \int_0^{v_j} x_j(z, v_{-j})dz$. Our **Sybil-proofness** axiom is weaker than the false-name-proofness of [6], which also allows the deviator to misreport: we ask only for immunity to attacks in which a bidder reports truthfully from his original account and creates a single Sybil with an arbitrary bid; the characterisation therefore holds a fortiori under false-name-proofness. For all $v \in \mathbb{R}_+^{\mathbb{N}}$, finite $\mathcal{N} \subset \mathbb{N}$, $i \in \mathcal{N}$, $j \in \mathbb{N} \setminus \mathcal{N}$ and $u \geq 0$,

$$v_i x_i^{\mathcal{N}}(v) - p_i^{\mathcal{N}}(v) \geq v_i \left(x_i^{\mathcal{N}\cup\{j\}}(v, u) + x_j^{\mathcal{N}\cup\{j\}}(v, u) \right) - p_i^{\mathcal{N}\cup\{j\}}(v, u) - p_j^{\mathcal{N}\cup\{j\}}(v, u).$$

▶ **Theorem 1.** *A direct mechanism is payment normalised, non-wasteful, symmetric, incentive compatible, and Sybil-proof if and only if it is a second price auction with symmetric tie-breaking.*

The proof turns Sybil-proofness into a lower bound on payoffs: under Myerson payments, $U_j(v) = \int_0^{v_j} x_j(z, v_{-j})dz$, and a bidder who reports u against $n-1$ bidders reporting $v < u$ can add a Sybil bidding v, which by symmetry and non-wastefulness wins with probability $\frac{1}{n}[1 - x_1]$ at a price of at most v. Ruling out these deviations for every n gives $\limsup_n x_1(u, v, \ldots, v) = 1$ and hence $U_1(u, v) \geq u - v$ for two bidders, while non-wastefulness pins $\frac{1}{u}\int_0^u U_1(u, v)dv$ to $\frac{u}{2}$, so that $U_1(u, v) = u - v$ and $x_1(z, u) = 0$ for every $z < u$. Induction extends this beyond two bidders; symmetry then forces uniform tie-breaking and incentive compatibility the second-price payments.

3 Bayesian Sybil-proofness

Bidders now hold a prior over bid profiles, so that both valuations and the number of real bidders can be unknown. Conditional on the set of real bidders, private values are i.i.d. F, twice differentiable and strictly increasing on its support $(0, \bar{v})$. **Bayesian Sybil-proofness**, similar to the notion in [3], requires that Sybils do not benefit a bidder in expectation if all others report truthfully and use a single identity.

▶ **Example 2.** Let Z_{n-1} be the maximum of an i.i.d. sample of size $n-1$ from F, let $y^\star = \min\{y : (\overline{v}-x)f(x) \le 1-F(x)\ \forall x \in [y,\overline{v}]\}$, and for each integer $n \ge 2$ define the *ticket price* $a_n = \max\{y^\star, E[Z_{n-1}], F^{-1}(\frac{n-2}{n-1})\}$. The **lottery mechanism with increasing ticket price** for n bids allocates as follows: if all or all but one bids are below a_n, the item is allocated by a second price auction with uniform tie-breaking; if at least $m \ge 2$ bids are at least a_n, then each of those m bidders wins with probability $1/m$ and the winner pays a_n.

The mechanism uses a monotonic allocation rule with Myerson payments and is hence dominant strategy incentive compatible; it is symmetric and non-wasteful by construction. As long as $y^\star < \overline{v}$, that is, as long as the tail of F close to $\overline{v}$ is no heavier than that of the uniform distribution on $[0,\overline{v}]$, it differs from a second price auction with positive probability for every n, and allocating randomly at a fixed price among qualifying bidders yields a weakly higher ex-interim utility. For F uniform on the unit interval, a lottery among at least two bidders occurs with probability $1-a_n^n-n(1-a_n)a_n^{n-1}$, which tends to $1-2/e \approx 0.26$.

▶ **Theorem 3.** *The lottery mechanism with increasing ticket prices is Bayesian Sybil-proof. In particular, if $y^\star < \overline{v}$, then there exists a mechanism that is payment normalised, non-wasteful, symmetric, incentive compatible, and Bayesian Sybil-proof that is non-equivalent to the second price auction.*

4 Further results and open questions

Indirect mechanisms cannot circumvent the impossibility: the full version establishes a revelation principle for Sybils, in dominant strategies and in Bayes-Nash equilibrium. The characterisation extends beyond linear valuations, to types with convex and superadditive interim utility, but not to multi-unit allocation with unit-demand bidders, where mechanisms other than the uniform-price mechanism satisfy all axioms. Open questions include how well equilibria of Sybil-proof mechanisms approximate distributional objectives, what the general class of Bayesian Sybil-proof mechanisms is and what the admissible class is when Sybils are costly.

References

1 Eric Budish and Aditya Bhave. Primary-market auctions for event tickets: Eliminating the rents of "bob the broker"? *American Economic Journal: Microeconomics*, 15(1):142–170, 2023.

2 Xi Chen, Christos Papadimitriou, and Tim Roughgarden. An axiomatic approach to block rewards. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 124–131, 2019. doi:10.1145/3318041.3355470.

3 Yotam Gafni, Ron Lavi, and Moshe Tennenholtz. Vcg under sybil (false-name) attacks-a bayesian analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1966–1973, 2020. doi:10.1609/AAAI.V34I02.5567.

4 Jacob D Leshno and Philipp Strack. Bitcoin: An axiomatic approach and an impossibility theorem. *American Economic Review: Insights*, 2(3):269–286, 2020.

5 Alejandro M. Manelli and Daniel R. Vincent. Bayesian and dominant-strategy implementation in the independent private-values model. *Econometrica*, 78(6):1905–1938, 2010.

6 Makoto Yokoo, Yuko Sakurai, and Shigeo Matsubara. The effect of false-name bids in combinatorial auctions: New fraud in internet auctions. *Games and Economic Behavior*, 46(1):174–188, 2004. doi:10.1016/S0899-8256(03)00045-9.

Belief-Aware Pivotal Mechanism for DAO Committees

Nuno Braz
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal

Diogo Poças
Instituto de Telecomunicações, Instituto Superior Técnico, Universidade de Lisboa, Portugal

Miguel Correia
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal

Abstract

Decentralized Autonomous Organizations (DAOs) increasingly delegate decisions to small committees whose members hold two independent kinds of private information: idiosyncratic preferences over alternatives (what they want) and beliefs about which alternative best serves the organization (what they know). Members have no reason to reveal what they know unless they are incentivized to do so.

Standard voting rules are designed to extract only what members want. If these two pieces of information are not aligned, the organization can end up making a suboptimal decision. Existing mechanisms for eliciting (extracting) beliefs from experts, such as Decision Scoring Rules and Decision Markets, face well-known impossibility results in multi-agent settings under deterministic decision rules, and ignore the fact that agents can have idiosyncratic preferences over alternatives.

This paper proposes a mechanism for binary committee decisions that augments the pivotal mechanism, an instance of the Groves mechanism, with a reward that depends on the outcome distributed after a boolean signal, indicating success or failure of the outcome, is observed. The mechanism aggregates the experts' private information to maximize the probability that its decision agrees with a weighted linear pooling of the experts' beliefs, framing the designer's problem as classification rather than welfare maximization.

An affine family of reward rules is proposed, and the region of parameters that simultaneously satisfy dominant-strategy incentive compatibility, interim individual rationality, and budget feasibility is derived. In informative-belief environments, the mechanism outperforms majority voting, including majority voting augmented with the same outcome-contingent rewards, especially when the committee's average preferences are biased against the superior alternative.

2012 ACM Subject Classification Theory of computation → Algorithmic mechanism design; Applied computing → Electronic commerce

Keywords and phrases Mechanism Design, Decentralized Autonomous Organizations, Committees, Information Aggregation

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.17

Supplementary Material *Software*: `https://github.com/Nunobrazz/commitee-simulator` archived at `swh:1:dir:ad34d49c6e83305835e1e63b3da599a62838a0d1`

Funding This work was financially supported by Project Blockchain.PT – Decentralize Portugal with Blockchain Agenda (Project no 51), WP 6: Digital Assets Management, Call no 02/C05-i01.01/2022, funded by the Portuguese Recovery and Resilience Program (PRR), The Portuguese Republic, and The European Union (EU) under the framework of the Next Generation EU Program. This work was also supported by national funds through Fundação para a Ciência e a Tecnologia, I.P. (FCT) under projects UID/50008/2025 (DOI: `https://doi.org/10.54499/UID/50008/2025`) and UID/50021/2025 (DOI: `https://doi.org/10.54499/UID/50021/2025`).



© Nuno Braz, Diogo Poças, and Miguel Correia;
licensed under Creative Commons License CC-BY 4.0



1 Introduction

A Decentralized Autonomous Organization (DAO) is an organization governed by rules encoded as smart contracts on a blockchain, where members coordinate and make collective decisions without centralized authority. In recent years, DAOs have grown into large-scale entities where thousands of members coordinate pseudonymously [34]. Unlike traditional contracts, which merely articulate rules and can only serve as legal proof, smart contracts can execute the rules they encode. The scale of these organizations creates a significant challenge in decision-making. Several studies [6, 33, 18] confirm that DAOs suffer from low participation rates and slow execution, showing the need for delegated governance structures. In large blockchains such as Bitcoin and Ethereum, the decision-making is done by the maintainers [8, 2]; Polkadot elects a council [30]; project Catalyst has a Community Advisors board and a veteran [31]. Either way, when governance councils are not elected, decision power is concentrated in a small number of members [18]. Empirical and analytical studies confirm that token-based voting is insufficient when participants have heterogeneous stakes and information [18, 14, 36, 11].

Restricting power to a small group or committee amplifies a natural problem: members often have personal interests that are not aligned with the organization's goal. For example, a committee governing a lending DAO might vote to approve a highly risky token as collateral simply because they personally own large amounts of it, exposing users to severe financial risk. Mechanisms such as majority voting and quadratic voting [19, 4] are designed to account only for what members *want* and not for what they *know* about which alternative benefits the organization. Furthermore, any non-continuous voting mechanism necessarily loses information by restricting experts to discrete reports. Talmon [35] identifies DAOs as an important application domain for Multi-Agent Systems and calls for the community to address their governance challenges. Our work addresses this by proposing a decision-making mechanism with formal incentive guarantees that improves collective decisions through better information aggregation.

Organizations often do an evaluation of the consequences of their decisions. These evaluations are usually informal: performance reviews, retrospectives, reputation effects. In a DAO, the evaluation can be formalized: the community can express whether a decision led to a positive or negative outcome after a predetermined period. More quantitative criteria also serve, such as treasury growth, total value locked, user adoption. The evaluation task is simpler than the original decision, as it concerns observed outcomes rather than predictions. Because smart contracts can hold funds in escrow and release them conditionally, members' compensation can naturally be tied to the actual consequences of the decisions they supported. This allows what Kiayias and Lazos [17] call *accountability*: a blockchain governance system satisfies the property of accountability if whenever participants bring in a change, they are held individually responsible for it in a clearly defined way by the platform.

Hanson's futarchy proposal [15] pioneered the idea of using prediction markets to guide policy decisions, leveraging the collective intelligence of the market to evaluate which policy would produce better outcomes. Decision Scoring Rules [28, 3] and Decision Markets [29, 5] formalize it by conditioning monetary transfers on the observed outcome, but they are designed for single experts or community-wide market participation. In practice, decision markets have been deployed in DAOs either as the decision mechanism itself (MetaDAO [22]) or as a filtering layer that delegates the final decision to a committee (DAOstack [7]). An analysis on the latter noted that the system became self-referential: the collective intelligence of the market never showed up, and proposals were passed not because they were deemed

relevant, but because their authors wished to see them approved [9]. One could envision an inverted DAOstack architecture: the committee decides, and the outcome is evaluated ex-post by a market or a peer prediction method such as the one of Miller et al. [24]. This paper formalizes a new mechanism structure where transfers can be conditioned on both reports and impact of the outcomes, abstracting practical considerations such as the medium of exchange, the committee selection process, and the outcome evaluation procedure, allowing the incentive analysis to apply across diverse instantiations.

Mechanism Overview

We focus on binary decisions, the most common type in DAOs: approximately 67% of analyzed DAO proposals use a binary voting pattern [36].

Each committee member – hereafter *expert* – possesses two independent, private types of information: *idiosyncratic preferences*, how much they personally stand to gain or lose from each alternative (what they want), and *subjective beliefs*, their assessment of which alternative will improve the organization's success metric (what they know). The designer wants to decide based on beliefs alone, but experts have no incentive to report their beliefs truthfully – their preferences create an incentive to misrepresent. Rather than trying to elicit (extract) preferences and beliefs separately, our mechanism induces each expert to submit a single continuous report that encodes both, and maximizes the probability that the aggregation of the reports matches the one the designer would make if all beliefs were observable.

Unlike Decision Scoring Rules, which elicit beliefs by directly scoring prediction accuracy, our mechanism ties payments to the realized outcome, so that each expert's belief is revealed through their own expected-utility maximization. Incentive compatibility comes from the pivotal structure, an instance of the Groves Mechanism [13], in which each agent's payment equals the externality their report imposes on the others. The role of the outcome-contingent rewards is to reshape the content of the true report, folding belief information into the scalar message that the pivotal mechanism would otherwise elicit as pure idiosyncratic preference. When experts do not have idiosyncratic preferences (a standard assumption in Decision Scoring Rules), the mechanism selects the alternative that a designer would select if given access to all private beliefs and aggregated them as a linear opinion pool, with each expert's weight equal to $\frac{b}{1+p(a-1)}$, where p is the expert's belief that their preferred alternative will succeed and a and b are positive numbers that can be different between experts. These parameters are the intuitive counterparts of the reward coefficients (α, β) derived in Section 4.

The mechanism sidesteps the strict-properness impossibilities of decision scoring rules under deterministic decision rules [29, 28] and ensures that each expert's report is strictly optimal whenever they assign non-zero probability to any aggregate of the other agents' reports.

Several mechanisms have been proposed for aggregating preferences in binary decision settings, each with different trade-offs regarding incentive compatibility, information elicitation, and budget requirements. However, most of these mechanisms do not account for the distinction between experts' idiosyncratic preferences and their beliefs about which alternative benefits the organization. Table 1 compares decision-making mechanisms along the following dimensions: whether the mechanism elicits experts' subjective beliefs about the success of each alternative (*Belief Elicitation*); whether it mitigates the influence of experts' private preferences on the collective decision (*Reduce Idiosyncratic Noise*); whether it supports multiple simultaneous participants (*Multi-Agent*) and whether it operates in a single round (*One-Shot*). In this paper, we construct a mechanism comprising an allocation

rule, a pivotal transfer rule, and an outcome-contingent reward rule that depends on a boolean signal observed after the decision is implemented. We formulate a design objective for committee decision-making – maximizing the probability that the mechanism's decision matches a *linear opinion pool* of the experts' private beliefs, framing the designer's problem as classification rather than welfare maximization. We design an affine parameterization of the reward rules and derive restrictions of the feasible region of design parameters that simultaneously satisfy dominant-strategy incentive compatibility, interim individual rationality, and budget feasibility. We derive the dominant strategy, in which each expert's report jointly encodes preferences and beliefs into a single scalar, with the reward parameters controlling how strongly beliefs enter the report and how experts' messages are scaled. Finally, we evaluate the mechanism via Monte Carlo simulations, showing that when experts receive signals correlated with the true success probabilities it outperforms majority voting with outcome-contingent rewards and a DSR applied to a random expert in most scenarios.

Table 1 Comparison of decision-making mechanisms in decentralized organizations. ● = satisfied, ○ = partially satisfied, ○ = not satisfied. †Outcome-contingent subsidies can indirectly mitigate idiosyncratic noise, but these mechanisms were not designed for this purpose.

Feature	This	MV	QV	DSR	DM
Belief Elicitation	●	●	●	○	○
Reduce Idiosyncratic Noise	◐	◐	●	●†	●†
Multi-Agent	○	●	●	●	●
One-Shot	●	○	●	◐	○

The remainder of the paper is organized as follows: Section 2 reviews related work. Section 3 presents the model, the mechanism structure, and the designer's objective. Section 4 progressively derives the admissible reward rules that satisfy a set of important properties. Section 5 analyzes the design parameters and presents simulation results. Section 6 concludes.

2 Related Work

This paper sits at the intersection of three areas: governance in decentralized autonomous organizations, information aggregation in committees, and outcome-conditioned elicitation mechanisms (decision scoring rules and decision markets). DAO governance has identified the problem we address: committees making decisions where members hold both idiosyncratic preferences and private beliefs about what best serves the organization. We review information aggregation in committees and mechanisms with similar structure to ours.

Committees

A foundational result in committee decision-making is the Condorcet Jury Theorem (1785): if each member of a group independently identifies the correct alternative with probability greater than 1/2, then majority voting selects the correct alternative with probability approaching 1 as the group grows [10]. The theorem assumes identical preferences and sincere voting. Austen-Smith and Banks [1] find that honest voting by all individuals is not rational even when they have common preferences. Li et al. [21] study committee decisions where members have conflicting preferences but share a common interest in making the correct choice, an objective close to ours. In their model, preferences are publicly known while private information (a signal about which alternative is correct) must be elicited through reports. The members' preferences derive from the personal costs of incorrect decisions, and

the model does not distinguish between bias manifested in prior beliefs and bias manifested in preferences. They show that manipulation incentives persist under any decision rule – even with continuous reports – so no equilibrium fully aggregates members' information, though some sharing still occurs. Nitzan and Paroush [27] studied the optimal weighted majority rule for a group of individuals with identical objectives but heterogeneous abilities. Each individual votes for one of two alternatives, identifying the correct one with a known probability that may differ across individuals.

Our mechanism departs from these settings by using transfers to establish a dominant strategy for each agent. The known equilibrium strategy allows the designer to provide formal incentive guarantees and to tune the mechanism's parameters to control how private information enters the aggregate, even without observing individual types.

Decision Scoring Rules and Decision Markets

A DSR pairs a scoring function that rewards a single expert based on their report and the actual outcome with a decision rule that selects an action from the report; in decision markets, experts instead produce forecasts conditional on each action. Both were first introduced by [29]. Oesterheld and Conitzer [28] showed that DSRs can only strictly incentivize reporting the expected utility of the recommended action, not the full probability distribution over outcomes. Chen et al. [5] showed that in decision markets it is impossible to strictly incentivize truthful reports about the likelihood of achieving some outcomes under deterministic decision rules; in other words, to strictly incentivize truthful reports the decision maker must risk choosing a suboptimal alternative. The authors further study the case where a single expert simply recommends an action, extending the model introduced by [29].

Our setting departs from the single-expert focus of decision scoring rules: we aim to follow the action favored by the *aggregate* of the experts' beliefs rather than that of any single expert. Extending DSRs to multiple agents directly introduces strategic manipulation. As in [29], we restrict attention to two possible outcomes, "good" and "bad". Our mechanism assigns each expert a reward that differs according to whether the realized outcome was good or bad, and charges the pivotal transfer on their report. This makes truthful reporting a dominant strategy, strictly optimal whenever an expert assigns non-zero probability to every possible aggregate of the other experts' reports. As a result, the designer can select the action it would have chosen with direct access to the experts' private beliefs under a linear pooling rule.

Considering experts with their own preferences, [3] addresses self-interested experts by introducing compensation rules that offset the expert's inherent utility from the decision. However, constructing proper compensation rules requires the principal to know the expert's utility function; when this knowledge is unavailable, properness cannot be guaranteed in general. Our mechanism does not assume knowledge of experts' preferences, though a known type distribution lets the designer anticipate its expected accuracy.

Two-Stage Mechanisms

The idea of conditioning transfers on post-decision information is not new. Mezzetti [23] proposed a two-stage Groves mechanism where agents report realized payoffs after the outcome, and [26] strengthened this to strict ex-post incentive compatibility. Our mechanism shares the two-stage structure but departs in two ways: the objective is classification rather than efficiency, and the second stage does not require agents to report their realized payoffs – a single boolean $\Delta \in \{-1, 1\}$ is observed, eliminating the need for incentive-compatible second-stage strategies.

3 Model

This section presents the model and the necessary background. We consider a single decision (one-shot setting) in which a DAO faces a choice between two mutually exclusive alternatives $\{A, B\}$. The committee consists of a set $N = \{1, \ldots, n\}$ of risk-neutral experts who aim to maximize their expected utility. Following the implementation of the alternative, a boolean state of the world, $\Delta \in \{-1, 1\}$ is observed, where $\Delta = 1$ denotes a positive impact and $\Delta = -1$ denotes a negative impact. Only the implemented alternative is realized, so the mechanism observes Δ for the chosen option alone and never the counterfactual outcome of the discarded one; conditioning transfers on Δ requires only the realized impact of the chosen alternative to be resolvable to a boolean against a benchmark, not that the two alternatives be compared directly. We take the procedure that produces Δ – a community retrospective, a quantitative KPI threshold such as treasury growth or total value locked, or an arbitration method such as Kleros [20] – as given. Its choice is orthogonal to the incentive analysis, but the guarantees hold only if Δ is observed honestly (and experts understand how it is determined).

Experts have quasi-linear utility: a payment of x to the mechanism reduces the expert's utility by exactly x, with no indirect benefit from the collective pool increasing. The medium of exchange is left unspecified – it may be tokens from the DAO treasury, reputation points, or any other transferable unit of value. We assume that transfers are enforceable, e.g. via smart contracts, and that posting collateral does not change the expert's utility function. Experts are risk neutral, thus they value the payment now as much as they will value it later. The rules of the mechanism are common knowledge. We focus on environments where the number of experts is insufficient for traditional decentralized prediction markets to function effectively.

3.1 Information Structure

Each expert $i \in N$ has a private type – a collection of private information that fully characterizes the expert – comprising their idiosyncratic preferences and subjective beliefs.

The expert's idiosyncratic utility is captured by the pair $v_i = (v_i^A, v_i^B) \in \mathbb{R}^2$, where v_i^A and v_i^B represent the expert's personal utility derived from the implementation of option A and option B, respectively. Since exactly one of these options will be implemented, the expert can normalize their preference parameters accordingly so without loss of generality, at least one of these alternatives yields a non-negative idiosyncratic utility for the expert.

The expert's subjective beliefs are represented by the pair $p_i = (p_i^A, p_i^B) \in \mathcal{P}^2$, where $\mathcal{P} = (0, 1)$ and p_i^A and p_i^B denote the expert's private assessment of the probability that alternatives A and B will lead to a positive outcome for the organization, respectively. We exclude the cases where the experts are certain, to avoid unnecessary notation. They are formally defined as the conditional probabilities of observing a positive outcome given the chosen alternative: $p_i^A = \mathbb{P}_i(\Delta > 0 \mid A)$ and $p_i^B = \mathbb{P}_i(\Delta > 0 \mid B)$. These are subjective conditionals: they may differ across experts and generally will not coincide with the true success probabilities.

The expert's type is given by $\theta_i \in \Theta_i$, where $\theta_i = (v_i^A, v_i^B, p_i^A, p_i^B)$. We denote the type profile of all experts' types by $\theta = (\theta_1, \ldots, \theta_n)$ and the belief profile by $\mathbf{p} = (p_1, \ldots, p_n)$ with $p_i = (p_i^A, p_i^B)$. We operate under the Independent Private Values (IPV) model: each expert's type θ_i is drawn independently from some distribution over the type space Θ_i.

To clarify this assumption, we contrast our setting with the mineral rights auction [16]. There, each bidder receives a private geological signal, but the true value of the tract is a common function of all signals. Learning another bidder's signal would cause a bidder

to revise their own valuation. A correlated values model is necessary precisely because no public deliberation stage resolves the common-value uncertainty before bids are placed. In DAO governance the mechanism is preceded by forum discussions, temperature checks, and community calls that generate a common public signal observed by all experts. Each expert absorbs this shared information and updates accordingly. We capture what stays private after discussion. In this way, an expert's private signal carries no information about another expert's signal: learning it would not cause a further update.

We do not assume that the two pairs (v_i^A, v_i^B) and (p_i^A, p_i^B) are correlated. Experts tend to benefit from the options they believe will succeed which aligns private incentives with informational contributions and makes the designer's task easier. Independence is therefore a conservative assumption.

The mechanism takes the committee as given: the selection procedure determines robustness to coordination among experts. The properties rely on independent private values; when experts coordinate before reporting – through shared information, bribery, or vote-buying – this assumption breaks down.

3.2 Mechanism Structure

A direct revelation mechanism asks each participant to report their private information and then determines an outcome and monetary transfers as a function of the submitted messages. Formally, it consists of two components: an *allocation rule*, which maps reports to an outcome, and a *transfer rule*, which determines the payment each participant makes or receives based on the reports. We introduce a third component, a *reward rule*, that distributes payments after the outcome is observed. The reward rule is defined separately from the transfer rule because the expert evaluates it only in expectation over the unknown outcome at the time of reporting, while the transfer is known. This separation allows us to restrict the admissible reward rules independently of the allocation and transfer rules.

In our case, the mechanism proceeds in two stages, illustrated in Figure 1. First, each expert submits a message encoding how strongly the expert favors one alternative over the other. The mechanism aggregates these messages, selects an alternative, and charges each expert a transfer that depends only on the submitted messages. Second, after the outcome is observed, the mechanism distributes a reward that depends on whether the decision led to a positive or negative outcome. The expert's payoff is therefore determined by three factors: which option is chosen, how much they pay upfront, and what reward they receive once the outcome is revealed.

All experts share a common message space $\mathcal{R}$. The profile of messages submitted by all experts is denoted by $\mathbf{m} = (m_1, \ldots, m_n) \in \mathcal{R}^n$. We use the notation $\mathbf{m}_{-i}$ for the profile of all experts except i, and sometimes write $\mathbf{m} = (m_i, \mathbf{m}_{-i})$.

We formally define a mechanism in this setting as a triplet $\mathcal{M} = (x, t, r)$. We use $\mathbf{t}, \mathbf{r}$ to refer to vectors denoting the application of each rule:

- $x : \mathcal{R}^n \to \{A, B\}$ is the deterministic allocation rule representing the collective decision.
- $\mathbf{t} = (t_1, \ldots, t_n)$ is the transfer rule, where $t_i : \mathcal{R}^n \to \mathbb{R}$ represents a monetary transfer to expert i based solely on the submitted messages. A positive transfer means the expert receives money, while a negative transfer means they pay.
- $\mathbf{r} = (r_1, \ldots, r_n)$ is the reward rule, where $r_i : \mathcal{R}^n \times \{-1, 1\} \to \mathbb{R}$ represents the ex-post reward distributed to expert i, contingent on the realized state of the world Δ.

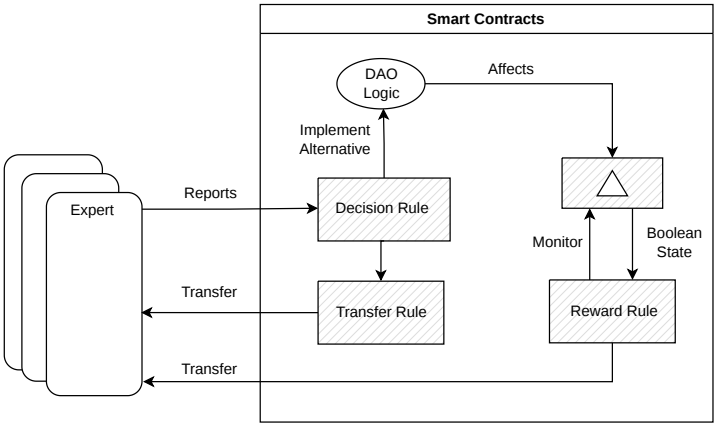


Figure 1 Mechanism Overview.

Given a type $\theta_i = (v_i^A, v_i^B, p_i^A, p_i^B)$ and a profile of submitted messages $\mathbf{m}$, let $o = x(\mathbf{m}) \in \{A, B\}$ denote the chosen alternative. We write v_i^o, p_i^o for the expert's utility and belief associated with the chosen alternative. The expected utility of expert i reflects the three factors described above:

$$\mathbb{E}[u_i(\mathbf{m} \mid \theta_i)] = v_i^o + t_i(\mathbf{m}) + \mathbb{E}_\Delta[r_i(\mathbf{m}, \Delta) \mid p_i^o] \tag{1}$$

3.3 Background: The Pivotal Mechanism

Our mechanism builds on the Pivotal Mechanism [13]. Each expert submits a scalar $m_i \in \mathbb{R}$ representing the intensity of their preference. The allocation rule selects A if the sum of messages is non-negative, otherwise it selects B:

$$x(\mathbf{m}) = \begin{cases} A & \text{if } \sum_{i=1}^n m_i \geq 0 \\ B & \text{otherwise} \end{cases} \tag{2}$$

The transfer charges each pivotal expert the externality they impose on others:

$$t_i(\mathbf{m}) = \begin{cases} -\sum_{j \neq i} m_j & \text{if } -m_i > \sum_{j \neq i} m_j > 0 \\ \sum_{j \neq i} m_j & \text{if } -m_i < \sum_{j \neq i} m_j < 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Non-pivotal experts pay nothing and reporting $m_i = v_i^A - v_i^B$ is a dominant strategy.

In the pivotal mechanism, each expert has a *winning set*: the set of profiles where their preferred alternative is selected. An expert who prefers A ($m_i > 0$) wins when $\sum_j m_j \geq 0$; submitting a larger positive message pushes the aggregate toward their winning set. An expert who prefers B ($m_i < 0$) wins when $\sum_j m_j < 0$; submitting a more negative message pushes toward theirs. The *critical value* is the minimum message needed to enter the winning set: $c_i = -\sum_{j \neq i} m_j$. The pivotal transfer (3) charges exactly this value to pivotal experts. By Myerson's lemma [25], a mechanism is DSIC if and only if the allocation is monotone in the message and each agent pays their critical value. DSIC is characterized in terms of the sum of both transfers: $t_i(\mathbf{m}) + \mathbb{E}_\Delta[r_i(\mathbf{m}, \Delta)]$.

3.4 Designer's Objective

Unlike most mechanisms that seek to maximize the sum of idiosyncratic preferences v_i, here these valuations are exactly the idiosyncratic noise the designer wants to filter out. The designer instead wants to select the alternative most likely to yield a positive outcome for the organization, which is given by the experts' beliefs, a quantity no agent wants to report directly.

The designer's goal is to select the alternative that an informed designer would choose if given direct access to all experts' private beliefs about each alternative's success probability. Let $x^* : \mathcal{P}^{2n} \to \{A, B\}$ denote an ideal decision rule – the allocation the designer would choose if all beliefs were observable. However, the designer does not observe p_i; instead, the mechanism collects messages m_i that encode both v_i and p_i. The designer seeks to maximize the probability that the mechanism's decision agrees with the ideal one:

$$\max_{x,t,r} \mathbb{P}(x(\mathbf{m}^*) = x^*(\mathbf{p}))\,, \tag{4}$$

where $\mathbf{m}^* = (m^*(v_1, p_1), \ldots, m^*(v_n, p_n))$ is the message profile under the dominant strategy m^* induced by (x, t, r), and the probability is over the random types (v_i, p_i).

A *strategy* for agent i is a function $m : \mathbb{R}^2 \times \mathcal{P}^2 \to \mathbb{R}$ mapping the agent's type (v_i, p_i) to a message $m_i = m(v_i, p_i)$. The designer seeks a mechanism in which each agent has a dominant strategy that depends on both components of the type. Since the mechanism's instruments (x, t, r) cannot depend on p_i directly, the dominant strategy must be *implementable*: it must arise from the agent's optimization under rules independent of p_i.

We instantiate the general objective by choosing an ideal decision rule x^* based on *linear pooling* [37, 32]. Linear pooling is a well-studied method for combining expert forecasts: it averages the individual probability assessments with weights set by the designer. In our setting, this amounts to selecting A whenever the experts' average belief differential favors it:

$$x^*(\mathbf{p}) = \begin{cases} A & \text{if } \sum_{i=1}^n w_i^A p_i^A - w_i^B p_i^B > 0 \\ B & \text{otherwise} \end{cases} \tag{5}$$

An alternative is logarithmic pooling, which gives disproportionate weight to extreme assessments: if one expert assigns 90% to A while five others assign 40%, logarithmic pooling selects A, whereas linear pooling selects B.

Since the designer does not observe (p_i^A, p_i^B), the mechanism must rely on the messages m_i. The designer therefore seeks a reward rule r such that the sign of the aggregate message matches the sign of the aggregate belief differential:

$$\operatorname{sgn}\left(\sum_{i=1}^n m_i\right) = \operatorname{sgn}\left(\sum_{i=1}^n w_i^A p_i^A - w_i^B p_i^B\right), \quad \forall \theta \in \Theta \tag{6}$$

subject to the constraints defined in the next section. We consider equal weights throughout the rest of the paper.

3.5 Formalization of the Properties

The desired properties for the decision-making mechanism are formalized below.

The mechanism must remain invariant to the relabeling of alternatives. Since the reports are reflected by the experts' utility functions we formalize this requirement as:

▶ **Property 1** (Symmetry). *A mechanism satisfies Symmetry if, for every agent i, every type $\theta_i = (v_i^A, v_i^B, p_i^A, p_i^B)$ with relabeled type $\tilde{\theta}_i = (v_i^B, v_i^A, p_i^B, p_i^A)$:*

$$\mathbb{E}[u_i(\mathbf{m} \mid \theta_i)] = \mathbb{E}[u_i(-\mathbf{m} \mid \tilde{\theta}_i)]$$

where $-\mathbf{m}$ is the profile obtained by negating each message.

Next, we establish the condition under which experts are willing to participate in the DAO's decision-making process, formally known as Individual Rationality (IR). We adopt the notion of interim IR: each expert knows their own preferences and beliefs when deciding whether to vote, but does not know what the other experts will report. Additionally, there also exist the notions of ex-ante (before learning one's type) and ex-post (after all messages are submitted).

▶ **Property 2** (Interim Individual Rationality). *A decision-making mechanism satisfies satisfies Interim Individual Rationality (IR) with reservation level $\bar{u}$ if:*

$$\forall i \in N, \forall \theta_i \in \Theta_i, \forall \mathbf{m}_{-i} \in \mathcal{R}_{-i} : \quad \mathbb{E}[u_i(m_i^*, \mathbf{m}_{-i} \mid \theta_i)] \geq \mathbb{E}[u_i(\mathbf{m}_{-i} \mid \theta_i)] + \bar{u}$$

for some $m_i^* \in \mathcal{R}$.

A committee decision is *non-excludable*: expert i is bound by the collective outcome whether or not they vote. The relevant outside option is therefore not zero utility but *abstention* – the expert declines to influence the vote while the others still decide, receiving $\mathbb{E}[u_i(\mathbf{m}_{-i} \mid \theta_i)]$, in which $t_i = 0$, the reward vanishes, and they obtain only their valuation $v_i^{x(\mathbf{m}_{-i})}$ of the outcome chosen by others. Participation is individually rational when the expected gain over this abstention payoff clears a reservation level $\bar{u} \leq 0$, which measures how far below the abstention payoff an expert will still tolerate in order to keep their seat. When exit is costless, an expert participates only if doing so is at least as good as abstaining, so $\bar{u} = 0$.

Having established the conditions for expert participation, we must also consider the financial viability of the mechanism from the perspective of the DAO. A mechanism satisfies the Budget Constraint if the total net compensation issued by the DAO to the experts is bounded by a maximum subsidy. This bound applies regardless of the medium of exchange: even when transfers are denominated in reputation points rather than tokens, the mechanism is constrained by the maximum value that experts attribute to reputation.

▶ **Property 3** (Budget Feasibility). *A decision-making mechanism satisfies Budget Feasibility with budget $c \geq 0$ if:*

$$\forall \mathbf{m} \in \mathcal{R}^n, \forall \Delta \in \{-1, 1\} : \quad \sum_{i \in N} \Big(t_i(\mathbf{m}) + r_i(\mathbf{m}, \Delta) \Big) \leq c$$

A mechanism is incentive compatible if experts maximize their expected utility by honestly revealing this information rather than behaving strategically. A DSIC mechanism guarantees that reporting honestly is an expert's best strategy regardless of the reports of other participants.

▶ **Property 4** (Dominant Strategy Incentive Compatibility). *A decision-making mechanism is Dominant Strategy Incentive Compatible (DSIC) if,*

$$\forall i \in N, \forall \theta_i \in \Theta_i, \forall m_i \in \mathcal{R}, \forall \mathbf{m}_{-i} \in \mathcal{R}^{n-1} :$$

$$\mathbb{E}[u_i(m_i^*, \mathbf{m}_{-i} \mid \theta_i)] \geq \mathbb{E}[u_i(m_i, \mathbf{m}_{-i} \mid \theta_i)]$$

where m_i^* denotes the honest reporting strategy for type θ_i.

Ideally, the designer would observe each expert's full type $\theta_i = (v_i, p_i)$, discard the preferences v_i, and decide based on the beliefs p_i alone. But experts have no incentive to report p_i truthfully – their preferences create an incentive to misrepresent their beliefs. Rather than eliciting v_i and p_i separately, we design a mechanism that induces experts to report a single number m_i^* that encodes both, in a way that brings the collective decision closer to the designer's goal. What counts as "truthful" depends on the question being asked. Throughout the paper, we use DSIC to mean that the mechanism has a dominant strategy $m_i^*(\theta_i)$. By the revelation principle, this is without loss of generality [12].

4 Mechanism Design Space

The mechanism uses the pivotal transfer from Section 3.3 and augments it with an outcome-contingent reward rule. The protocol proceeds as follows: (1) each expert deposits a stake into the governance smart contract; (2) each expert submits a message $m_i \in \mathbb{R}$; (3) the allocation rule selects an alternative; (4) the pivotal transfer is deducted from each expert's escrow; (5) after a predetermined evaluation period, the on-chain tool observes $\Delta \in \{-1, 1\}$; (6) the reward is distributed and remaining escrow is returned.

We restrict attention to *anonymous* reward rules, where every expert faces the same rule $r_i = r$. Non-anonymous rules, assigning per-expert parameters, let the designer attribute different weights to experts – prioritizing those with demonstrated expertise – without affecting the incentive guarantees. The properties from Section 3.5 progressively narrow the space of admissible reward rules.

Symmetry

The reward rule has four components: r_A^+, r_A^-, r_B^+, and r_B^-, corresponding to the two alternatives and two outcome realizations. These are expert i's reward functions; throughout this paragraph we suppress the expert subscript. Since the mechanism must not favor either alternative, the reward for choosing A under a given profile must equal the reward for choosing B under the relabeled profile. This reduces the design space from four functions to two:

▶ **Proposition 5.** *The mechanism (x, t, r) satisfies Symmetry if and only if*

$$r_A^+(\mathbf{m}) = r_B^+(-\mathbf{m}), \qquad r_A^-(\mathbf{m}) = r_B^-(-\mathbf{m}) \qquad \forall\, \mathbf{m}$$

Proof. Let $o = x(\mathbf{m})$ denote the chosen alternative under a message profile $\mathbf{m}$ and $\bar{o} = x(-\mathbf{m})$. We have

$$\mathbb{E}[u_i(\mathbf{m} \mid \theta_i)] = v_i^o + t_i(\mathbf{m}) + p_i^o\, r_o^+(\mathbf{m}) + (1 - p_i^o)\, r_o^-(\mathbf{m})$$

$$\mathbb{E}[u_i(-\mathbf{m} \mid \tilde{\theta}_i)] = v_i^{\bar{o}} + t_i(-\mathbf{m}) + p_i^{\bar{o}}\, r_{\bar{o}}^+(-\mathbf{m}) + (1 - p_i^{\bar{o}})\, r_{\bar{o}}^-(-\mathbf{m})$$

By the neutrality of the allocation rule (negating all messages selects the opposite alternative: $\bar{o} = x(-\mathbf{m})$ is the opposite of $o = x(\mathbf{m})$) and definition of the pivotal transfer, we have, $v_i^o + t_i(\mathbf{m}) = v_i^{\bar{o}} + t_i(-\mathbf{m})$. Since the relabeled type $\tilde{\theta}_i$ swaps $p_i^A \leftrightarrow p_i^B$, we have $p_i^{\bar{o}}$ under $\tilde{\theta}_i$ equals p_i^o under θ_i. Thus $\mathbb{E}[u_i(\mathbf{m} \mid \theta_i)] = \mathbb{E}[u_i(-\mathbf{m} \mid \tilde{\theta}_i)]$ iff $p_i^o\, r_o^+(\mathbf{m}) + (1 - p_i^o)\, r_o^-(\mathbf{m}) = p_i^o\, r_{\bar{o}}^+(-\mathbf{m}) + (1 - p_i^o)\, r_{\bar{o}}^-(-\mathbf{m})$ for all $p_i^o \in (0, 1)$. This holds iff $r_o^+(\mathbf{m}) = r_{\bar{o}}^+(-\mathbf{m})$ and $r_o^-(\mathbf{m}) = r_{\bar{o}}^-(-\mathbf{m})$, which gives $r_A^+(\mathbf{m}) = r_B^+(-\mathbf{m})$ and $r_A^-(\mathbf{m}) = r_B^-(-\mathbf{m})$ for all $\mathbf{m}$. ◀

Symmetry thus collapses the four reward functions to two. We define $r_i^+(\mathbf{m}) := r_A^+(\mathbf{m})$ and $r_i^-(\mathbf{m}) := r_A^-(\mathbf{m})$, so that

$$r_i(\mathbf{m}, \Delta) = \begin{cases} r_i^+(\mathbf{m}) & \text{if } \Delta = 1 \\ r_i^-(\mathbf{m}) & \text{if } \Delta = -1 \end{cases}$$

It remains to determine how r_i^+ and r_i^- may depend on the message profile. We write them as functions of m_i and $\mathbf{m}_{-i}$; whenever the subscript i appears explicitly among the arguments, we drop it in the function name. The *expected transfer* of expert i – the transfer plus the reward evaluated in expectation over Δ according to the expert's belief p_i^o – is:

$$\hat{t}_i(\mathbf{m}) \;=\; t_i(\mathbf{m}) + p_i^o\, r^+(m_i, \mathbf{m}_{-i}) + (1 - p_i^o)\, r^-(m_i, \mathbf{m}_{-i}). \tag{7}$$

Budget Feasibility

The Budget Feasibility restricts how r^+ and r^- can grow with $|t_i|$:

▶ **Proposition 6.** *If the mechanism (x, t, r) satisfies the Budget Feasibility and the message space is unrestricted ($m_i \in \mathbb{R}$), then r^+ and r^- are bounded above by an affine function of $|t_i|$.*

Proof. For any fixed $\mathbf{m}_{-i}$, the transfer t_i can be made arbitrarily large in magnitude by varying m_i. If r^+ (or r^-) grows faster than $|t_i|$ in the positive direction, the total payout $\sum_{i \in N}(t_i + r_i^{\pm})$ grows without bound, violating the Budget Feasibility. Hence the positive part of r^+ and r^- is at most affine in $|t_i|$. ◀

The proposition bounds the reward from above but not from below, and does so for each fixed $\mathbf{m}_{-i}$ separately. A priori, the admissible form is therefore:

$$r^+(m_i, \mathbf{m}_{-i}) = \big(\alpha^+(\mathbf{m}_{-i}) - 1\big)\, t_i(m_i, \mathbf{m}_{-i}) + \beta^+(\mathbf{m}_{-i}) - g^+(m_i, \mathbf{m}_{-i}),$$
$$r^-(m_i, \mathbf{m}_{-i}) = \big(\alpha^-(\mathbf{m}_{-i}) - 1\big)\, t_i(m_i, \mathbf{m}_{-i}) + \beta^-(\mathbf{m}_{-i}) - g^-(m_i, \mathbf{m}_{-i})$$

where $g^+, g^- \geq 0$ are arbitrary non-negative functions representing additional penalties beyond the affine component. Dominant-strategy incentive compatibility (Theorem 9) requires the induced transformation to be independent of $\mathbf{m}_{-i}$, which forces $\alpha^{\pm}$ and $\beta^{\pm}$ to be constant in $\mathbf{m}_{-i}$; otherwise the dominant strategy would depend on the other experts' reports. We set $g^+ = g^- = 0$, restricting to affine rewards, which implement the linear opinion pool that matches our instantiation of the designer's goal. Non-affine rewards remain DSIC whenever they correspond to a strictly increasing message transformation (Lemma 8). These functions might be used to implement a nonlinear aggregation rule; characterizing the non-affine family is left to future work. The resulting affine parameterization is:

$$r^+(m_i, \mathbf{m}_{-i}) = (\alpha^+ - 1)\, t_i(m_i, \mathbf{m}_{-i}) + \beta^+ \qquad r^-(m_i, \mathbf{m}_{-i}) = (\alpha^- - 1)\, t_i(m_i, \mathbf{m}_{-i}) + \beta^- \tag{8}$$

where $\alpha^+, \alpha^-, \beta^+, \beta^- \in \mathbb{R}$ are design parameters. The coefficient $(\alpha^{\pm} - 1)$ measures how much the reward amplifies or attenuates the pivotal transfer when the outcome is positive or negative, and $\beta^{\pm}$ is a fixed payment distributed independently of the value transfer.

Substituting (8) into (7) and writing $\alpha(p) = \alpha^+ p + \alpha^-(1-p)$ and $\beta(p) = \beta^+ p + \beta^-(1-p)$, the expected transfer simplifies to:

$$\hat{t}_i(\mathbf{m}) = \alpha(p_i^o) \cdot t_i(\mathbf{m}) + \beta(p_i^o), \tag{9}$$

where $o = x(\mathbf{m})$ is the chosen alternative.

The β term acts as a belief-dependent shift of the expert's valuation; write $\hat{v}_i := v_i + \beta(p_i^A) - \beta(p_i^B)$ for the resulting *effective valuation*, where $v_i = v_i^A - v_i^B$. The expert behaves as a pivotal agent with valuation $\hat{v}_i$ in place of v_i.

The four parameters $(\alpha^+, \alpha^-, \beta^+, \beta^-)$ of the mechanism class (8) constitute the design space. The budget feasibility further restricts this space:

▶ **Proposition 7.** *Under the Budget Feasibility, the design parameters satisfy* $\alpha^+ \geq 0$, $\alpha^- \geq 0$, $\beta^+ \leq c/n$, *and* $\beta^- \leq c/n$.

Proof. Substituting (8) into the Budget Feasibility for $\Delta = 1$:

$$\sum_{i \in N} \Big(t_i(\mathbf{m}) + (\alpha^+ - 1)\, t_i(\mathbf{m}) + \beta^+ \Big) = \alpha^+ \sum_{i \in N} t_i(\mathbf{m}) + n\beta^+ \;\leq\; c \qquad \forall\, \mathbf{m}.$$

If the expert is pivotal, $t_i(\mathbf{m}) \leq 0$ for all i, and $\sum_i t_i(\mathbf{m})$ can be made arbitrarily negative by increasing the magnitude of the messages. If $\alpha^+ < 0$, the term $\alpha^+ \sum_i t_i$ is positive and unbounded, violating the constraint. Hence $\alpha^+ \geq 0$. Given $\alpha^+ \geq 0$, the left-hand side is maximized when $\sum_i t_i = 0$ (no expert is pivotal), yielding $n\beta^+ \leq c$, i.e., $\beta^+ \leq c/n$. The same argument applied to $\Delta = -1$ gives $\alpha^- \geq 0$ and $\beta^- \leq c/n$. ◀

It remains to determine which parameter values $(\alpha^+, \alpha^-, \beta^+, \beta^-)$ of the mechanisms defined by (8) yield DSIC and IR, and what dominant strategy they induce.

Dominant Strategy Incentive Compatibility

The following lemma shows that applying a strictly increasing transformation to the message space of a DSIC mechanism preserves incentive compatibility, with the critical value transformed by the inverse.

▶ **Lemma 8.** *Let* (x, t) *be a DSIC mechanism with winning set* W_i *and critical value* $c_i = \inf\{m_i : x(m_i, \mathbf{m}_{-i}) \in W_i\}$. *If the designer wants to coerce an agent to submit* $\varphi(v_i)$ *instead of* v_i, *where* $\varphi : \mathcal{R} \to \mathcal{R}$ *is strictly increasing, then the transformed mechanism is DSIC with critical value* $\varphi^{-1}(c_i)$.

Proof. The original mechanism is DSIC, so the allocation rule $x(v_i, \mathbf{m}_{-i})$ is monotonically increasing in v_i and agents pay their critical value when in the winning set. Under the transformation, the allocation becomes $x(\varphi(v_i), \mathbf{m}_{-i})$, which is increasing in v_i since φ is strictly increasing. The new critical value is

$$c_i' = \inf\{v_i : \varphi(v_i) \in W_i(\mathbf{m}_{-i})\} = \inf\{v_i : \varphi(v_i) \geq c_i\} = \varphi^{-1}(c_i).$$

Since monotonicity is preserved and with critical value $\varphi^{-1}(c_i)$, the transformed mechanism is DSIC. ◀

In other words, if the designer applies a strictly increasing transformation φ to the critical value, then the mechanism remains DSIC and the agents have dominant strategy $\varphi^{-1}(v)$.

▶ **Theorem 9.** *Let* $\alpha^+, \alpha^- \geq 0$ *with* $(\alpha^+, \alpha^-) \neq (0, 0)$. *The mechanism* (x, t, r) *is DSIC with dominant strategy*

$$m^*(v_i, p_i) = \begin{cases} \dfrac{v_i + \beta(p_i^A) - \beta(p_i^B)}{\alpha(p_i^A)} & \text{if } v_i + \beta(p_i^A) - \beta(p_i^B) > 0 \\ \dfrac{v_i + \beta(p_i^A) - \beta(p_i^B)}{\alpha(p_i^B)} & \text{if } v_i + \beta(p_i^A) - \beta(p_i^B) < 0 \end{cases} \tag{10}$$

where $v_i = v_i^A - v_i^B$.

Proof. Fix expert i with beliefs $p_i = (p_i^A, p_i^B)$. The β term shifts the expert's effective valuation to $\hat{v}_i$; the α term scales the transfer by $\alpha(p_i^o)$. When $\hat{v}_i > 0$ the expert prefers A, so being in the winning set means $o = A$ and the scaling is $\alpha(p_i^A)$; when $\hat{v}_i < 0$ the expert prefers B, giving $\alpha(p_i^B)$. Define the transformation

$$\varphi(\hat{v}_i) = \begin{cases} \hat{v}_i/\alpha(p_i^A) & \text{if } \hat{v}_i > 0 \\ \hat{v}_i/\alpha(p_i^B) & \text{if } \hat{v}_i < 0 \end{cases}$$

which is strictly increasing when $\alpha(p_i^A), \alpha(p_i^B) > 0$. By Lemma 8, the mechanism is DSIC with dominant strategy $m_i^* = \varphi(\hat{v}_i)$, yielding (10). ◀

Note that φ must depend only on the expert's own type, not on $\mathbf{m}_{-i}$: if it did, the dominant strategy would depend on $\mathbf{m}_{-i}$ as well, and the mechanism would no longer be DSIC. The dominant strategy is weakly optimal for all type profiles and strictly optimal whenever the expert assigns non-zero probability to being pivotal, i.e., to the event that their report changes the collective decision.

When experts have no idiosyncratic preferences, the mechanism reduces to pure belief elicitation, let, $\Delta\beta := \beta^+ - \beta^-$,

▶ **Corollary 10.** *When $v_i = 0$ for all i and $\Delta\beta > 0$, the dominant strategy reduces to $m_i^* = \Delta\beta\,(p_i^A - p_i^B)/\alpha(p_i^o)$. In particular, taking $\alpha(p_i^o) = 1$ (e.g. $\alpha^+ = \alpha^- = 1$) recovers the equal-weight linear opinion pool of* (5) *exactly; other admissible α yield a belief-weighted pool with weights $1/\alpha(p_i^o)$.*

If the designer assigns non-anonymous reward rules, each expert i receives a per-expert $\Delta\beta_i$, which lets the designer weight experts by criteria such as expertise. Taking $\alpha = 1$, $m_i^* = \Delta\beta_i\,(p_i^A - p_i^B)$, the ratio $\Delta\beta_i/\Delta\beta_j$ fixes how much more expert i's belief differential counts than expert j's in the aggregate. For instance, setting $\Delta\beta_i = 2\,\Delta\beta_j$ makes expert i pivotal-equivalent to two experts j holding the same belief.

Interim Individual Rationality

Using the reservation level $\bar{u}$ from Property 2.

▶ **Proposition 11.** *The mechanism satisfies Interim IR if and only if $\beta(p) \geq \bar{u}$ for all $p \in (0, 1)$.*

Proof. Under DSIC, truthful reporting weakly dominates any alternative report, including any report that makes expert non-pivotal. It therefore suffices to verify that the non-pivotal deviation satisfies IR; the truthful (possibly pivotal) report can only improve on it. When non-pivotal, $t_i = 0$ and the allocation is unchanged. The participation gain over $v_i^{x(\mathbf{m}_{-i})}$ is $\beta(p_i^{x(\mathbf{m})})$. IR requires this to be at least $\bar{u}$ for all $p \in (0, 1)$. Since $\beta(p) = \beta^+ p + \beta^-(1-p)$ is linear in p, the condition $\beta(p) \geq \bar{u}$ for all $p \in (0, 1)$ holds if and only if $\beta^+, \beta^- \geq \bar{u}$. ◀

The four design parameters $(\alpha^+, \alpha^-, \beta^+, \beta^-)$ are constrained by the mechanism's structural and incentive properties. The resulting feasible region restrictions, are summarized in Table 2. The mechanisms of the form (8) leave the designer free to choose the following parameters $(\alpha^+, \alpha^-, \beta^+, \beta^-)$, where the ratio α^+/α^- controls the reweighting of experts valuation and the β parameters determine the strength of the belief-based incentive.

Table 2 Parameter restrictions from each mechanism property.

Property	Constraint	Reference
Budget Feasibility	$\alpha^+, \alpha^- \geq 0$ $\beta^+, \beta^- \leq c/n$	Proposition 7
DSIC	$(\alpha^+, \alpha^-) \neq (0, 0)$	Theorem 9
Interim IR	$\beta^+, \beta^- \geq \bar{u}$	Proposition 11
Feasible region restrictions	$\alpha^+, \alpha^- \geq 0,\ (\alpha^+, \alpha^-) \neq (0, 0)$ $\beta^+, \beta^- \in [\bar{u},\ c/n]$	

A note on Complexity

The mechanism is computationally lightweight. The allocation rule computes a single sum $\sum_{i=1}^{n} m_i$ in $O(n)$ arithmetic operations. The Pivotal transfer for each expert requires evaluating $\sum_{j \neq i} m_j = \sum_{i=1}^{n} m_i - m_i$, which is $O(1)$ per expert given $\sum_{i=1}^{n} m_i$, yielding $O(n)$ total. The reward rule performs a constant number of operations per expert, thus the entire mechanism therefore runs in $O(n)$ time. The remaining cost is the evaluation of Δ, a single evaluation per decision. From a smart contract perspective, the on-chain cost is equally modest. The contract stores n messages and the budget c, computes the sum, determines the allocation, and records the Pivotal transfers – all in a single transaction with $O(n)$ storage writes and arithmetic. The ex-post reward is computed in a second transaction after Δ is resolved, requiring $O(n)$ operations. The only overhead beyond a standard voting mechanism is the evaluation of Δ, which depends on the choice of the evaluation tool, not on the mechanism itself.

5 Parameter Analysis

This section studies how the choice of parameters affects the mechanism's decisions, how the β terms control the strength of the belief signal and how the ratio α^+/α^- reweights experts according to their beliefs. We further reformulate the designer's choice as an optimization problem over the mechanisms of the form (8). The resulting objective has no closed form, so we evaluate it through Monte Carlo simulation, comparing the mechanism against majority voting with and without outcome-contingent rewards and a DSR applied to a random expert across committee sizes, preference biases, and belief-noise regimes.

5.1 Parameters

Each parameter pair shapes the dominant strategy in a distinct way: the β terms control the strength of the belief signal, while the α reweights experts according to their beliefs.

Betas

The belief differential $\beta(p_i^A) - \beta(p_i^B) = \Delta\beta\,(p_i^A - p_i^B)$, determines how strongly the expert's message reflects their beliefs. When $\Delta\beta > 0$, experts who believe A is more likely to succeed ($p_i^A > p_i^B$) shift their message toward A, and symmetrically, experts who believe B is more likely to succeed ($p_i^B > p_i^A$) shift their message toward B. The magnitude $\Delta\beta$ controls the strength of this incentive in both directions.

Each expert's preferred-outcome belief p_i^o depends on the sign of $\hat{v}_i = v_i + \Delta\beta\,(p_i^A - p_i^B)$, so varying $\Delta\beta$ can flip which alternative an expert prefers, thereby changing the α weight. For this reason, widening $\Delta\beta$ can induce additional preference flips that alter the α-weights and may not always improve classification. Maximizing the belief signal is therefore not unconditionally optimal. In the environments we simulate, however, this effect does not bind: accuracy is monotone in $\Delta\beta$, so the optimum is attained at the budget bound $\Delta\beta = c/n - \bar{u}$ (i.e. $\beta^+ = c/n, \beta^- = \overline{u}$).

Alphas

Each expert's message is scaled by $1/\alpha(p_i^o)$, where p_i^o is the expert's belief that their preferred alternative will lead to a positive outcome. Since $\alpha(p)$ is linear in p, the relative values of α^+ and α^- determine how the mechanism reweights experts based on p_i^o. When $\alpha^+ < \alpha^-$, experts with high p_i^o have smaller $\alpha(p_i^o)$ and thus larger messages, contributing more to the aggregate. When $\alpha^+ > \alpha^-$, the opposite holds. When $\alpha^+ = \alpha^-$, all experts are weighted equally regardless of p_i^o.

▶ **Proposition 12.** *For any $\lambda > 0$, the mechanisms with parameters (α^+, α^-) and $(\lambda\alpha^+, \lambda\alpha^-)$ select the same alternative for every type profile.*

Proof. Under the dominant strategy, $m_i^* = \hat{v}_i/\alpha(p_i^o)$. Replacing (α^+, α^-) by $(\lambda\alpha^+, \lambda\alpha^-)$:

$$m_i^*\big|_{(\lambda\alpha^+,\lambda\alpha^-)} = \frac{\hat{v}_i}{\lambda\alpha^+ p_i^o + \lambda\alpha^-(1-p_i^o)} = \frac{1}{\lambda\alpha^-}\cdot\frac{\hat{v}_i}{1 + p_i^o(\frac{\alpha^+}{\alpha^-} - 1)}$$

Since $\lambda, \alpha^- > 0$, the sign of $\sum_i m_i^*$ is preserved. Hence the decision depends on (α^+, α^-) only through the ratio α^+/α^-. ◀

When $\alpha^- > 0$, the decision can be parameterized by the ratio $\frac{\alpha^+}{\alpha^-}$. When $\frac{\alpha^+}{\alpha^-} < 1$, the denominator decreases with p, so experts with higher p_i^o contribute more to the aggregate. At $\alpha^- = 0$, the transfer is refunded when the outcome is negative, so the mechanism remains DSIC yet experts are never held accountable for pivotal decisions that lead to bad outcomes. Under the dominant strategy, the expert's net financial exposure is $\alpha^-|t_i|$ and the required escrow is invariant to α^-. When reducing α^-, experts compensate by submitting proportionally larger messages. The net exposure is always capped by the message the experts submit.

Parameterized Designer's Objective

The Designer's Problem from Section 3.4 – matching the sign of the aggregate message to the sign of the aggregate belief differential, as in (6) – can now be instantiated with the derived dominant strategy. The designer chooses the mechanism parameters $(\alpha^+, \alpha^-, \beta^+, \beta^-)$ to maximize the probability of correct classification for a given environment. The designer's problem:

$$\max_{\alpha^+,\alpha^-,\beta^+,\beta^-} \mathbb{P}_{v_i,p_i^A,p_i^B}\left(\operatorname{sgn}\left(\sum_{i=1}^{n}\frac{v_i + \Delta\beta(p_i^A - p_i^B)}{\alpha(p_i^o)}\right) = \operatorname{sgn}\left(\sum_{i=1}^{n}(p_i^A - p_i^B)\right)\right) \tag{11}$$

subject to

$$\alpha^+, \alpha^- \geq 0, \quad (\alpha^+, \alpha^-) \neq (0,0),$$
$$\beta^+, \beta^- \in [\bar{u},\ c/n]$$

for some random distributions of the agents types.

5.2 Simulations

Since the belief p_i^o enters the denominator of each expert's dominant strategy, and the value of o depends on the sign of $\hat{v}_i$, the message aggregate is a sum of ratios of dependent random variables with no known closed form. We therefore evaluate the mechanism's performance via Monte Carlo simulation: for each parameter configuration, we repeatedly draw random type profiles from specified distributions, compute every expert's dominant-strategy message, apply the allocation rule, and report *accuracy*: the fraction of the drawn profiles on which the mechanism's decision coincides with the objectively superior alternative – the one with the higher true success probability.

A direct empirical comparison with decision markets is not well-posed in our setting. A decision market's realized accuracy is not a function of the type distribution alone; it depends on the market microstructure (trader's budget, risk attitude, the market scoring rule, and the number and order of trading rounds), none of which our mechanism requires. Moreover, under a deterministic decision rule no decision market admits a dominant strategy in the multi-agent setting; restoring strict properness requires a randomized decision rule [5]. Quadratic voting and the standard pivotal mechanism are also excluded, because neither is able to extract beliefs from experts.

We therefore compare our mechanism against two decision rules: *majority voting with rewards* (MV w/ rewards), where each expert votes for A if $\hat{v}_i > 0$ and the majority wins; and *majority voting without rewards* (MV), where each expert votes for A if $v_i > 0$. All three rules are evaluated on the same drawn type profiles.

We further compare against a decision scoring rule (DSR). A DSR pays a single expert a positive affine function of the principal's realized utility [28]; with a binary outcome and principal utility $u = 1$ for a good outcome and $u = -1$ for a bad one, this rewards the expert for the realized impact of the recommended action. Since DSR requires a single expert, we implement the DSR as a *random dictator*: one expert is drawn uniformly at random and their recommended alternative is taken.

Setup

We specify a parametric model for the experts' types, defining distributions for idiosyncratic preferences and belief noise. The true success probabilities are symmetric around 0.5, separated by a *quality gap* parameter $\delta \in \{0.05, 0.15, 0.25\}$, corresponding to hard, moderate, and easy classification tasks, capturing the case the designer typically cares about most: one alternative is likely to succeed while the other is likely to fail. When both alternatives would succeed (p^A, p^B both high) or both would fail (p^A, p^B both low), the stakes of choosing correctly are lower. By the Symmetry property, we can without loss of generality let A be the superior alternative.

Idiosyncratic preferences are drawn independently with mean $\mu_v \in \{0, -1\}$, where $\mu_v = 0$ is the unbiased case and $\mu_v = -1$ represents a committee that on average opposes the superior alternative. The variance is normalized to one, so $\Delta\beta$ is measured in units of preference intensity. Each expert receives a noisy private signal centered at the truth with standard deviation $\sigma \in \{0.2, 0.8, 2.0\}$ (*belief noise*), truncated to $(0, 1)$. A small σ corresponds to a well-informed committee, while a large σ represents dispersed information. Each expert's type is drawn as:

$$v_i \sim \mathcal{N}(\mu_v, 1), \qquad p_i^A \sim \mathrm{TN}\big(\tfrac{1}{2} + \delta,\ \sigma^2,\ 0,\ 1\big), \qquad p_i^B \sim \mathrm{TN}\big(\tfrac{1}{2} - \delta,\ \sigma^2,\ 0,\ 1\big),$$

where $\mathrm{TN}(\mu, \sigma^2, a, b)$ denotes a truncated normal distribution on $(0, 1)$. Results with uniform preferences $v_i \sim U(\mu_v - 1, \mu_v + 1)$ are similar and omitted for brevity.

A grid search over $\alpha^+/\alpha^- \in [0, 2]$ reveals that the optimal ratio is consistently close to zero: the accuracy gap between $\alpha^+ = 0$ and the optimal α^+/α^- never exceeds 3% across all tested parameter configurations, so we adopt $\alpha^+ = 0$ as a default.

For each parameter configuration, we draw 10^6 independent type profiles. We vary $\Delta\beta \in [0, 5]$ on a grid of 30 equally spaced values and committee sizes $n \in \{5, 49\}$ (chosen odd to avoid ties in majority voting).

All simulations were implemented in Python with a fixed random seed. Table 3 summarizes the full set of varied parameters and their ranges.

Table 3 Parameters varied in the Monte Carlo simulations.

Parameter	Symbol	Values
Quality gap	δ	{0.05, 0.15, 0.25}
Preference bias	μ_v	{0, −1}
Belief noise	σ	{0.2, 0.8, 2.0}
Committee size	n	{5, 49}
Belief budget	$\Delta\beta$	[0, 5]

Majority Voting

Figures 2 and 3 show that our mechanism outperforms majority voting across almost every parameter configuration, the exception being the small quality gap ($\delta = 0.05$) in the large committee ($n = 49$), where the two are comparable. The margin is widest when the belief budget $\Delta\beta$ is small.

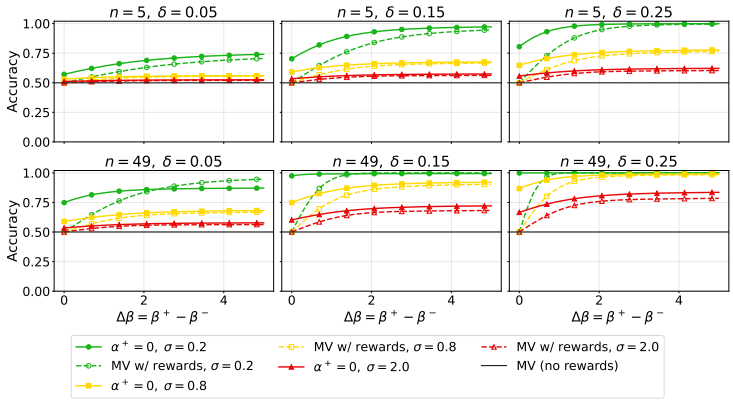


Figure 2 Unbiased committee ($\mu_v = 0$): classification accuracy as a function of $\Delta\beta$. Solid: our mechanism ($\alpha^+ = 0$). Dashed: MV with rewards. MV without rewards is the horizontal line at 0.5. Rows: committee size n. Columns: quality gap δ. Colors: belief noise σ. Preferences: $v_i \sim \mathcal{N}(\mu_v, 1)$.

At $\Delta\beta = 0$ the mechanism already reweighs experts through $1/\alpha(p_i^o)$, so beliefs scale each report even with no belief budget. For $\Delta\beta > 0$, messages are additionally pushed toward the expert's belief differential and accuracy rises steeply before flattening, as the belief signal comes to dominate preference noise. The quality gap δ and belief noise σ act as expected: a larger δ separates the true success probabilities and eases the task for every method, while a larger σ disperses the signals and degrades accuracy.

From a design standpoint, $\Delta\beta$ is bounded above by $c/n - \bar{u}$, so the designer can pick the minimum $\Delta\beta$ – and hence the minimum per-expert budget the DAO must commit – needed to reach a target accuracy.

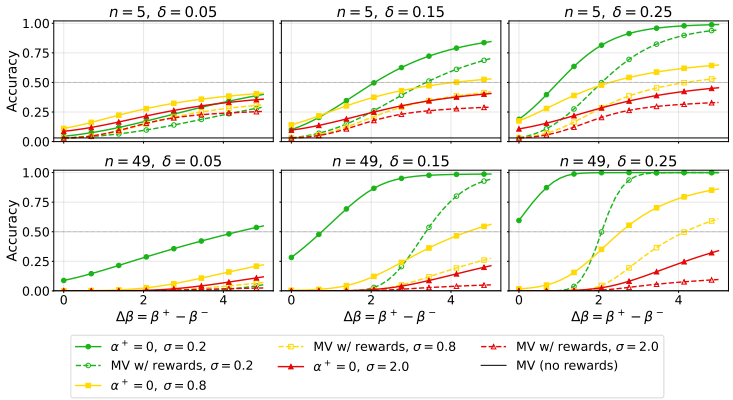


Figure 3 Biased committee ($\mu_v = -1$), where members on average prefer the inferior alternative. Conventions as in Figure 2, except for the MV with no rewards that has a value close to 0.

For the biased committee (Figure 3), majority voting without rewards has approximately 0 accuracy, and with rewards it keeps only the sign of the shifted valuation, discarding the magnitude. Our mechanism instead scales each report by $1/\alpha(p_i^o)$, so an expert backing an option they believe is unlikely to succeed carries little weight in the aggregate.

Results with uniform preferences $v_i \sim U(\mu_v - 1, \mu_v + 1)$ are qualitatively similar, with one exception. At $\mu_v = -1$ the support is $[-2, 0]$, so every expert prefers B: at $\Delta\beta = 0$ no expert ever votes for A, reweighting alone cannot shift the decision, and accuracy is zero – unlike the normal case, whose tails leave some experts with $v_i > 0$. Once $\Delta\beta > 0$ the belief signal flips some experts toward A and the mechanism recovers.

Random dictator

Figure 4 shows that in the unbiased committee our mechanism outperforms the DSR across every configuration.

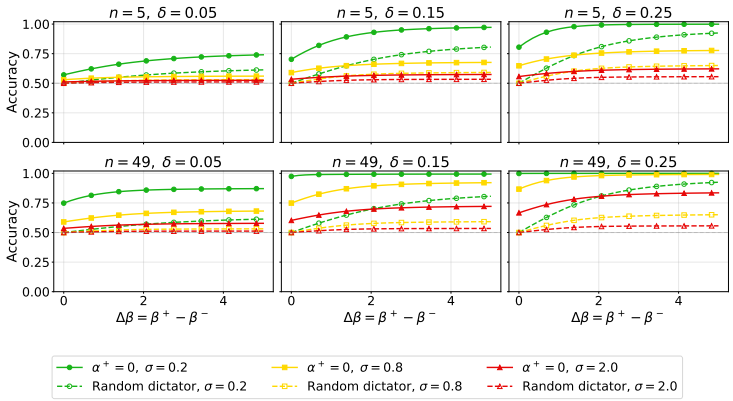


Figure 4 Unbiased committee ($\mu_v = 0$): classification accuracy of our mechanism ($\alpha^+ = 0$, solid) versus the random dictator with the same outcome-contingent incentives (dashed), as a function of $\Delta\beta$. Conventions as in Figure 2 σ. Preferences: $v_i \sim \mathcal{N}(\mu_v, 1)$.

In the biased committee (Figure 5) the DSR is more accurate at small $\Delta\beta$ in some scenarios.

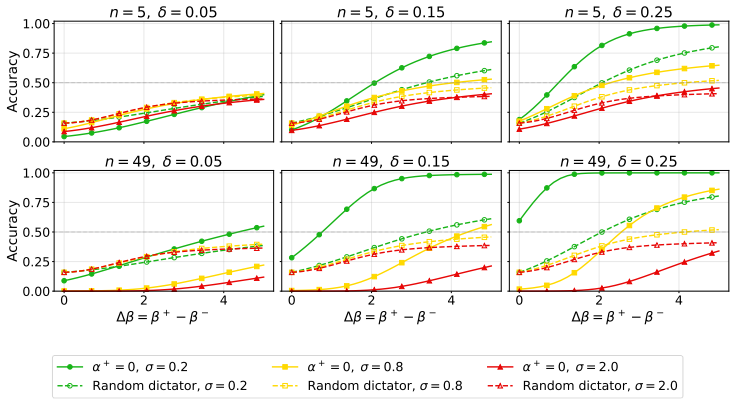


Figure 5 Biased committee ($\mu_v = -1$): our mechanism ($\alpha^+ = 0$, solid) versus the random dictator (dashed). Conventions as in Figure 2.

6 Conclusions

This paper presents a mechanism for committee decision-making with outcome-contingent transfers that satisfies dominant-strategy incentive compatibility, interim individual rationality, and budget feasibility. Each expert submits a single scalar that jointly encodes preferences and beliefs, and the mechanism maximizes the probability that its decision agrees with a linear opinion pool of the experts' private beliefs. The framework abstracts away the medium of exchange, the method of outcome evaluation, and the committee election process, providing a reference construction that practical designs can instantiate. The mechanism provides value even without a budget: the α reweighs experts, so informative beliefs can enter the aggregate, yielding higher classification accuracy than a standard pivotal mechanism at no additional cost. The budget amplifies this effect by adding an explicit belief shift to each expert's message.

When experts have no idiosyncratic preferences, the dominant strategy reduces to a scaled belief differential and the mechanism selects the alternative according to the linear opinion pool. This elicitation is strictly incentive compatible whenever each expert assigns non-zero probability to every possible aggregate of the other experts' reports. Decision scoring rules cannot achieve this in the multi-agent setting: under a deterministic decision rule, each expert's expected payoff depends on the collective decision, which in turn depends on the other experts' reports, so agents have an incentive to misreport to change the collective decision in their favor [29]. Restoring properness requires randomized decision rules [5], which introduce practical difficulties. Our mechanism sidesteps this impossibility by inducing a dominant strategy under a deterministic allocation rule and across multiple agents.

We focus on binary choices, which cover the majority of DAO proposals [36]. The designer's objective reduces to matching the sign of a linear opinion pool. Although the linear opinion pool generalizes naturally to k alternatives, the mechanism design side is less straightforward. With two alternatives, the classification boundary is a single threshold that maps to the pivotal mechanism's allocation rule. With $k \geq 3$, the designer must match an arg max over k pooled probabilities, and the interaction between outcome-contingent rewards and a multi-dimensional message space remains an open problem.

References

1 David Austen-Smith and Jeffrey S. Banks. Information aggregation, rationality, and the Condorcet jury theorem. *American Political Science Review*, 90(1):34–45, 1996.

2 Martin Becze, Hudson Jameson, et al. EIP-1: EIP purpose and guidelines. *Ethereum Improvement Proposals*, 27, 2015.

3 Craig Boutilier. Eliciting forecasts from self-interested experts: scoring rules for decision makers. *arXiv preprint arXiv:1106.2489*, 2011. arXiv:1106.2489.

4 Vitalik Buterin, Zoë Hitzig, and E. Glen Weyl. A flexible design for funding public goods. *Manage. Sci.*, 65(11):5171–5187, November 2019. doi:10.1287/mnsc.2019.3337.

5 Yiling Chen, Ian Kash, Mike Ruberry, and Victor Shnayder. Decision markets with good incentives. In *International Workshop on Internet and Network Economics*, pages 72–83. Springer, 2011. doi:10.1007/978-3-642-25510-6_7.

6 Lin William Cong, Daniel Rabetti, Charles CY Wang, and Yu Yan. Centralized governance in decentralized organizations. *Available at SSRN 5168660*, 2025.

7 DAOstack. DAOstack whitepaper: An operating system for collective intelligence. https://www.allcryptowhitepapers.com/daostack-whitepaper/, 2018. Accessed: 2026-03-14.

8 Luke Dashjr. BIP 2: BIP process, revised. https://github.com/bitcoin/bips/blob/master/bip-0002.mediawiki, 2016. Bitcoin Improvement Proposal.

9 David Davó, Javier Arroyo, Samer Hassan, and Silvia Semenzin. The rise and fall of DAOstack: lessons for decentralized autonomous organizations. *PeerJ Computer Science*, 11:e3320, 2025. doi:10.7717/PEERJ-CS.3320.

10 Nicolas de Condorcet. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Imprimerie Royale, Paris, 1785.

11 Nicola Dimitri. Voting in DAOs. *Distributed ledger technologies: Research and practice*, 2(4):1–12, 2023. doi:10.1145/3624574.

12 Allan Gibbard. Manipulation of voting schemes: A general result. *Econometrica*, 41(4):587–601, 1973.

13 Jerry Green and Jean-Jacques Laffont. *Incentives in public decision-making*. Elsevier North-Holland, 1979.

14 Jiasun Han, Jongsub Lee, and Tao Li. DAO governance. *Journal of Corporate Finance*, 2025.

15 Robin Hanson. Shall we vote on values, but bet on beliefs? *Journal of Political Philosophy*, 21(2):151–178, 2013.

16 John H Kagel and Dan Levin. The winner's curse and public information in common value auctions. *The American economic review*, pages 894–920, 1986.

17 Aggelos Kiayias and Philip Lazos. SoK: blockchain governance. In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, pages 61–73, 2022. doi:10.1145/3558535.3559794.

18 Stefan Kitzler, Stefano Balietti, Pietro Saggese, Hans-Martin Bernhard, and Markus Strohmaier. The governance of decentralized autonomous organizations: A study of contributors' influence, networks, and shifts in voting power. In *Financial Cryptography and Data Security (FC)*, 2024.

19 Steven P Lalley and E Glen Weyl. Quadratic voting: How mechanism design can radicalize democracy. In *AEA Papers and Proceedings*, volume 108, pages 33–37. American Economic Association 2014 Broadway, Suite 305, Nashville, TN 37203, 2018.

20 Clément Lesaege, Federico Ast, and William George. Kleros: Short paper v1.0.7, 2019. URL: https://kleros.io/whitepaper.pdf.

21 Hao Li, Sherwin Rosen, and Wing Suen. Conflicts and common interests in committees. NBER Working Paper 7158, National Bureau of Economic Research, 1999.

22 MetaDAO. MetaDAO, n.d. Accessed: 2026-03-13. URL: https://www.metadao.fi/.

23 Claudio Mezzetti. Mechanism design with interdependent valuations: Efficiency. *Econometrica*, 72(5):1617–1626, 2004.

24 Nolan Miller, Paul Resnick, and Richard Zeckhauser. Eliciting informative feedback: The peer-prediction method. *Management Science*, 51(9):1359–1373, 2005. doi:10.1287/MNSC.1050.0379.

25 Roger B. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58–73, 1981. doi:10.1287/MOOR.6.1.58.

26 Swaprava Nath and Onno Zoeter. A strict ex-post incentive compatible mechanism for interdependent valuations. *Economics Letters*, 121(2):321–325, 2013.

27 Shmuel Nitzan and Jacob Paroush. Optimal decision rules in uncertain dichotomous choice situations. *International Economic Review*, 23(2):289–297, 1982.

28 Caspar Oesterheld and Vincent Conitzer. Decision scoring rules. In *WINE*, page 468, 2020. URL: https://link.springer.com/content/pdf/10.1007/978-3-030-64946-3.pdf#page=471.

29 Abraham Othman and Tuomas Sandholm. Decision rules and decision markets. In *AAMAS*, pages 625–632, 2010. URL: https://dl.acm.org/citation.cfm?id=1838288.

30 Polkadot Wiki. Web3 and Polkadot, 2025. Accessed: 2026-03-12. URL: https://wiki.polkadot.com/general/web3-and-polkadot/.

31 Project Catalyst. Project catalyst documentation. https://docs.projectcatalyst.io/, n.d. Accessed: 2026-03-12.

32 Roopesh Ranjan and Tilmann Gneiting. Combining probability forecasts. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 72(1):71–91, 2010.

33 Olivier Rikken, Marijn Janssen, and Zenlin Kwee. Governance challenges of blockchain and decentralized autonomous organizations. *Information Polity*, 24(4):397–417, 2019. doi:10.3233/IP-190154.

34 T Sharma, Y Potter, K Pongmala, H Wang, A Miller, D Song, and Y Wang. Future of algorithmic organization: Large-scale analysis of decentralized autonomous organizations (DAOs). arXiv, 2024.

35 Nimrod Talmon. Social choice around decentralized autonomous organizations: On the computational social choice of digital communities. In *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1768–1773, 2023. doi:10.5555/3545946.3598837.

36 Qin Wang, Guangsheng Yu, Yilin Sai, Caijun Sun, Lam Nguyen, and Shiping Chen. Understanding DAOs: An empirical study on governance dynamics. *IEEE Transactions on Computational Social Systems*, PP:1–19, January 2025. doi:10.1109/TCSS.2025.3539889.

37 Robert L. Winkler, Yael Grushka-Cockayne, Kenneth C. Lichtendahl, and Victor Richmond R. Jose. Probability forecasts and their combination: A research perspective. *Decision Analysis*, 16(4):239–260, 2019. doi:10.1287/DECA.2019.0391.

Credible Auctions via MPC Gadgets: Bounding Information Leakage Under Abort

Matheus V. X. Ferreira
University of Virginia, Charlottesville, VA, USA

Abstract

The design of credible auctions – mechanisms where a revenue-maximizing auctioneer has no incentive to deviate from the protocol – faces a fundamental cryptographic barrier when the auctioneer controls shill bidders. While a natural approach is to use Secure Multi-Party Computation (MPC) to remove the trusted auctioneer, Cleve's (1986) impossibility of fair coin flipping implies that monolithic MPC protocols grant the auctioneer a "free option": they can learn the auction's outcome and unilaterally abort if the revenue is unsatisfactory. Previous attempts to mitigate this abort asymmetry using cryptographic commitments and ex-ante penalties fail for heavy-tailed distributions. We overcome this impossibility by introducing the *MPC Decomposition Principle*. Rather than encrypting the entire mechanism, we use MPC strictly as an information-restriction tool. We isolate the *winner determination problem* into a minimal MPC gadget that computes and reveals the winner's identity but no payment information. This qualitative restriction mathematically bounds the information leaked upon an abort. By combining this gadget with sequential revelation and finite economic penalties, we design the *Sequential Revelation Auction* (SRA). We prove that bounding the information leakage strictly bounds the value of the free option, showing that a penalty of $\kappa \geq \sum_{i=1}^{n} \textsc{Rev}(F_i)$ is sufficient for credibility, and tight in the worst case: for the equal-revenue distribution – where this threshold coincides exactly with the optimal revenue – every smaller penalty admits a profitable deviation. Leveraging low-round-complexity MPC, the SRA resolves an open question of [1] and [11] by providing the first constant-round, incentive-compatible, revenue-optimal credible auction for all product distributions.

2012 ACM Subject Classification Theory of computation → Algorithmic mechanism design; Security and privacy → Cryptography

Keywords and phrases Multi-Party Computation, Credible Auctions, Revenue-optimal Auctions, Cryptographic Auctions

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.18

Related Version *Full Version*: `https://arxiv.org/abs/2609.27402`

Acknowledgements I thank Justin Taylor and Vamsi Policharla for helpful discussions on constant-round MPC with identifiable abort via homomorphic encryption.

1 Introduction

The study of mechanism design traditionally assumes a trusted intermediary who faithfully implements the mechanism's rules. In practice – ranging from large-scale digital advertising to decentralized finance [4] – the auctioneer is a rational agent who may deviate if doing so is both undetectable and increases their revenue. The literature on *credible auctions*, initiated by [1], formalizes this problem. In the full information setting (where the auctioneer can observe all the actions from bidders), they proved a fundamental impossibility: the ascending price auction is the unique revenue-optimal, incentive-compatible auction without undetectable profitable deviations, a desideratum we call credibility. Unfortunately, this auction requires an unbounded amount of communication in the worst case.

To circumvent this impossibility, [11] proposed transforming the auction into a game of incomplete information using cryptography. Their *Deferred Revelation Auction (DRA)* requires bidders to cryptographically commit to their valuations, revealing them only after


© Matheus V. X. Ferreira;
licensed under Creative Commons License CC-BY 4.0

the auctioneer commits to the execution path. Moreover, participants deposit collaterals to incentivize opening of commitments. However, this approach hits a fundamental barrier when applied to bidder valuations drawn from heavy-tailed distributions (e.g., the equal-revenue distribution), where [11] proved that no finite ex-ante penalty can deter a malicious auctioneer. In the DRA, this failure occurs because the auctioneer's shill bidder can wait for honest bidders to reveal their commitments, observe their valuations, and then refuse to open its own commitment. This forced abort triggers a restart where the auctioneer now knows the bids. For heavy-tailed distributions, the expected revenue gained from exploiting this acquired knowledge is infinite, meaning no finite collateral can deter the abort.

A natural alternative is to abandon commit-reveal schemes and use Secure Multi-Party Computation (MPC) to encrypt the entire auction. However, this monolithic MPC approach hits a fundamental cryptographic barrier first identified by [8] in the context of fair coin flipping: the *abort problem*. Cleve established that in protocols without an honest majority (which includes an auctioneer commanding unbounded shill bids), an adversary can always learn their output and unilaterally abort the protocol before honest parties receive theirs.

In economic terms, Cleve's abort asymmetry grants the auctioneer a "free option". If the auctioneer encrypts the entire optimal mechanism (allocation and payments) within an MPC protocol, they can run the protocol, observe the provisional revenue, and if it is unsatisfactory, abort the execution and restart. Since the abort reveals that all active bidders have valuations below a certain threshold, the restart is not an independent trial but a biased one. This allows the auctioneer to safely explore the bidders' valuations using shills, effectively extracting full surplus and violating the credibility of the mechanism.

In this work, we introduce the *Sequential Revelation Auction (SRA)*, which resolves this impossibility and achieves credibility for *any* product distribution over bidder valuations. We overcome the free option problem by introducing the *MPC Decomposition Principle*. Rather than following the traditional paradigm of "encrypting everything," we use MPC strictly as an *information-restriction tool*. Instead of computing the entire mechanism inside the MPC, we isolate the smallest possible piece of information necessary: the identity of the winner. We introduce a *Winner Determination Gadget (WDG)*, a minimal MPC functionality that takes committed bids and solves the Winner Determination Problem (WDP) by outputing *only* the identity of the winner, revealing *no payments* or *bids*.

This decomposition shifts the information asymmetry. If the auctioneer uses a shill to probe the price and subsequently aborts the gadget, they learn only *who* was winning, but not *how much* bidders were willing to pay. We formally quantify the value of this leaked information. Our main technical contribution, the Information Leakage Lemma (lemma 32) bounds the extra revenue the auctioneer can extract from the knowledge acquired during an abort. Therefore, by setting an ex-ante abort penalty κ to at least this threshold, we ensure that the cost of aborting always exceeds the value of the free option.

1.1 Technical Overview

We assume the existence of an MPC protocol with identifiable abort that implements any ideal functionality f. This protocol allows a set of parties to compute any ideal functionality $f(v_1, \ldots, v_n) = (y_1, \ldots, y_n)$ where v_i is the input that player i contributes to evaluating f and y_i is the output only he observes. A key feature of this model is that any player i can abort the execution at any time learning y_i and preventing others from learning y_{-i}; however, if they do so, all other parties learn the identity of the deviating player.

In the SRA, we instantiate f as the WDP: given a set of bids $\boldsymbol{v}$, output some $i^* \in \arg\max_i v_i$. Then, we ask the winner to reveal v_{i^*}. It is not hard to see that the only undetectable deviation available to the auctioneer is to modify $\boldsymbol{v}$ by adding a set shill bids

$r_1 < r_2 < \cdots < r_k$ into the input and if chosen as the winner, to refuse to reveal v_{i^*} in which case the protocol recomputes the WDP without i^*. The core logic of our credibility proof relies on an "unraveling" argument to argue that this abort does not leak sufficient information to violate credibility. By bounding the information leaked during such an abort, we show that setting the penalty $\kappa \geq \sum_{i=1}^{n} \textsc{Rev}(F_i)$ ensures that aborting is never profitable where $(F_1, \ldots, F_n)$ is the value distribution profile of real bidders and $\textsc{Rev}(F_i)$ is the optimal revenue (lemma 7) an auctioneer can obtain in a single-bidder auction with a bidder with valuation drawn from F_i.

1.2 Our Contribution

We provide the first constant-round, incentive-compatible, revenue-optimal auction that is credible for all product distributions over bidder valuations, resolving the open problem left by [11]. Our contributions are threefold:

1. **The MPC Decomposition Principle:** We introduce a new design methodology for credible auctions based on *decomposing* the auction computation. Rather than implementing the entire mechanism (allocation and payments) inside MPC – which, as we show in Section 3, inevitably leaks enough information to violate credibility – we isolate the minimal subroutine needed: a Winner Determination Gadget that computes *only* the Winner Determination Problem. Payments are then determined *outside* the MPC through sequential revelation. This decomposition principle is the key conceptual contribution: it shows that for credibility, minimizing the computed function is more important than encrypting everything.
2. **General Credibility:** We prove that for any product distribution $\boldsymbol{F}$, the SRA is revenue-optimal, incentive-compatible, and credible provided the penalty κ satisfies $\kappa \geq \sum_{i=1}^{n} \textsc{Rev}(F_i)$ (Theorem 39). This overcomes the fundamental barrier identified by [11]: for heavy-tailed distributions (e.g., the equal-revenue distribution) where they exhibit a single non-MHR bidder for which *no* finite penalty makes the Deferred Revelation Auction credible. Our protocol circumvents this impossibility by using MPC to change the information structure of the game.
3. **Tightness and Round Complexity:** We show that the penalty bound $\kappa \geq \sum_{i=1}^{n} \textsc{Rev}(F_i)$ is tight: for the equal-revenue distribution, any penalty $\kappa < \sum_{i=1}^{n} \textsc{Rev}(F_i)$ admits profitable deviations (Theorem 40). Additionally, the SRA runs in constant rounds on the equilibrium path (assuming standard constant-round MPC constructs). This complements the ascending protocol of [10], which achieves general credibility for n i.i.d. bidders in constant rounds *in expectation* but has unbounded worst-case round complexity. Our guarantee differs along two axes: the round bound on the equilibrium path is deterministic rather than in expectation, and our credibility guarantee holds for arbitrary *independent, non-identical* distributions $F_1, \ldots, F_n$ rather than the i.i.d. case.

Practical Collateral Requirements

The threshold $\kappa \geq \sum_{i=1}^{n} \textsc{Rev}(F_i)$ is practically reasonable, not merely finite. On the canonical stress test – the equal-revenue distribution $F(v) = 1 - 1/v$ for $v \geq 1$, where [11] showed the Deferred Revelation Auction needs *infinite* collateral even against a single bidder – the SRA is credible at $\kappa = 1$ per bidder, and since $v \geq 1$ on the support, no bidder ever posts more than their own valuation. Feasibility gives $\textsc{Opt}(\boldsymbol{F}) = \mathbb{E}\left[\max_i \bar{\phi}_i(v_i)^+\right] \leq \sum_{i=1}^{n} \mathbb{E}\left[\bar{\phi}_i(v_i)^+\right] = \sum_{i=1}^{n} \textsc{Rev}(F_i)$, so the requirement is $\textsc{Opt}(\boldsymbol{F}) \leq \kappa \leq n \cdot \textsc{Opt}(\boldsymbol{F})$: between one and n times the revenue the auction generates. It is also a *liquidity* requirement rather than an expected cost, since every bidder who follows the protocol recovers κ in full.

1.3 Related Work

Secure Multi-Party Computation

Secure Multi-Party Computation (MPC) was introduced by [19] with the "Millionaires' Problem" and [3] with "Coin Flipping by Telephone. " The first large-scale commercial application of MPC was the Danish Sugar Beet Auction [4], where farmers traded production contracts. In this system, robustness against bidder dropout was handled by re-running the computation or assuming the servers were honest. This approach to *safety* was acceptable because the threat model did not involve a malicious auctioneer. In contrast, our work focuses on *credibility*, where the auctioneer is the primary adversary. We show that the restart mechanism, essential for safety, becomes a vulnerability for credibility. For our auction to have small round complexity, we rely on works showing that any functionality can be realized in a constant number of rounds. Two rounds suffice for security *with abort* [13, 16], relying on powerful cryptographic primitives like *Multi-Key Fully-Homomorphic Encryption* (MK-FHE) and *indistinguishability obfuscation* (iO). Our protocol requires the stronger guarantee of *identifiable* abort, since the penalty must be charged to a named party, and identifiability provably costs additional rounds: [7] show that four rounds are necessary and sufficient in the plain model. Constant-round protocols with identifiable abort are known [15, 2, 9], which is all our result requires. While these primitives are currently computationally expensive, they establish the theoretical feasibility of avoiding the high round complexity of previous credible mechanisms [10].

Credible Auctions

The credibility framework was established by [1], who showed that unconditional non-trivial credible auctions are impossible. [11] introduced cryptographic assumptions to the problem, constructing two-round revenue optimal credible auctions for MHR distributions. However, their negative result for heavy-tailed distributions left open the question of general feasibility. Our work resolves this open problem. [10] also achieved general credibility, for n i.i.d. bidders with a finite monopoly price, in constant rounds in expectation but with unbounded worst-case round complexity. In contrast, our SRA runs in constant rounds (assuming standard constant-round MPC constructs) on the equilibrium path, and handles independent but non-identical distributions. Like theirs, our worst-case round complexity is unbounded: the number of rounds grows linearly with the number of bidders that abort, but aborting is a dominated strategy for all bidders (real or shills).

[6] extended DRA to settings where bidders can broadcast messages, which is relevant to our MPC-based approach since MPC with identifiable abort requires broadcast channels. [12] extended the DRA to multi-item settings with matroid constraints but also require similar distributional assumptions to [11] and [10].

There is a rich literature on implementing auction mechanisms using MPC [18, 5]. Most of this work focuses on *privacy* (keeping bids secret) or *fairness* (preventing early bid peeking). Our focus on *credibility* is distinct: we view the auctioneer as the adversary and specifically target the strategic implications of the abort capability. Our result highlights that for credibility, "encrypting everything" is not the solution; rather, careful information management via minimizing the computed function is key.

1.4 Paper Organization

Section 2 defines the credibility framework. Section 3 illustrates the failure of black-box MPC. Section 4 details the SRA protocol. Section 5 provides a warm up by showing that the SRA is credible for the equal-revenue distribution. This addresses the open problem of [11]. Section 6 bounds the information an abort leaks, and Section 7 uses that bound to prove credibility for every product distribution, which addresses the open problem of [11]. Section 8 provides the matching lower bound. Proofs omitted for space, together with a worked equal-revenue warmup, appear in the full version.

2 Preliminaries

We consider a single-item auction setting with a set of honest, risk-neutral bidders $N = \{1, \dots, n\}$. Additionally, there exists a countable set of potential shill bidders $S = \{n+1, n+2, \dots\}$. Honest bidders $i \in N$ have private valuations $v_i \in V_i \subseteq \mathbb{R}_{\geq 0}$ drawn independently from cumulative distribution functions F_i with probability density functions f_i. Shill bidders $s \in S$ have a fixed valuation $v_s = 0$. Crucially, shill bidders share the auctioneer's utility function (revenue maximization) and do not behave as self-interested agents maximizing their own allocation utility. We denote the valuation profile of honest agents by $\boldsymbol{v} = (v_1, \dots, v_n) \in \boldsymbol{V} = \times_{i \in N} V_i$. For value profile $\boldsymbol{v} = (v_i)_{i \in N \cup S}$ and subset $A \subseteq N \cup S$, we write $\boldsymbol{v}_A = (v_i)_{i \in A}$ for the restriction of $\boldsymbol{v}$ to bidders in A. For any function f, we write $f^+ := \max\{f, 0\}$ for the positive part.

We begin by formally defining the core concepts of optimal auctions.

▶ **Definition 1** (Mechanism). *A direct revelation mechanism $M = (x, p)$ consists of an* allocation rule $x : \boldsymbol{V} \to [0,1]^n$, *where $x_i(v)$ is the probability that bidder i receives the item and feasibility requires $\sum_{i \in N} x_i(v) \leq 1$ for all $v \in \boldsymbol{V}$, together with a* payment rule $p : \boldsymbol{V} \to \mathbb{R}^n$, *where $p_i(v)$ is the expected payment from bidder i.*

We assume agents have quasi-linear utility functions.

▶ **Definition 2** (Utility and DSIC). *The utility of bidder i with valuation v_i who reports v_i' while the others report v_{-i} is $u_i(v_i', v_{-i}; v_i) = v_i \cdot x_i(v_i', v_{-i}) - p_i(v_i', v_{-i})$. A mechanism is* Dominant Strategy Incentive Compatible (DSIC) *if truthful reporting maximizes utility for every honest player $i \in N$, valuation v_i, and profile v_{-i}: $u_i(v_i, v_{-i}; v_i) \geq u_i(v_i', v_{-i}; v_i)$ for all $v_i' \in V_i$.*

▶ **Theorem 3** ([17]). *A direct revelation mechanism (x, p) is DSIC if and only if:*

1. **Allocation rule is monotone.** *For every bidder i and declarations v_{-i}, $x_i(v_i, v_{-i})$ is non-decreasing in v_i.*
2. **Payment identity.** *The payment rule is determined by the allocation rule: for every bidder i, v_i, and v_{-i}, $p_i(v_i, v_{-i}) = v_i \cdot x_i(v_i, v_{-i}) - \int_0^{v_i} x_i(z, v_{-i}) dz$.*

▶ **Definition 4** (Revenue). *The* ex-post revenue *of a mechanism $M = (x, p)$ for valuation profile $\boldsymbol{v}$ is the sum of payments, $R^M(\boldsymbol{v}) = \sum_{i \in N} p_i(\boldsymbol{v})$; its* expected revenue *is $\textsc{Rev}^M(\boldsymbol{F}) = \mathbb{E}_{\boldsymbol{v} \sim \boldsymbol{F}}\left[R^M(\boldsymbol{v})\right]$, and we write $R(\cdot)$ for short when M is clear from context. We write $\textsc{Opt}(\boldsymbol{F}) = \max_{M \ DSIC} \textsc{Rev}^M(\boldsymbol{F})$ for the optimal expected revenue over DSIC mechanisms, and, for a* single *distribution F_i, $\textsc{Rev}(F_i) = \sup_{p \geq 0} p \cdot (1 - F_i(p))$ for the optimal* monopoly *revenue obtainable from bidder i alone by posting a price. Throughout, $\textsc{Rev}(\cdot)$ applied to a single distribution always denotes this monopoly revenue, while revenue of a mechanism always carries the mechanism in the superscript.*

Throughout we assume each F_i has a vanishing revenue tail, $\lim_{v\to\infty} v\,(1-F_i(v)) = 0$; this is the hypothesis of Lemma 7 below, and it implies $\textsc{Rev}(F_i) < \infty$, so the penalty $\sum_i \textsc{Rev}(F_i)$ is finite.

▶ **Definition 5** (Regular Distribution). *A differentiable cumulative distribution function F with density f is said to be* regular *if the virtual valuation function $\phi(v) = v - \frac{1-F(v)}{f(v)}$ is monotone non-decreasing.*

▶ **Definition 6** (Ironed Virtual Valuation). *For any distribution F, the* ironed virtual valuation *function $\bar{\phi}(v)$ is constructed by applying Myerson's ironing procedure to $\phi(v)$ to ensure monotonicity. Specifically, let $H(q) = \int_0^q \phi(F^{-1}(t))dt$ for $q \in [0,1]$. We define G as the convex hull of H, i.e., the largest convex function such that $G(q) \le H(q)$ for all q. Then, $\bar{\phi}(v) = G'(F(v))$. If F is regular, $\bar{\phi}(v) = \phi(v)$. We define the* inverse $\bar{\phi}^{-1}(y) := \inf\{v : \bar{\phi}(v) \ge y\}$*, which is well-defined even when $\bar{\phi}$ has flat regions due to ironing.*

▶ **Lemma 7** (Virtual surplus identity; [17]). *For any DSIC mechanism M allocating a single item to agents with independent private values drawn from distributions F_i satisfying $\lim_{v\to\infty} v\,(1-F_i(v)) = 0$, the expected revenue equals the expected virtual surplus:*

$$\textsc{Rev}^M(\boldsymbol{F}) = \mathbb{E}_{\boldsymbol{v}\sim\boldsymbol{F}}\left[\sum_i \bar{\phi}_i(v_i)x_i(\boldsymbol{v})\right]$$

where $\bar{\phi}_i(v_i)$ is the ironed virtual valuation function of bidder i.

Myerson's result identifies the revenue-optimal auction as the mechanism that allocates the item to the bidder with the highest ironed virtual value, provided it is non-negative. Ironing makes ties a positive-measure event rather than a null one – $\bar{\phi}_i$ is flat on each ironed interval, so every profile landing inside one produces a tie – and the Winner Determination Gadget of Section 4 publishes which of the tied parties is provisionally winning. We therefore fix the comparison and the tie-break together, as a single ranking, rather than treating the latter as an afterthought.

▶ **Definition 8** (Ranking). *For* participants *$i \ne j$ submitting bids b_i, b_j, we write $\bar{\phi}_i(b_i) < \bar{\phi}_j(b_j)$ to mean that the mechanism ranks j above i: either $\bar{\phi}_i(b_i)$ is numerically smaller than $\bar{\phi}_j(b_j)$; or the two are numerically equal and $b_i < b_j$; or both of these are equalities and $j < i$. We write $\le$ for the reflexive version. A higher ironed virtual value therefore wins, ties go to the higher raw bid, and any remaining tie goes to the smaller index; the last clause separates any two distinct participants, so this is a strict total order and every finite set of participants has a unique maximum.*

Two conventions keep this unambiguous. First, the overload applies only between two participants*' virtual values; a comparison against a constant, such as $\bar{\phi}_i(v_i) \ge 0$, is always the ordinary numeric one. Second, the relation is determined by the pairs (i, b_i) and (j, b_j) rather than by the two numbers alone, so it is antisymmetric even where the numbers coincide: $\bar{\phi}_i(b_i) \le \bar{\phi}_j(b_j)$ and $\bar{\phi}_j(b_j) \le \bar{\phi}_i(b_i)$ hold together only when $i = j$. The bids and identities are always clear from context.*

The ranking is monotone in a participant's own bid*: for $z < z'$ we have $\bar{\phi}_i(z) \le \bar{\phi}_i(z')$ numerically because $\bar{\phi}_i$ is non-decreasing, and either that inequality is strict, in which case the first clause ranks z' above z, or the two agree numerically and the second clause does.*

▶ **Definition 9** (Myerson mechanism). *The mechanism allocates to the highest-ranked participant whose ironed virtual value is non-negative, $x_i(\boldsymbol{b}) = \mathbb{1}_{\bar{\phi}_i(b_i)\ge 0} \cdot \mathbb{1}_{\bar{\phi}_j(b_j) < \bar{\phi}_i(b_i)\ \forall j\ne i}$, and charges the winner the threshold payment induced by that rule, $p_i(\boldsymbol{b}) = \inf\{z \ge 0 : x_i(z, \boldsymbol{b}_{-i}) = 1\}$, losers paying nothing.*

Winning under Definition 9 requires i to outrank the fixed best competitor and to satisfy $\bar{\phi}_i(b_i) \geq 0$; both conditions are upward closed in i's own bid by the monotonicity of Definition 8, so the allocation rule is monotone and, with its threshold payments, the mechanism is DSIC by Theorem 3.

We now turn from the economic setting to the protocol design problem. The key tension is between *safety* (the protocol should handle dropouts gracefully) and *credibility* (the auctioneer should not be able to exploit the protocol). Our definitions make this tension precise.

2.1 Credibility

In this section, we formalize the notion of credibility we consider. Intuitively, a mechanism is credible if the auctioneer cannot profit by deviating from the protocol in a way that is indistinguishable from honest behavior.

▶ **Definition 10** (Auction Protocol). *An* auction protocol *is a pair* (M, Π)*, where* $M = (x, p)$ *is a mechanism and* Π *is a cryptographic protocol that purportedly realizes* M*.*

▶ **Definition 11** (Safe Deviation [1]). *A deviation by the auctioneer in protocol* Π *is* safe *if, for every honest bidder* i*, there exists a strategy profile of the other bidders* $N \cup S \setminus \{i\}$ *such that* i*'s view under the deviation is computationally indistinguishable from* i*'s view under the honest execution of* Π *with that strategy profile.*

To illustrate, consider a second-price auction. The deviation where the auctioneer shill bids is a*safe* deviation since from any real bidder's perspective, the shill is indistinguishable from a real bidder. In contrast, an *unsafe* deviation would be one where the auctioneer refuses to allocate the item to the winner. This aims to capture an auctioneer that is willing to deviate as long as deviations cannot be detected by honest bidders.

▶ **Definition 12** (Credibility). *An auction protocol* (M, Π) *is* credible *if there exists no safe deviation for the auctioneer within* Π *that yields a strictly higher expected revenue than the honest execution of* M *without shills.*

We can now state our primary design goal:

▶ **Definition 13** (Optimal Credible Auction Design Problem). *The goal is to find an auction protocol* (M, Π) *maximizing* $\textsc{Rev}(M)$ *subject to* M *being DSIC for the honest bidders* N *and* (M, Π) *being credible.*

One further requirement is what makes that problem hard, and it is the reason the strawman of Section 3 fails. A protocol must remain usable when a participant simply stops responding.

▶ **Definition 14** (Safety against Honest Abort). *A protocol* Π *satisfies* Safety against Honest Abort *if for any set of bidders* P*, if a non-auctioneer player* $i \in P$ *aborts, the protocol can continue and the expected revenue – that is, the expected sum of payments from the remaining bidders – equals the expected revenue of the mechanism executed on the set* $P \setminus \{i\}$*.*

To satisfy Definition 14, a protocol must ensure that a single bidder's refusal to participate does not destroy the auction's value, which in practice requires it to restart or to switch to a mode that ignores the aborting party's input. Paradoxically, this creates the following vulnerability: it hands the auctioneer a "free option" to abort (perhaps through a shill) whenever the outcome is unfavorable, knowing the protocol *must* offer a recourse that preserves revenue – so the auctioneer retries the auction having learned the outcome of the first attempt. Section 3 turns this observation into a concrete example.

2.2 Cryptography

This section introduces basic cryptographic definitions used in the paper.

▶ **Definition 15** (Cryptographic Commitment). *A commitment scheme lets a committer publish $c = \mathit{Commit}(v, r)$, a* commitment *to a message v under randomness r. We require it to be* computationally hiding *– for distinct v, v' the distributions $\{\mathit{Commit}(v, r)\}_r$ and $\{\mathit{Commit}(v', r)\}_r$ are computationally indistinguishable –* perfectly binding *– no two pairs $(v, r) \neq (v', r')$ with $v \neq v'$ satisfy $\mathit{Commit}(v, r) = \mathit{Commit}(v', r')$ – and* non-malleable*, so that from c no adversary can produce a commitment whose value bears a non-trivial relation to v. See [14] for the formal definitions.*

A *multi-party computation (MPC) with identifiable abort* is a protocol that realizes an ideal functionality $\mathcal{F}$ that takes inputs $\boldsymbol{z} = (z_1, \ldots, z_n) \in (\mathcal{Z} \cup \{\perp\})^n$ from n parties and computes a function $f(\boldsymbol{z}) = (y_1, \ldots, y_n)$, where $y_i \in \mathcal{Y} \cup \{\perp\}$. An input of $\perp$ means that the party contributed with no input, and an output of $\perp$ means that the party gets no output.

While an ideal MPC would guarantee output delivery to all parties, this is known to be impossible in the presence of a dishonest majority [8]. To build intuition, consider the simple "Coin Flipping" problem where two parties want to generate a uniform random bit $r = r_1 \oplus r_2$. In the ideal world, a trusted party receives r_1, r_2 and sends r to both. In any real protocol, one party must send their value (or open their commitment) last. If this last party sees the result and doesn't like it (e.g., they wanted Heads but the result is Tails), they can simply abort. This allows the last party to bias the output (forcing an abort is equivalent to rejecting the coin flip). This impossibility result implies that in any auction protocol with a potentially malicious auctioneer (dishonest majority), we must allow for the possibility of *abort*.

This is particularly relevant in the context of auctions, where the auctioneer is allowed to control any number of shill bidders. We therefore rely on the standard relaxation of security with *identifiable abort*, which guarantees that if the adversary prevents honest parties from learning the output, their identity is revealed.

▶ **Definition 16** (Ideal Functionality). *Let $\mathcal{F} : \mathcal{Z}^n \to \mathcal{Y}^n$ be an n-party function. Ideally, a trusted party computes $(y_1, \ldots, y_n) = \mathcal{F}(z_1, \ldots, z_n)$ and sends y_i to party P_i.*

We use the standard simulation-based notion of security with identifiable abort, in which the adversary may abort at any point during the execution; we state the ideal experiment explicitly, since the timing of the abort is what drives our incentive analysis.

▶ **Definition 17** (Ideal model with identifiable abort). *Fix an n-party ideal functionality $\mathcal{F}$ and a set $I \subset [n]$ of corrupted parties. In the* ideal execution*, a simulator $\mathcal{S}$ controlling the parties in I interacts with the trusted party computing $\mathcal{F}$ as follows.*

1. *Each honest party $i \notin I$ sends its input z_i to the trusted party; $\mathcal{S}$ sends inputs $\{z_i\}_{i \in I}$ of its choice (possibly $\perp$).*
2. *The trusted party computes $(y_1, \ldots, y_n) = f(\boldsymbol{z})$ and sends $\{y_i\}_{i \in I}$ to $\mathcal{S}$.*
3. *$\mathcal{S}$ replies with either* continue*, in which case each honest party i receives y_i; or* (abort, i^*) *for some $i^* \in I$ of its choice, in which case every party outputs $(\perp, i^*)$.*

▶ **Definition 18** (Realization with Identifiable Abort). *A protocol Π* securely realizes *$\mathcal{F}$* with identifiable abort *if for every probabilistic polynomial-time adversary $\mathcal{A}$ statically corrupting a set $I \subset [n]$ in the real execution of Π, there exists a probabilistic polynomial-time simulator $\mathcal{S}$ in the ideal model of Definition 17 such that the joint distribution of the honest parties' outputs and the adversary's view is computationally indistinguishable in the two executions. See [14] and [15] for the formal treatment.*

Note that Definition 17 already encodes the worst case for our analysis: the adversary learns its own outputs $\{y_i\}_{i \in I}$ *before* deciding whether honest parties receive theirs, and an abort names a corrupted party. Every deviation we analyze in Section 7 is a choice of continue versus (abort, i^*) at Step 3 of Definition 17, so no generality is lost by restricting attention to that decision.

▶ **Theorem 19** ([15, 2, 9]). *Under standard cryptographic assumptions and the availability of a broadcast channel, for any ideal functionality $\mathcal{F}$, there exists a protocol Π that securely realizes $\mathcal{F}$ with identifiable abort in the presence of a static adversary controlling any number of corrupted parties $t < n$ in a* constant *number of rounds of communication.*

3 Strawman

In this section, we analyze the credibility of a straightforward attempt to realize a revenue-optimal auction using generic Multi-Party Computation (MPC). We define the ideal functionality $\mathcal{F}_{OPT}$ for the single-item Myerson auction – which receives bids, computes the optimal allocation and payments, and distributes the results – and demonstrate that any protocol realizing $\mathcal{F}_{OPT}$ while satisfying *Safety against Honest Abort* (Definition 14) is vulnerable to a welfare-extraction attack by a malicious auctioneer.

Algorithm 1 Optimal Auction Functionality $\mathcal{F}_{OPT}$.

Input: Bid profile $\boldsymbol{b} = (b_1, \ldots, b_n)$ for bidders P.
Computation:
1. Determine allocation x:

$$x_i(\boldsymbol{b}) = \begin{cases} 1 & \text{if } i = \operatorname{argmax}_{j \in [n]} \{\bar{\phi}_j(b_j) \mid \bar{\phi}_j(b_j) \geq 0\} \\ 0 & \text{otherwise} \end{cases}$$

2. Compute payment p:

$$p_i(\boldsymbol{b}) = \begin{cases} \inf\{b_i : \bar{\phi}_i(b_i) \geq \max_{j \neq i}\{0, \bar{\phi}_j(b_j)\}\} & \text{if } x_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

return $(\boldsymbol{x}, \boldsymbol{p})$.

We now consider a protocol Π that realizes $\mathcal{F}_{OPT}$. To satisfy *Safety against Honest Abort* (Definition 14) – ensuring that a single bidder's dropout allows the auctioneer to recover the revenue from the remaining bidders – practical implementations (like [4]) must be able to exclude the aborting party and complete the auction (e.g., via restart).

Algorithm 2 Protocol that realizes $\mathcal{F}_{OPT}$ with Safety Restart.

Input: Bid profile $\boldsymbol{b} = (b_1, \ldots, b_n)$ for bidders P.
Input: MPC protocol Π that realizes $\mathcal{F}_{OPT}$ with identifiable abort
Round 1:
Parties invoke Π on inputs $\boldsymbol{b}$.
if Π *outputs* $\perp$ *due to abort by* $k \in P$ **then**
  Identify aborter k.
  Update $P \leftarrow P \setminus \{k\}$.
  Restart round 1 with remaining players P.
end
return $(\boldsymbol{x}, \boldsymbol{p})$ of Π to all participants.

▶ **Theorem 20.** *Let* Π *be any protocol that realizes* $\mathcal{F}_{OPT}$*. If* Π *satisfies* Safety against Honest Abort*, then there exists a safe deviation for the auctioneer that can extract the full social welfare.*

Proof. The core idea is that the auctioneer uses one shill to "probe" the payment – which in the Myerson mechanism reveals the bidder's valuation – and then aborts to exploit this information in a restart.

Consider a setting with a single bidder with valuation v drawn from a distribution F with support $[\underline{v}, \bar{v}]$ (or $[\underline{v}, \infty)$). Suppose the auctioneer controls two shills S_1 and S_2. In the first execution, S_1 bids $\underline{v}$ and S_2 bids a value B exceeding the upper bound of the support of F. Since S_2 has the highest bid, S_2 wins the auction. Upon learning the output, the auctioneer observes the payment charged to S_2, which equals v (the threshold at which S_2's virtual value exceeds the honest bidder's). The auctioneer then instructs S_2 to abort. By safety, the protocol restarts with S_2 excluded but S_1 still active. Since Algorithm 2 restarts from scratch, S_1 may submit a fresh bid. The auctioneer now sets S_1's bid to $v - \epsilon$ for arbitrarily small $\epsilon > 0$, extracting $v - \epsilon$ from the honest bidder. ◀

This impossibility result reveals the fundamental weakness of monolithic MPC for auction design: the protocol reveals the *entire outcome* (winner and price) to the adversary before honest parties can act. In particular, the adversary learns the highest honest valuation through the payment identity, and the safety restart mechanism provides a costless option to exploit this information.

Why committing bids does not help

One might attempt to fix this by requiring bids to be cryptographically committed, so that the auctioneer cannot modify shill bids after learning the outcome. However, this defense is easily circumvented: the auctioneer commits to k shill bidders with bids $1, 2, \ldots, k$ and aborts all shills bidding above the highest honest bidder, effectively probing the honest valuation at no cost.

In the next section, we present the Sequential Revelation Auction, which avoids this vulnerability by decomposing the computation into minimal gadgets that reveal only the winner identity – never the price – and by imposing penalties that make aborting unprofitable.

4 The Protocol

The strawman analysis (Section 3) revealed that treating the MPC as a black box that realizes an auction functionality is fundamentally flawed for credibility: the adversary learns the full outcome upon abort, providing a "free option" to retry with extra information. Our protocol takes a fundamentally different approach. Rather than compute the entire auction mechanism – allocation *and* payments – inside MPC, we decompose the computation into a minimal *gadget* that reveals only the winner identity, and we compute payments *outside* the MPC via simultaneous revelation of committed bids. This design ensures that an abort reveals no payment information, and economic penalties make aborting unprofitable.

The key insight behind our design is that the auctioneer's "free option" in the strawman protocol stems from learning payment information upon abort. If the MPC computed only the winner identity – but not the payment – an abort would reveal who is winning, but not how much they would pay. Since payments, not allocations, directly determine revenue, this limited information alone is not enough for the auctioneer to construct a profitable deviation, provided the restart is costly. We formalize this idea via the *Winner Determination Gadget*.

▶ **Definition 21** (Winner determination gadget). *Let* $\mathbf{c} = (c_i)_{i \in A}$ *be a vector of commitments, broadcast by the bidders in the commit phase,* before *the gadget is invoked (Step 3 of Algorithm 3). The* Winner Determination Gadget $\mathcal{F}_{\mathcal{WDG}}{}^{\mathbf{c}}$ *is the ideal functionality, parameterized by* $\mathbf{c}$ *and by the Myerson-optimal allocation rule* x^**, that behaves as follows:*

1. ***Input:*** *Each party* $i \in A$ *submits a private input* (v_i, ρ_i)*.*
2. ***Computation:*** *Let* $A' = \{i \in A : c_i = \mathit{Commit}(v_i, \rho_i)\}$ *be the set of parties whose opening is consistent with their commitment. Parties in* $A \setminus A'$ *are treated as having contributed no input. The functionality computes the allocation* $x^*(\boldsymbol{v}_{A'})$ *and the identity* i^* *of the winner, if any.*
3. ***Output:*** *Every party* $i \in A$ *receives the pair* $\big(i^*, A \setminus A'\big)$*, where* $i^* = \perp$ *if the item is unallocated.*

Two features of Definition 21 deserve comment. First, the winner's identity is a *public* output, which the protocol needs – Algorithm 3 must detect when the provisional winner refuses to reveal, and the penalty can only be charged to a party the honest bidders can name – and which is harmless for credibility: wherever a real bidder wins, every shill is outbid and revealing weakly dominates aborting (Lemma 26), so the auctioneer never acts on the extra information and Lemma 32 is unaffected. Second, the functionality does not itself abort. Commitment verification is a computation on the parties' inputs, not a protocol event: a party with an inconsistent opening is treated as absent and its identity reported. Aborts are handled in the ideal model of Definition 17, where the adversary may, after learning its own output, send (abort, i^*) for a corrupted i^*, in which case all parties output $(\perp, \{i^*\})$. The gadget therefore reveals nothing about payments, or about others' valuations beyond what the allocation implies.

▶ **Example 22** (Walk-through of the SRA). Consider a single real bidder with valuation $v = 5$ drawn from the equal-revenue distribution $F(v) = 1 - 1/v$ on $[1, \infty)$, for which $\bar{\phi}(v) = v - (1 - F(v))/f(v) = v - v = 0$ for every v in the support. The optimal reserve is $r^* = 1$ and $\textsc{Rev}(F) = 1$. The auctioneer introduces two shills with bids $r_1 = 2$ and $r_2 = 8$, committing them to the same transformation $\bar{\phi}$ as the real bidder. Since $\bar{\phi} \equiv 0$, all three bids carry the same ironed virtual value, so the ranking of Definition 8 is resolved entirely on the second coordinate: the item goes to the highest raw bid.

Round 1. The WDG is invoked with all three bids and identifies shill $r_2 = 8$ as the provisional winner. The auctioneer must decide between two options. *Reveal* r_2: the shill wins, no real bidder is allocated the item, and the auctioneer collects revenue 0 – this is the cost of committing to a price floor of 8. *Hide* r_2: abort, forfeit the penalty κ, and restart without r_2.

Suppose the auctioneer hides r_2. Penalty κ is burned, and the auction restarts with $\{\text{bidder}, r_1\}$.

Round 2. The WDG runs on $\{\text{bidder}, r_1\}$. Since $v = 5 > r_1 = 2$, the real bidder wins. The bidder reveals $v = 5$ and deposits 5. Then r_1 is revealed as 2. The payment is the threshold of Definition 9 – the least bid at which the bidder still wins – and since all bids tie at $\bar{\phi} \equiv 0$ that threshold is settled on the raw bid, giving $r_1 = 2$ rather than the $\bar{\phi}^{-1}(\bar{\phi}(r_1)) = 1$ a numeric inverse would suggest. The auctioneer refunds $5 - 2 = 3$, keeping revenue 2.

On this particular realization the deviation nets $2 - \kappa$, which for $\kappa = 1$ is a gain of 1 over the honest revenue – so *ex post* the abort looks attractive. The decision to hide r_2, however, must be made *before* v is known, and at that point the auctioneer is conditioning only on the

Algorithm 3 Sequential revelation auction (SRA).

Parameters: Penalty κ per abort, MPC protocol Π_{WDG} that realizes $\mathcal{F}_{\mathcal{WDG}}$, Myerson mechanism (x, p).
Input: Valuations v_i from bidders $i \in A$.
Setup: Bidders $i \in A$ sample ρ_i and send $c_i = \text{Commit}(v_i, \rho_i)$ to Π_{WDG}.
Output: An ex-post allocation $x(v_A)$ and payment $p(v_A)$.

Commit Phase:
Each bidder $i \in A$ deposits a collateral κ in escrow
Each bidder $i \in A$ samples randomness ρ_i and broadcast $c_i = \text{Commit}(v_i, \rho_i)$
Reveal Phase:
Invoke Π_{WDG} with input $\{(v_i, \rho_i, c_i)\}_{i \in A}$
if Π_{WDG} *aborts* **then**
    Let k be the faulty bidder identified by Π_{WDG}
    Exclude k: $A \leftarrow A \setminus \{k\}$
    Restart from Reveal Phase with surviving bidders A
end
else
    Π_{WDG} publicly outputs the identity i^* of the *provisional winner*
    if $i^* = \perp$ **then**
        The item is unsold; the auctioneer refunds all collaterals and the protocol halts
    end
    The provisional winner reveals their commitment by broadcasting (v_{i^*}, ρ_{i^*}), then transfers v_{i^*} to the auctioneer
    if *the provisional winner refuses to participate* **then**
        Exclude the provisional winner: $A \leftarrow A \setminus \{i^*\}$
        Restart from Reveal Phase with remaining bidders A
    end
    else
        The provisional winner becomes the *final winner*. All bidders $j \in A \setminus \{i^*\}$ reveal their commitment by broadcasting (v_j, ρ_j)
        for *Each* $j \in A \setminus \{i^*\}$ **do**
            if j *fails to reveal* **then**
                Exclude j: $A \leftarrow A \setminus \{j\}$
            end
        end
        Auctioneer refunds $v_{i^*} - p_{i^*}(v_A)$ to the winner (where A is the set of currently active bidders)
        The auctioneer returns the collateral to each bidder that did not abort (i.e., $i \in A$), and **burns** the collateral for all others
    end
end

event that all real bids fall below 8. In expectation over v, the continuation revenue after the abort is at most $\textsc{Rev}(F) = 1$, so the net expected payoff from hiding is at most $1 - \kappa \leq 0$. Thus the auctioneer has no incentive to hide r_2 in the first place. Section 6 is precisely the general form of this calculation: it bounds the value of the information the auctioneer buys by aborting.

4.1 Safety

The requirement that the winner i^* deposits v_{i^*} *before* other bidders reveal their commitments is structural, not incidental. It resolves a potential conflict between safety and credibility. Specifically, if i^* could observe the other bids before committing funds, they could abort upon discovering an unfavorable price (e.g., $p_{i^*} \approx v_{i^*}$), forcing a restart. By acting first, i^* is effectively bound: an abort after this point results in forfeiture of v_{i^*}, which by individual rationality exceeds any possible payment $p_{i^*}(v)$. Thus, the auctioneer need not restart; the forfeited deposit v_{i^*} serves as sufficient revenue.

▶ **Theorem 23.** *The SRA satisfies safety against honest aborts.*

Proof. Recalling Definition 14, we must demonstrate that upon any abort by a non-auctioneer participant, the protocol either (i) restarts with the remaining participants $P' \subset P$, preserving the original mechanism's outcome on P', or (ii) terminates with revenue at least that of the mechanism on P'. We classify the possible abort conditions into three disjoint cases based on the protocol's progress:

1. **Abort during Computation (WDG):** If the MPC aborts, the Identifiable Abort property ensures the deviation is attributed to a specific party k. The protocol acts as if k never participated. The protocol restarts with $P \setminus \{k\}$, satisfying the definition.
2. **Abort during Revelation (Ex-Ante):** Suppose a bidder j aborts *before* the winner's deposit is secured.
   - If j is the winner i^*, they are excluded, and the protocol restarts. The revenue equals the revenue with $P \setminus \{i^*\}$.
   - If j is a loser, they are similarly excluded. The allocation rule implies the winner remains i^* (since $v_{i^*} > v_j$), and the payment remains determined by the highest remaining losing bid. Thus, the outcome is identical to the outcome with $P \setminus \{j\}$.
3. **Abort during Revelation (Ex-Post):** Suppose the winner i^* aborts *after* depositing v_{i^*}. The protocol guarantees revenue v_{i^*}. Since $v_{i^*} \geq p_{i^*}(v)$ (by individual rationality) this ensures i^* abort can only increase the auctioneer's revenue.

In all cases, the protocol recovers at least the revenue of the Myerson mechanism on the surviving set of bidders. ◀

4.2 Characterizing Deviations

We characterize the auctioneer's safe deviations in the SRA. Unlike real bidders, whose valuations are drawn from known distributions F_i, shill bidders are entirely controlled by the auctioneer. Crucially, the auctioneer must commit not only to each shill's bid b_j, but also to a monotone non-decreasing *ironed virtual value transformation* $\bar{\phi}_j : \mathbb{R}_+ \to \mathbb{R}$ that determines how shill j's bid is compared against real bidders in the winner determination gadget. This transformation is analogous to the ironed virtual value function for real bidders, but is chosen strategically rather than derived from a distribution.

The auctioneer's strategy thus consists of:

1. Selecting a set of shill bidders S, each $j \in S$ with bid b_j and committed virtual value transformation $\bar{\phi}_j(\cdot)$.
2. When shill $j \in S$ is identified as the winner, deciding whether to *reveal* b_j or *hide* (abort and restart without j).
3. When shill $j \in S$ is not identified as the winner, deciding whether to *reveal* b_j or *hide* (abort and restart without j).

For credibility, we require that the honest mechanism weakly dominates *all* such deviations – regardless of the choice of shill bids $\{b_j\}_{j\in S}$ and transformations $\{\bar{\phi}_j\}_{j\in S}$.

▶ **Definition 24** (Threshold virtual value). *For bidding profile $\boldsymbol{b}$ and bidder $i \in N \cup S$, the* threshold virtual value *is $\beta_i(\boldsymbol{b}) \coloneqq \max_{j\neq i} \bar{\phi}_j(b_j)$, the highest ironed virtual value among all other bidders.*

We analyze the auctioneer's decision by considering whether a shill is outbid (does not win) or wins the allocation.

▶ **Observation 25.** *Shill $j \in S$ is identified as the winner if and only if $\beta_j(\boldsymbol{b}) < \bar{\phi}_j(b_j)$ and $\bar{\phi}_j(b_j) \geq 0$, where $\boldsymbol{b}$ is the bidding profile of all non-aborting bidders.*

Proof. The winner determination gadget implements the Myerson optimal allocation of Definition 9, which allocates to the highest-ranked participant with non-negative ironed virtual value. Shill j is therefore the winner exactly when it outranks every other non-aborting bidder, real or shill. ◀

Refinements

This section, we show that it is without loss of generality to impose assumptions on the deviations of the auctioneer. First, we assume a shill bidder never aborts during the WDG computation phase since the protocol restarts without him. This is strategically equivalent to the shill bidder aborting immediately after the WDG output is revealed (identifying them as the winner or loser). Thus, we can focus our analysis on deviations that occur during the revelation phase.

The next Lemma argues that if a shill is outbid by a real bidder, then revealing dominates hiding.

▶ **Lemma 26** (Outbid shills). *For shill $j \in S$, if $\bar{\phi}_j(b_j) < \beta_j(\boldsymbol{b})$, then revealing b_j weakly dominates aborting.*

Proof. By Observation 25, if $\bar{\phi}_j(b_j) < \beta_j(\boldsymbol{b})$ then shill j is not the winner, so some other bidder $i^* \neq j$ wins and reveals first. At this point revealing b_j acts as a *price floor*: the winner pays the least bid beating $\beta_{i^*}(\boldsymbol{b})$, and including b_j among the competitors can only raise that maximum, hence can only raise the payment. Aborting instead removes b_j from the competitor set, weakly decreasing revenue while incurring penalty κ. ◀

▶ **Definition 27** (Refined safe deviation). *A* refined safe deviation *is a safe deviation in which, when choosing whether to reveal or abort $j \in S$, the deviation reveals b_j if any bid (shill or real) has been revealed so far, and otherwise chooses freely.*

▶ **Lemma 28** (Refinement). *For any safe deviation there exists a refined safe deviation with weakly greater expected revenue. Moreover, the shills of any safe deviation may be relabelled $r_1, \ldots, r_k$ so that $\bar{\phi}(r_1) < \bar{\phi}(r_2) < \cdots < \bar{\phi}(r_k)$.*

Proof. The ordering is immediate: the ranking is a strict total order (Definition 8), so it linearly orders any finite set of shills whatever transformations $\{\bar{\phi}_j\}_{j\in S}$ they were committed to.

For the reveal-after-reveal property, suppose some bid has been revealed. The winner is then already determined, so every remaining shill j satisfies $\bar{\phi}_j(b_j) < \beta_j(\boldsymbol{b})$, and Lemma 26 says revealing b_j weakly dominates aborting. ◀

4.3 DSIC Property of the SRA

We now establish that when all shills are revealed, the SRA reduces to a DSIC mechanism.

▶ **Theorem 29** (DSIC under full revelation). *For a refined safe deviation of the Sequential Revelation Auction, if the auctioneer reveals all shills (i.e., never aborts), then this deviation is Dominant Strategy Incentive Compatible (DSIC) for the real bidders.*

Proof. Under full reveal no participant is excluded, so the allocation is exactly that of Definition 9 applied to $A = N \cup S$: the item goes to the highest-ranked participant with non-negative ironed virtual value, and the winner pays the least bid that would still beat $\beta_i(\boldsymbol{b})$. Since the allocation is monotone and its payments are the induced threshold payments, so Theorem 3 gives that the deviation is DSIC for the real bidders. ◀

We close this section by verifying that honest behavior is also optimal for the real bidders in the *extensive form*, not merely in the reported values. Two steps of Algorithm 3 warrant this check: the provisional winner must escrow the full v_{i^*} before learning the price, and the losers must open commitments that serve only to raise the winner's payment.

▶ **Proposition 30** (Honest play is weakly dominant for real bidders). *Fix $\kappa > 0$ and a real bidder $i \in N$ participating in the SRA. Conditional on reaching any information set of Algorithm 3, following the protocol weakly dominates aborting.*

Proof. A bidder who aborts forfeits the collateral κ and is excluded from the continuation, so it suffices to show that continuing yields a non-negative payoff. There are two cases.

If i is a loser, then i receives no allocation and makes no payment whether or not they open their commitment: their payoff from continuing is 0, while aborting costs $\kappa > 0$. (Real bidders are not revenue-maximizers, so the fact that opening raises the winner's payment does not enter i's utility.)

If i is the provisional winner, continuing means escrowing v_i and later receiving the refund $v_i - p_i(\boldsymbol{v}_A)$, for a net payoff of $v_i - p_i(\boldsymbol{v}_A) \geq 0$ by individual rationality of the Myerson payment rule: $p_i(\boldsymbol{v}_A)$ is the least bid beating $\beta_i(\boldsymbol{v}_A)$, which is at most v_i whenever i wins. Aborting instead forfeits κ and yields no allocation, for a payoff of $-\kappa < 0$. ◀

5 Example: Equal Revenue Distribution

We now illustrate credibility for the canonical equal revenue distribution $F(v) = 1 - 1/v$ ($v \geq 1$). This distribution serves as a critical stress test for credibility due to two distinct properties:

1. **Heavy Tail:** The polynomial tail $\mathbb{P}[v > t] = 1/t$ implies that extremely high valuations occur with non-negligible probability. This allows shill-bidding strategies to chase huge payouts without the risk vanishing exponentially fast.
2. **Constant Revenue:** The "equal revenue" property ($r \cdot \mathbb{P}[v \geq r] = 1$) implies that every reserve price yields the identical expected revenue of 1. The distribution is "flat" with respect to optimization.

[11] demonstrated that for deferred revelation auctions (without MPC gadgets), *no finite penalty* suffices for this distribution – the heavy tail allows geometric shill strategies to extract unbounded profit. As a warmup, we show how a penalty of $\kappa \geq 1$ prevents unbounded profit for a natural class of shill strategies under the SRA. In Section 8, we prove this result is tight.

▶ **Example 31** (Warmup: Equal Revenue Distribution)**.** Consider a single real bidder with valuation drawn from the equal revenue distribution $F(v) = 1 - 1/v$ for $v \geq 1$. Suppose the auctioneer employs a safe deviation using k shill bids $r_1 < r_2 < \cdots < r_k$, where they hide $r_i, \ldots, r_k$ if and only if $v \in [r_{i-1}, r_i)$. With a penalty $\kappa \geq \textsc{Rev}(F) = 1$, we show this deviation yields expected revenue at most 1.

Without loss of generality, $r_j \geq 1$ for all j (shills below the support never win). Set $r_0 = 1$ (the lower bound of the support) and $r_{k+1} = \infty$. For the equal revenue distribution, the mechanism allocates to the highest bidder (any reserve $r \geq 1$ is optimal).

When $v \in [r_{i-1}, r_i)$ the auctioneer hides $r_i, \ldots, r_k$ (each incurring penalty κ), while shills $r_1, \ldots, r_{i-1}$ are outbid and revealed as price floors (by lemma 26). The bidder wins and pays r_{i-1}, giving ex-post revenue:

$$R(v) = r_{i-1} - (k - i + 1) \cdot \kappa.$$

We decompose the expected revenue into gross payments minus penalties. Using $\mathbb{P}[v \in [r_{i-1}, r_i)] = 1/r_{i-1} - 1/r_i$, the gross payment term telescopes:

$$\text{Gross} = \sum_{i=1}^{k+1} \left(\frac{1}{r_{i-1}} - \frac{1}{r_i} \right) r_{i-1} = \sum_{i=1}^{k+1} \left(1 - \frac{r_{i-1}}{r_i} \right).$$

For the penalties, each shill r_j is penalized whenever it *wins*, i.e., when $v < r_j$, which occurs with probability $1 - 1/r_j$. Thus the total expected penalty is $\kappa \sum_{j=1}^{k} (1 - 1/r_j)$. Setting $\kappa = 1$ and separating the last term of the gross sum ($1 - r_k/r_{k+1} = 1$):

$$\mathbb{E}[R(v)] = 1 + \sum_{j=1}^{k} \left[\underbrace{\left(1 - \frac{r_{j-1}}{r_j} \right)}_{\text{gross from interval } j} - \underbrace{\left(1 - \frac{1}{r_j} \right)}_{\text{penalty for shill } r_j} \right] = 1 + \sum_{j=1}^{k} \frac{1 - r_{j-1}}{r_j}.$$

Since $r_{j-1} \geq r_0 = 1$ for all $j \geq 1$, each summand is non-positive, giving:

$$\mathbb{E}[R(v)] \leq 1 = \textsc{Rev}(F).$$

6 Bounding Information Leakage Under Abort

By Myerson's characterization, the expected revenue of any DSIC mechanism equals its expected virtual surplus. However, this classical equivalence holds only when the expectation is taken over the full, unconditioned prior distribution. In our credibility analysis, the auctioneer's decision to abort and restart the auction hinges on a conditional event – specifically, learning that the highest real bid is below a certain threshold. When we condition on such an upper-bound event – the event E_r of Lemma 32 below, on which no real bidder outranks the shill r – the boundary terms in Myerson's integration by parts no longer vanish, and expected revenue can exceed expected virtual surplus.

The *Information Leakage Lemma* quantifies precisely how much the conditional expected revenue can exceed the conditional expected virtual surplus. Crucially, we prove that this excess "leakage" is bounded by $\sum_{i=1}^{n} \textsc{Rev}(F_i)$ (the sum of the optimal monopoly revenues for each bidder). This bounds the maximum option value the auctioneer can extract from restarting, dictating the minimum penalty κ required to deter deviations.

Intuition via a single bidder

Consider a single bidder with valuation $v \sim F$, reserve price r^*, allocation rule $x(v) = \mathbb{1}_{v \geq r^*}$, and payment rule $p(v) = r^* \mathbb{1}_{v \geq r^*}$. Under DSIC, the utility of a bidder of type v is given by the envelope formula $u(v) = \int_0^v x(t)\, dt$, which represents the bidder's accumulated information rent. Myerson's identity shows that the unconditional expected revenue is $\textsc{Rev}(F) = r^*(1 - F(r^*)) = \mathbb{E}\left[\phi(v)^+\right]$, where $\phi(v)$ is the virtual value. Now, condition on the upper-bound event $E = \{v \leq c\}$ for some threshold $c > r^*$. Working with unnormalized expectations throughout, the truncated expected payment is $\mathbb{E}\left[p(v)\mathbb{1}_{v \leq c}\right] = \mathbb{E}\left[\left(vx(v) - \int_0^v x(t)\, dt\right) \mathbb{1}_{v \leq c}\right]$. When we perform integration by parts on the truncated information rent term $\int_0^c f(v) \int_0^v x(t)\, dt\, dv$, the boundary term at $v = c$ does not vanish, yielding:

$$\int_0^c f(v) \int_0^v x(t)\, dt\, dv = F(c) \int_0^c x(t)\, dt - \int_0^c F(v)x(v)\, dv.$$

Rearranging terms, the truncated expected revenue satisfies:

$$\mathbb{E}\left[p(v)\mathbb{1}_{v \leq c}\right] = \mathbb{E}\left[\phi(v)x(v)\mathbb{1}_{v \leq c}\right] + (1 - F(c)) \int_0^c x(t)\, dt.$$

The second term on the right-hand side is the non-vanishing boundary term representing the information leakage: the utility $u(c) = \int_0^c x(t)\, dt = c - r^*$ of the boundary type c weighted by the probability of exceeding c in the unconditioned world. Crucially, this leakage is bounded by

$$(1 - F(c))\,(c - r^*) \;\leq\; c\,(1 - F(c)) \;\leq\; \textsc{Rev}(F),$$

where the last inequality holds because posting the price c cannot beat the optimal monopoly price. The Information Leakage Lemma generalizes this analysis to n bidders.

We state the lemma for the class of mechanisms that arise as continuations of the SRA after some shills have been revealed: the Myerson optimal auction augmented with a price floor. Formally, for a shill bid $r \geq 0$ let $M^{(r)}$ denote the DSIC mechanism with allocation mrule

$$x_i(\boldsymbol{v}) = \mathbb{1}_{\bar{\phi}_j(v_j) < \bar{\phi}_i(v_i)\ \forall j \in N \setminus \{i\}} \cdot \mathbb{1}_{\bar{\phi}(r) < \bar{\phi}_i(v_i)} \cdot \mathbb{1}_{\bar{\phi}_i(v_i) \geq 0}$$

and the induced threshold payments, all comparisons being taken in the ranking of Definition 8. Placing r below every participant recovers the Myerson optimal auction, and $M^{(r_i)}$ is the continuation in which shills $r_1 < \cdots < r_i$ have been revealed.

▶ **Lemma 32** (Information Leakage Lemma). *Let there be n bidders with independent valuations drawn from distributions $F_1, \ldots, F_n$ with vanishing revenue tails, $\lim_{v \to \infty} v\,(1 - F_i(v)) = 0$. For a shill bid r with $\bar{\phi}(r) \geq 0$, define the event*

$$E_r = \{\boldsymbol{v} \mid \forall i,\ \bar{\phi}_i(v_i) \leq \bar{\phi}(r)\}$$

on which no real bidder outranks r. Let $R(\boldsymbol{v}) = \sum_{i=1}^n p_i(\boldsymbol{v})$ and $VS(\boldsymbol{v}) = \sum_{i=1}^n x_i(\boldsymbol{v})\bar{\phi}_i(v_i)$ denote the ex-post revenue and ex-post ironed virtual surplus, respectively, of $M^{(r')}$ for any floor $r' \leq r$. Then,

$$\mathbb{E}\left[R(\boldsymbol{v})\mathbb{1}_{E_r}\right] \leq \mathbb{E}\left[VS(\boldsymbol{v})\mathbb{1}_{E_r}\right] + M \cdot \mathbb{P}\left[E_r\right]$$

where $M = \sum_{i=1}^n \textsc{Rev}(F_i)$ and $\textsc{Rev}(F_i)$ is the expected revenue of the optimal monopoly posted price for distribution F_i.

Observe the event E_r is a product of half-lines, $E_r = \prod_{i \in N}\{v_i < c_i\}$, where c_i is the least bid at which i would overtake r. This product form is the only property of E_r the proof uses. Note that c_i need not be $\bar{\phi}_i^{-1}(\bar{\phi}(r))$: when $\bar{\phi}_i$ is flat at the level $\bar{\phi}(r)$ the tie is settled on the raw bid, and $c_i = r$. That case is exactly what the ranking is for, and the proof handles it directly.

▶ **Lemma 33** (Ironing Equivalence; [17]). *Let F_i be a distribution with virtual value function ϕ_i and ironed virtual value function $\bar{\phi}_i$. Let $x_i(\boldsymbol{v})$ be the allocation rule of the Myerson-optimal auction. Let c be any threshold that does not lie in the strict interior of an ironed interval of F_i. Then for any fixed $\boldsymbol{v}_{-i}$,*

$$\int_0^c x_i(v_i, \boldsymbol{v}_{-i})\, \phi_i(v_i)\, f_i(v_i)\, dv_i = \int_0^c x_i(v_i, \boldsymbol{v}_{-i})\, \bar{\phi}_i(v_i)\, f_i(v_i)\, dv_i.$$

This identity holds because the Myerson-optimal allocation rule x_i is constant on any ironed interval $I = [\underline{v}, \overline{v}]$ of F_i (i.e., any interval on which $\bar{\phi}_i$ is flat). Consequently, on each complete ironed interval, the allocation rule $x_i(v_i, \boldsymbol{v}_{-i})$ can be factored out of the integral. Since $\bar{\phi}_i$ is defined as the conditional expectation of ϕ_i with respect to the distribution F_i on that interval, we have $\int_I \phi_i(v_i) f_i(v_i)\, dv_i = \bar{\phi}_i(v) \int_I f_i(v_i)\, dv_i$ for any $v \in I$. Because the threshold c never bisects an ironed interval, the domain $[0, c]$ can be perfectly partitioned into complete ironed intervals and regions where $\phi_i = \bar{\phi}_i$. Thus, the equivalence holds over the truncated domain. For the foundational proof, see [17, Section 4].

The proof of Lemma 32 is deferred to the full version.

7 Sequential Revelation Auction is Credible

We now present our main result. The warmup (Section 5) demonstrated credibility for a restricted class of strategies against the equal-revenue distribution. The Information Leakage Lemma (Section 6) established the key quantitative tool: the excess revenue an auctioneer can extract from conditional information is bounded by $\sum_{i=1}^n \textsc{Rev}(F_i)$. We now combine these ingredients to prove credibility in full generality.

▶ **Theorem 34** (Credibility of the sequential revelation auction). *Consider a single-item auction with n bidders with independent valuations drawn from product distribution $\boldsymbol{F}$. Then the Sequential Revelation Auction (Algorithm 3) with penalty $\kappa \geq \sum_{i=1}^n \textsc{Rev}(F_i)$ is credible.*

Proof roadmap

The proof reduces the auctioneer's complex, adaptive strategy space to a finite number of deterministic cases, then bounds each one. The reduction has three layers:

1. **Refinement** (Lemma 28): Without loss of revenue, all shills share the common virtual value function and outbid shills are always revealed.
2. **Kuhn's Theorem** (Lemma 38): Any adaptive (behavioral) strategy is equivalent to a mixture of pure strategies, each of which commits to a threshold: "abort all shills above r_i, reveal r_i and below."
3. **Case analysis** (Lemma 37): For each threshold r_i, the expected revenue decomposes into virtual surplus on the "good event" (a real bidder wins) plus conditional revenue on the "bad event" (all real bidders are below r_i). The Information Leakage Lemma bounds the latter, and the penalty $\kappa \geq \sum_j \textsc{Rev}(F_j)$ offsets it.

The central tension is a *commitment vs. free option* trade-off. Revealing a shill commits the auctioneer to a fixed price floor, whereas hiding it opens a "free option" to restart – but at cost κ. We show the option value is always bounded by $\sum_j \textsc{Rev}(F_j)$, so the penalty neutralizes it.

We start by bounding the revenue on good events for the auctioneer – i.e., when a real bidder is guaranteed to be the winner. We show that conditioned on a real bidder being the winner, the expected revenue is equal to the expected maximum virtual surplus restricted to the event that a real bidder is the winner.

▶ **Lemma 35.** *Consider any safe deviation of SRA. Let r be the highest-ranked shill bid, and write $L = \bar{\phi}(r)$ for its ironed virtual value. Define the event $E = \{\boldsymbol{v} \in \mathbb{R}^N : \exists i \in N, \bar{\phi}_i(v_i) > L\}$ as the set of profiles where at least one real bidder outranks every shill, the comparison being taken in the ranking of Definition 8. Then the expected revenue of the safe deviation on event E equals the expected maximum virtual surplus restricted to the event E:*

$$\mathbb{E}\left[R(\boldsymbol{v}) \cdot \mathbb{1}_E\right] = \mathbb{E}\left[\max_{i \in N} \bar{\phi}_i(v_i) \cdot \mathbb{1}_E\right].$$

where $R(\boldsymbol{v}) = \sum_{i=1}^n p_i(\boldsymbol{v})$ denotes the ex-post revenue of the deviation.

The proof of Lemma 35 is deferred to the full version.

Bounding Refined Deviations

By Lemma 28, it suffices to consider refined safe deviations, where the auctioneer submits shill bids $r_1 < \cdots < r_k$ – ordered by the ranking of Definition 8 – and processes them from highest to lowest. We define r_{k+1} to be a formal top element that every participant precedes.

Let $\mathcal{E}_j$ be the event that no real bidder outranks the j-th shill:

$$\mathcal{E}_j := \{\boldsymbol{v}_N \in \mathbb{R}_+^N : \bar{\phi}_i(v_i) \le \bar{\phi}(r_j) \text{ for all } i \in N\}.$$

Because the ranking is a total order, $\mathcal{E}_j$ *is* the event that shill r_j is the provisional winner, by Observation 25 – not merely a superset of it. This is where the ranking earns its keep: had ties been left to a rule outside the comparison, a real bidder tied with r_j but carrying a higher raw bid would lie in $\mathcal{E}_j$ while r_j nonetheless loses, and the penalty accounting in Lemma 37 would charge for an abort that never happens. It is the exact complement of the event E of Lemma 35, and it coincides with the event E_{r_j} of the Information Leakage Lemma (Lemma 32), which is what lets the two lemmas compose in Lemma 37. Since $r_{i+1} < \cdots < r_k$, the events are nested: $\mathcal{E}_{i+1} \subseteq \cdots \subseteq \mathcal{E}_k$.

Let I_ℓ denote the information set where the auctioneer decides whether to reveal or abort r_ℓ. Reaching I_ℓ implies that all higher shills were aborted, and that $\boldsymbol{v}$ satisfies event $\mathcal{E}_\ell$. For any mechanism M, let $R(\boldsymbol{v}; M) = \sum_{i=1}^n p_i^M(\boldsymbol{v})$ denote the ex-post revenue. We write $R(\boldsymbol{v})$ when M is clear from context.

▶ **Lemma 36** (Revenue Decomposition)**.** *From information set I_ℓ, the ex-ante expected revenue is:*

$$\mathbb{E}\left[R(\boldsymbol{v})\right] = \mathbb{E}\left[VS(\boldsymbol{v}) \cdot \mathbb{1}_{\mathcal{E}_\ell^c}\right] + \mathbb{E}\left[R(\boldsymbol{v}) \cdot \mathbb{1}_{\mathcal{E}_\ell}\right]$$

The proof of Lemma 36 is deferred to the full version.

Next, we formalize the deterministic strategies. Any deterministic strategy corresponds to a case C_i for some $i \in \{0, 1, \ldots, k\}$:

C_i: The auctioneer reveals shill r_i (if $i \ge 1$) and aborts all higher shills $r_{i+1}, \ldots, r_k$. The case C_0 corresponds to aborting all shills.

▶ **Lemma 37** (Bounding Deterministic Cases). *For any $i \in \{0, 1, \ldots, k\}$, the expected deviation revenue under C_i is at most $\textsc{Opt}(\boldsymbol{F})$, provided $\kappa \geq \sum_{j=1}^{n} \textsc{Rev}(F_j)$.*

The proof of Lemma 37 is deferred to the full version.

Finally, we restrict the auctioneer's adaptive strategies to pure strategies.

▶ **Lemma 38** (Reduction to Pure Strategies). *The maximum expected revenue achievable by any behavioral strategy β in the refined deviation space is bounded by the maximum expected revenue of the deterministic cases $\{C_0, \ldots, C_k\}$.*

The proof of Lemma 38 is deferred to the full version.

Due to space constraints, the proof of Theorem 34 is deferred to the full version.

▶ **Theorem 39.** *For any product distribution $\boldsymbol{F}$, the Sequential Revelation Auction (Algorithm 3) with penalty $\kappa \geq \sum_{i=1}^{n} \textsc{Rev}(F_i)$ is a credible auction protocol that implements the revenue-optimal auction. Furthermore, on the equilibrium path without shills, the protocol terminates in a constant number of rounds.*

The proof of Theorem 39 is deferred to the full version.

8 Lower Bound

The following result shows that Theorem 34 is tight: the Sequential Revelation Auction (Algorithm 3) admits profitable safe deviations when the penalty κ is smaller than the optimal revenue.

▶ **Theorem 40** (Profitable Safe Deviation of the SRA). *Consider the Sequential Revelation Auction with $N \geq 1$ bidders whose valuations are drawn i.i.d. from the equal revenue distribution. If the penalty per hidden shill bid satisfies $\kappa < N$, then there exists a safe deviation with expected revenue strictly greater than N. In particular, since $\sum_{i=1}^{N} \textsc{Rev}(F_i) = N = \textsc{Opt}(\boldsymbol{F})$ for this distribution, the sufficient penalty of Theorem 39 is exactly tight.*

We defer the full proof to the full version.

9 Conclusion

We introduced the Sequential Revelation Auction (SRA), a protocol that achieves credibility for any product distribution by combining a minimal MPC sub-computation – the Winner Determination Gadget – with sequential revelation and economic penalties. The gadget reveals only the winner identity, never payment information, which prevents the adversary from exploiting the "free option" inherent in monolithic MPC approaches. We proved that a penalty of $\kappa \geq \sum_{i=1}^{n} \textsc{Rev}(F_i)$ is sufficient for credibility across all product distributions, and that it is tight in the worst case: for the equal-revenue distribution, where the threshold coincides exactly with the optimal revenue, every smaller penalty admits a profitable deviation.

Sharper penalties and richer adversaries

Our lower bound shows $\kappa \geq \sum_{i=1}^{n} \textsc{Rev}(F_i)$ is necessary *for the SRA*; whether another design – richer gadgets, or a different revelation order – is credible at a smaller penalty is open, as is tightening the constant in $\textsc{Opt}(\boldsymbol{F}) \leq \kappa \leq n \cdot \textsc{Opt}(\boldsymbol{F})$. Our round bound holds on the equilibrium path only: the worst case grows linearly in the number of aborts, so relative to [10] the SRA buys a deterministic on-path bound and non-identical distributions

without improving the worst case. We also assume independent private values and a static adversary; correlation breaks both Myerson's benchmark and the leakage bound, and the repeated-auction setting, where the auctioneer's information advantage compounds across rounds, is outside the single-shot notion analyzed here.

Practical implementation

Constant-round MPC with identifiable abort from MK-FHE or iO is currently impractical, so the SRA is at present a feasibility result rather than a deployable protocol. The decomposition principle, however, is also a concrete efficiency argument: the gadget evaluates a single arg max over n committed values, a circuit of size $O(n\ell)$ for ℓ-bit bids, rather than a full Myerson circuit with ironing and threshold-payment computation. This suggests efficient instantiations may be within reach of modern MPC constructions such as garbled circuits or secret sharing over small fields, which is the most promising route to a practical credible auction.

References

1 Mohammad Akbarpour and Shengwu Li. Credible auctions: A trilemma. *Econometrica*, 88(2):425–467, 2020. doi:10.3982/ECTA15925.

2 Carsten Baum, Emmanuela Orsini, Peter Scholl, and Eduardo Soria-Vazquez. Efficient constant-round mpc with identifiable abort and public verifiability. In *Advances in Cryptology – CRYPTO 2020*, pages 549–579, Cham, 2020. Springer. doi:10.1007/978-3-030-56880-1_20.

3 Manuel Blum. Coin flipping by telephone. In *COMPCON Spring*, pages 133–137, San Francisco, CA, 1981. IEEE.

4 Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, et al. Secure multiparty computation goes live. In *Financial Cryptography and Data Security (FC)*, pages 325–343, Berlin, Heidelberg, 2009. Springer. doi:10.1007/978-3-642-03549-4_20.

5 Felix Brandt. How to obtain full privacy in auctions. *International Journal of Information Security*, 5(4):201–216, 2006. doi:10.1007/s10207-006-0001-y.

6 Tarun Chitra, Matheus V. X. Ferreira, and Kshitij Kulkarni. Credible, optimal auctions via public broadcast. In *6th Conference on Advances in Financial Technologies (AFT 2024)*, volume 316 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 19:1–19:16, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.AFT.2024.19.

7 Michele Ciampi, Divya Ravi, Luisa Siniscalchi, and Hendrik Waldner. Round-optimal multi-party computation with identifiable abort. In *Advances in Cryptology – EUROCRYPT 2022*, pages 335–364, Cham, 2022. Springer. doi:10.1007/978-3-031-06944-4_12.

8 Richard Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing (STOC)*, pages 364–369, New York, NY, USA, 1986. ACM. doi:10.1145/12130.12168.

9 Ran Cohen, Jack Doerner, Yashvanth Kondi, and Abhi Shelat. Secure multiparty computation with identifiable abort via vindicating release. In *Advances in Cryptology – CRYPTO 2024*, pages 36–73, Cham, 2024. Springer. doi:10.1007/978-3-031-68397-8_2.

10 Meryem Essaidi, Matheus V. X. Ferreira, and S. Matthew Weinberg. Credible, strategyproof, optimal, and bounded expected-round single-item auctions for all distributions. In *Proceedings of the 13th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 215 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 66:1–66:19, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITCS.2022.66.

11 Matheus V. X. Ferreira and S. Matthew Weinberg. Credible, truthful, and two-round (optimal) auctions via cryptographic commitments. In *Proceedings of the 21st ACM Conference on Economics and Computation (EC)*, pages 683–702, New York, NY, USA, 2020. ACM. doi:10.1145/3391403.3399495.

12 Aadityan Ganesh and Qianfan Zhang. Truthful, credible, and optimal auctions for matroids via blockchains and commitments. In *Proceedings of the 26th ACM Conference on Economics and Computation (EC)*, pages 923–943, New York, NY, USA, 2025. ACM. doi:10.1145/3736252.3742652.

13 Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure mpc from indistinguishability obfuscation. In *Theory of Cryptography Conference (TCC)*, pages 74–94, Berlin, Heidelberg, 2014. Springer. doi:10.1007/978-3-642-54242-8_4.

14 Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*, volume 2. Cambridge University Press, Cambridge, 2004. doi:10.1017/CBO9780511721656.

15 Yuval Ishai, Rafail Ostrovsky, and Vassilis Zikas. Secure multi-party computation with identifiable abort. In *Advances in Cryptology – CRYPTO 2014*, pages 369–386, Berlin, Heidelberg, 2014. Springer, Springer. doi:10.1007/978-3-662-44381-1_21.

16 Pratyay Mukherjee and Daniel Wichs. Two-round multiparty computation via multi-key fhe. In *Advances in Cryptology – EUROCRYPT 2016*, pages 735–763, Berlin, Heidelberg, 2016. Springer. doi:10.1007/978-3-662-49896-5_26.

17 Roger B. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58–73, 1981. doi:10.1287/moor.6.1.58.

18 Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce (EC)*, pages 129–139, Denver, CO, USA, 1999. ACM. doi:10.1145/336992.337028.

19 Andrew C Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 160–164, Chicago, IL, 1982. IEEE. doi:10.1109/SFCS.1982.38.

Competing Auctions in Intermediated Markets

Bruno Mazorra
Flashbots, Barcelona, Spain

Minghao Pan
California Institute of Technology, Pasadena, CA, USA
Flashbots, Pasadena, CA, USA

Christoph Schlegel
Flashbots, Zurich, Switzerland

Abstract

We analyze competing auctions in intermediated markets, where a seller selects among parallel mechanisms for the sale of a single good, most prominently the relay-and-protocol architecture of proposer-builder separation in Ethereum.

When the intermediary can enforce single-homing on its bidders, sealed-bid second-price intermediary auctions fully unravel into the sealed first-price principal auction; open bidding-format intermediaries unravel only partially, collapsing into first-price in equilibrium under symmetric latency and sorting fast bidders to the intermediary under asymmetric latency. Any last-look advantage is removed through the availability of a credible sealed bidding channel. These results extend to multi-plexing environments (no enforcement by the intermediary).

While the unraveling result indicates that the availability of a sealed first-price bidding channel pushes the overall market to the same auction structure, the very assumption of the credibility of such a channel is problematic, as the seller may have an incentive to leak information: a first-price auction is leakage-resistant in the presence of a single "fast" bidder but not against two or more. However, if the seller can credibly commit to not leak bids, it is optimal for them to do so.

A main motivation is the forthcoming Glamsterdam update of Ethereum: our analysis suggests that the availability of an in-protocol (first-price) bidding channel severely limits the design space for out-of-protocol auctions by relays and other intermediaries.

2012 ACM Subject Classification Theory of computation → Algorithmic game theory and mechanism design

Keywords and phrases Competing auctions, proposer-builder separation, maximal extractable value

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.19

Related Version *Full Version*: `https://arxiv.org/abs/2606.06633`

1 Introduction

In many markets, the same good can be bought through different channels, often through competing auction mechanisms. In classical online marketplaces (eBay etc.) competing auctions usually mean that there are competing sellers of the same type of good, and the sellers themselves may multi-home, offering their goods or services across different platforms at the same time (prominent examples include ride-sharing services). In other contexts, however, there may be a single seller of a single good, while competition happens between intermediaries. Competing auctions by competing sellers have been studied in a rich literature starting with [10]; competing auctions run by intermediaries for a single good have received much less attention.

This is not only a theoretical gap, but practically very relevant. The most familiar example from traditional finance is order routing: a broker with a single order to execute chooses among competing venues running different auction mechanisms (lit exchanges, dark pools, retail wholesalers, periodic auctions, RFQ systems) and is required under best-execution


© Bruno Mazorra, Minghao Pan, and Christoph Schlegel;
licensed under Creative Commons License CC-BY 4.0



rules to route the order to whichever venue offers the best outcome. Blockchain systems, which will be our main motivating examples, are even more abundant in intermediation services, and we observe competing auctions throughout the transaction inclusion pipeline. Ethereum's MEV ecosystem offers a particularly rich set of examples, from order flow auctions (MEV-protect, MEV-blocker, etc.) over routing and RFQ services (1inch, CowSwap etc.) to builder competition (Titan, BuilderNet etc.) for searcher bundles. In each case, the transaction or intent is the single good, and competing intermediaries run different auction mechanisms to capture it.

A forthcoming Ethereum protocol upgrade, enshrined Proposer-Builder Separation (ePBS) [6], introduces another instance of competing auction mechanisms. Currently, Proposer-Builder Separation, the market structure through which block builders bid for their blocks to be proposed, is implemented out-of-protocol with the help of intermediaries (so-called relays). Under ePBS, the protocol provides a native bid-and-commitment layer for builder bids, while off-protocol channels may still coexist with it: bids may be routed through public protocol gossip, private proposer-facing builder RPCs, or new relay-like intermediaries. Thus, we will possibly observe competition between direct in-protocol and intermediated bids.

In this paper, we study this competition between an in-protocol bidding channel and intermediated ones. Our analysis predicts that the introduction of an in-protocol sealed-bid first-price channel severely constrains the design space available to off-protocol intermediaries, and we characterize which relay formats can and cannot survive this competition.

We analyze the question of what an in-protocol bidding channel does to relay intermediation from several angles: first we study what kind of pricing rule the competing auction can maintain in the presence of the in-protocol outside option. A plausible alternative choice to a first-price auction (sealed or open in the form of an English auction) would be to run a second-price auction format. A second-price auction has the advantage that bidders have an incentive to bid their value truthfully in it.[1] In the presence of a competing first-price channel, however, bidders still bid truthfully *conditional on entering* the second-price auction (Lemma 2), but entering it at all becomes unattractive: in the single-homing case, the second-price auction unravels and bidders self-select into the first-price (in-protocol) auction (Theorem 1). In the multi-homing case, the good will almost surely be allocated through the first-price auction (Theorem 4).

Second, we study how information can be shared throughout the auction: open bidding as in the status quo is naturally undermined by the availability of a sealed-bid in-protocol channel. Bidders have an incentive to listen to an open-bid feed but to bid through a sealed channel if available. However, the seller (or an intermediary acting on the seller's behalf) can choose to share bids received during the auction period, orchestrating a (partially) open-bid auction. Such information sharing naturally unravels in the sense that bidders have an incentive to bid as late as possible to avoid having their bids shared, effectively turning the auction into a sealed-bid auction (Propositions 7 and 14). However, in the presence of latency differences by bidders, information sharing (of the slower bidders' bids with the faster bidders) may be feasible. We analyze this in a model where a subset of bidders is "fast" and the seller or intermediary may share bids with these bidders. We find that sharing information with a single fast bidder is not optimal for the seller (Proposition 8), whereas sharing information with two or more fast bidders (in case there are that many fast bidders) is

[1] Currently, some relays like the Ultrasound relay experiment with "second-price" features.

Table 1 Summary of results. Each row corresponds to one main proposition or theorem in the paper, organized by section.

Section	Setting	Result
§ 3.1	Single-homing, sealed second-price relay	The native first-price auction is the unique symmetric equilibrium outcome (Theorem 1).
§ 3.2	Multi-plexing, second-price relay	The item is allocated through the first-price auction with probability 1 (Theorem 4).
§ 3.3.1	No last-look advantage	All-first-price is an equilibrium; the all-English profile is unstable in large markets (Proposition 7).
§ 3.3.2	One fast bidder	The first-price bidding channel is leakage-resistant (Proposition 8).
§ 3.3.2	Multiple fast bidders, $\|S\| \geq 2$	The proposer leaks in equilibrium (Theorem 9).
§ 3.3.3	Last-look with credible proposer commitment	The proposer weakly prefers to commit to a credible sealed-bid first-price channel (Proposition 13).

a best response for the seller (Theorem 9) and fast bidders would benefit from bid disclosure (Proposition 11). However, if the seller can credibly commit ex-ante to a disclosure rule, the optimal commitment is to share no information at all (Proposition 13).

The interpretation of the "information disclosure" results in the Ethereum context is straightforward: in the absence of a credible first-price sealed-bidding channel, if there are at least two "fast" bidders, a proposer would have an incentive to leak the bids it has received (in or out-of protocol), effectively making the in-protocol bidding channel leaky. Practically, this could be orchestrated through a service where proposers forward the received bids to a relay that integrates them in their bid stream. However, there are potentially ways to establish a credible sealed bid first-price channel: proposers could use reputation (in the case of large staking pools like Lido) or hardware solutions such as Trusted Execution Environments (TEEs) to commit to not leak bids that they have received. Alternatively, they can delegate to an intermediary, i.e. a potential "sealed-bid first-price" relay. As commitment to non-disclosure is optimal if possible, such a solution can quite plausibly recover the no-disclosure equilibrium institutionally.

We proceed as follows: in the next two subsections, we provide background on the literature on competing auctions, and institutional background on Ethereum's blockbuilding stack. In Section 2 we discuss our modeling choices for the subsequent analysis. The discussion is more detailed than the usual scope of such a section, primarily because some modeling choices are very much informed by institutional details of the block building ecosystem in Ethereum which we want to discuss in this context. In Section 3 we derive our results, which are summarized in Table 1. In Section 4, we conclude with practical implications of our analysis and open questions.

1.1 Competing Auctions & Related Literature

The literature on *competing auctions* [10, 15, 3, 7] studies the constellation where multiple sellers each offer their own good through a chosen mechanism, with buyers single-homing across sellers. The standard result is that competition pushes mechanisms toward second-price with a reserve in the large-market limit and disciplines reserve prices. Our setting differs structurally: a single seller has a single good and faces multiple parallel mechanisms run by intermediaries, with the winning mechanism determined ex-post by a max-revenue

rule. The strategic forces are therefore quite different: bidders are not choosing between substitute goods but between channels for the same allocation, and competition operates on intermediaries rather than on sellers.

The literature on *platform competition* [4, 1, 16] introduces the distinction between single-homing and multi-homing that also plays a role in our analysis. The general finding – that buyer-side multi-homing intensifies competition between platforms, while single-homing produces winner-take-all dynamics – is consistent with the broad shape of our results, though our setting is narrower and our focus is on equilibrium auction outcomes rather than platform pricing.

Our information leakage results connect to the literature on *seller-controlled information disclosure in auctions*. The classical result here is the linkage principle of Milgrom and Weber [11], which establishes that under affiliated and interdependent values, the seller benefits from committing to reveal information that links the price more strongly to bidders' signals. This logic does not apply in our setting: at the builder level, values are private (although possibly drawn from a correlated distribution) and known with certainty, so additional information about other bidders' valuations has no effect on a bidder's own valuation. The relevant disclosure question becomes strategic rather than informational: the seller's choice is not about helping bidders refine their valuations, but about changing the strategic environment by giving some bidders the ability to react to others' bids. The closer reference is therefore the literature on information design in auctions under private values [2]. The commitment-vs-no-commitment distinction in our results has a natural interpretation in terms of whether disclosure is delegated to a trusted intermediary.

The closest non-blockchain analog of our setting is in *market microstructure*, where a substantial literature studies how a single trade is routed across competing exchanges, dark pools, and wholesalers under best-execution obligations [8, 12]. A particularly close institutional parallel to our latency-advantaged-bidder setting is the case of last-look provision in FX markets [13], where dealers selectively respond to client quote requests based on observed market information.

1.2 From MEV-Boost to ePBS

Ethereum's current block-building market is organized through MEV-Boost, an off-chain implementation of proposer-builder separation. Validators outsource execution-payload construction to specialized builders. Builders compete by submitting blocks and associated bids to relays, and the proposer signs the highest-paying blinded block header returned by MEV-Boost. The relay is therefore the central intermediary: it runs the auction, hides the block contents until the proposer has committed, and then releases the full payload. In the standard, non-optimistic flow, the relay also verifies block validity and proposer payment before forwarding the bid. The builder sets itself as the block's `feeRecipient` and includes a final transfer to the proposer's registered `feeRecipient` address; the relay checks that this transfer matches the advertised bid.

Relay designs have evolved to reduce latency on the critical path. In optimistic relay designs, builders post collateral, allowing the relay to accept bids before completing full validation. The relay still intermediates the auction and forwards the blinded header to the proposer, but payment and validity checks are performed ex-post. If the winning block is invalid or fails to pay the promised amount, the builder's collateral is used to compensate the proposer. Optimistic relaying therefore makes the MEV-Boost auction more latency-efficient, since builders can submit competitive bids later in the slot, but it does not remove the relay as the trusted off-chain intermediary.

Thus, MEV-Boost implements PBS without changing Ethereum consensus, but it does so by relying on off-chain infrastructure. Relays solve the fair-exchange problem between proposers and builders: builders need protection against payload theft or payload leakage before commitment, while proposers need assurance that the promised block is valid and pays the advertised amount. The resulting market is naturally modeled as an open, continuous, relay-mediated auction. Builders can update bids during the slot, relays intermediate the information flow, and latency determines how late bids can arrive and still be competitive.

Enshrined proposer-builder separation, or ePBS, changes this architecture by moving the core commitment-and-payment logic into the protocol. In the current EIP-7732 design [6], the beacon block no longer contains the full execution payload on the attestation-critical path. Instead, it contains a signed builder commitment, the `SignedExecutionPayloadBid`, which commits to a later execution payload and specifies the value to be paid to the proposer. Builders become protocol-recognized entities with beacon-chain balances, so the committed payment can be deducted from the builder's balance and later credited to the proposer. Timely reveal is checked by a Payload Timeliness Committee (PTC), while full execution validation is deferred to a later point in the slot. The current EIP-7732 design has some potential flaws if payload-delivery failure is not sanctioned and the PTC deadlines are positioned too far in the slot [9]. As there are plausible mitigation mechanisms for this problem, we assume throughout the analysis that these "free-option" considerations do not play a role for the questions we seek to answer in this paper.

These protocol changes also alter the market structure. Under MEV-Boost, the relay observes and intermediates the bidding process, and the auction is effectively open, continuous, and latency-sensitive. Under ePBS, the protocol provides a native bid-and-commitment layer, while off-protocol channels may still coexist with it. The post-ePBS market may therefore contain competing auction formats: a native in-protocol mechanism and external auction channels that can differ in observability, timing, trust assumptions, and payment guarantees. Our goal is to understand when these mechanisms coexist, when one unravels into the other, and how the change from trusted relay-mediated exchange to protocol-mediated exchange affects builder and proposer incentives.

Trustless vs Trusted Payments. In MEV-Boost, payments are mediated by trusted off-chain infrastructure: the relay verifies that the builder's block contains the promised payment and releases the payload only after the proposer has signed the blinded block. In ePBS, by contrast, payment can be enforced in protocol. A builder's signed bid commits to a payment amount, which can be deducted from the builder's beacon-chain balance and later credited to the proposer.

The main cost of trustless payments is that builders must pre-fund their bids. This can increase entry barriers, since a builder needs sufficient in-protocol balance to participate. For ordinary blocks, this constraint may be mild: most bids are small, and if deposit and withdrawal frictions are low, the required working capital need not be large. For high-value blocks, however, the balance constraint can become binding. A builder with high value but insufficient balance may be unable to express its full willingness to pay through the trustless mechanism. In such cases, bids may still be routed through trusted relationships, relays, or other intermediaries that can provide credit, guarantees, or faster settlement. Subsequently, we will not model this particular friction explicitly, but the reader should keep in mind that inter-mediated bids may still have a role and certain advantages over in-protocol bidding, even if trustless bidding would be optimal based on "pure auction considerations" that are the main focus of our analysis.

2 Modeling Consideration

2.1 Private Values

We will work throughout the paper in a private value setting. Some justification is needed here, as this often seems to be a point of possible confusion. Most block value is generated from financial transactions, which inherently have a common-value component (with at least some uncertainty and heterogeneous signal on this value). However, this does not mean that block builders have a common value for proposing the next block. An uncertain common value is effectively realized up-stream at the searcher level. Block builders receive bundles and compose them into a block whose value for the builder is deterministic and known with certainty. This would be different if builders did searching themselves, a market dynamic called "searcher-builder integration" that has been discussed extensively in the past [14]. However, this dynamic has never materialized in reality, and at the current point in time the dominant Ethereum builders (Titan, BuilderNet, Quasar) are neutral and do not search. In particular, "price discovery" in the PBS auction, sometimes cited as an advantage of the current open auction format, is – if at all – relevant for searchers to update their bids to the builders. Thus, our analysis focuses on the practically relevant case of private values. As different builders tend to receive similar bundles (or even the same bundles from the same searcher), these private values are naturally positively correlated. However, we refrain from modeling the correlation structure explicitly, for the sake of analytical simplicity. We note, however, that positive correlation generally pushes in the same direction as the logic of most of our results, so that they likely generalize to models of positively correlated private valuations.[2] Thus, throughout the paper we assume that valuations are drawn i.i.d. from a distribution F that is absolutely continuous and strictly increasing on a compact interval $[\underline{v}, \overline{v}]$. We denote its density by f.

2.2 Single vs Multi-plexing of Bids

The in-protocol auction in ePBS is permissionless. The protocol can neither restrict who enters the auction nor restrict whether they submit multiple bids through different channels (in-protocol and directly to the proposer or through a relay) or just a single bid. Thus, the ePBS auction is by default "multi-plexing" friendly. The case of relays is more subtle. Relays already today restrict certain functionalities (optimistic relaying) to registered and staked participants while being generally open to all participants for their standard service. In a future with ePBS, it is still to be seen whether relays become "more permissioned" than they are today. One potential reason they could develop in this direction is that they want to offer additional services to remain attractive even with an in-protocol mechanism in place; for example, block-merging by the relay is easier to coordinate with a permissioned set of builders. More importantly for our context, a relay might want to sustain an auction format that is otherwise not sustainable in the multi-plexing case. The current open-bidding PBS

[2] A trivial form of introducing correlation for which our result would hold without modification of the proofs is through a block-level public state: Fix a block and condition on its realized public state; let F denote the value distribution induced by that state. Assume there is a family $\mathcal{F}$ of distributions supported on the same compact interval. For the realized distribution $F \in \mathcal{F}$, bidders' values satisfy

$$v_i \mid F \sim_{\text{i.i.d.}} F, \qquad i = 1, \ldots, n.$$

Thus values can be correlated before conditioning: the public state can shift the distribution faced by all bidders in the block. Economically, F captures the public market conditions of the current block. Whenever a proposition refers to F, it can be read as conditional on an arbitrary realized $F \in \mathcal{F}$.

auction, for example, is hard to maintain in an ePBS world: if a relay offers an open-bidding mechanism, then bidders have a natural incentive to listen to the bidding feed of the relay but to place their own bid in a sealed way directly to the proposer, with the relay auction unraveling as a consequence. If a relay wants to fight this logic, it might restrict bidding to a permissioned set of builders and impose that, in case a builder bids with them in a particular auction round, they need to commit to not bidding through an external channel in the same round. This assumes, of course, that deviations are detectable with sufficient probability by the relay. If a relay manages to implement single-homing within its service, it makes the entire market single-homing: bidders need to select into one of the two mechanisms before they bid.

We find both scenarios – single-homing enforced at the relay level and multi-homing – plausible at first view. Thus, we think it is valuable to analyze both cases.

2.3 Latency and Last-Look Advantage

The performance and equilibrium analysis of auctions in time-sensitive environments may be influenced by bidder latency considerations. The current MEV-Boost auction, for example, is naturally modeled, at an idealized level, as an English or candle-style auction. Relays receive continuously updated bids during the slot, builders can react to the current competitive level, and the process ends only when the proposer must commit to a block. But in the current MEV-Boost world, bid streams and timing frictions make latency economically meaningful: a builder who observes a competing bid and can still respond before the deadline while others cannot obtain a form of *last look* [14] advantage.

For our analysis of open bidding and bid disclosure we, therefore, consider two types of environments: We first consider the clean benchmark in which no bidder has a last-look advantage. In that benchmark, the relay auction may reveal information internally among relay participants, but bidders can submit a last bid that is not observed by competitors. Moreover, the relay winner does not observe the native first-price bids before choosing the proposer-facing bid and vice versa.

The no-last-look benchmark is an idealization. We then study the more interesting case of last look explicitly. We treat technological latency as exogenous, but the exploitation of latency as endogenous: low-latency builders can use their advantage only if some information is made available before the deadline. That information may be public, relay-mediated, or selectively disclosed by the seller.

This leads to two versions of the last-look problem. In the first, the seller cannot credibly commit to a disclosure strategy. After receiving early bids, the seller may decide history by history which private bids to leak and to whom (of the latency advantaged bidders).[3] In the second, the seller can commit ex-ante to a specific disclosure rule. Practically, this disclosure can be implemented through a trusted intermediary, such as a relay in the PBS case. The distinction matters because a sealed-bid procedure can be ex-ante attractive, while after the early bids have arrived the seller may have an incentive to leak them to latency-advantaged builders.

[3] For the PBS application, practically, the proposer will still decide ex-ante, before the auction, how to configure their node (or more likely which third-party side-car software to install). The point is that while the proposer chooses their strategy ex-ante in this application, they cannot credibly commit to a disclosure strategy that is not an equilibrium strategy.

3 Results

Proofs omitted for space are provided in the full version.

3.1 Single-Homing: First-Price vs Second-Price Auction

We first analyze the single-homing case. There are $n \geq 2$ bidders who bid for the right to decide on the content of the next block. The market structure is as follows: there are two competing auctions. The first auction is a first-price sealed-bid auction (the native ePBS auction) and the second a competing relay auction. The bidders can submit a bid to one of the two auctions or neither auction. The two auctions are run on the submitted bid. Whichever auction yields higher revenue allocates the item and charges the winner.

Bidders choose ad interim, i.e. after they know their valuation but before they send their first bid, through which channel they bid in the auction. The natural interpretation is that of an intermediary (relay) that can detect deviations (with high probability) where bidders bid through both channels and punish this "multi-plexing" behavior.

We first study whether a competing auction can enforce a second-price pricing rule. We restrict attention to symmetric equilibria.[4]

▶ **Theorem 1** (Single-homing unraveling)**.** *If bidders must choose to participate in exactly one of the first-price auction and the sealed second-price auction, then, up to F-null sets, there is a unique symmetric Bayes–Nash equilibrium. In this equilibrium, every type chooses the first-price auction and bids according to the standard symmetric first-price equilibrium.*

We separate the proof into several lemmas.

▶ **Lemma 2.** *In equilibrium, if an agent bids in the second-price auction, then she bids truthfully almost surely.*

Proof. By symmetry, we may assume that bidders who participate in the second-price auction bid truthfully almost surely. The reason is simple. Truthful bidding weakly dominates every other bid in the second-price auction: lowering the bid below one's value can only remove wins at profitable prices, while raising the bid above one's value can only add wins at loss-making prices. Moreover, in a symmetric equilibrium any positive-measure set of non-truthful second-price bids would make this domination strict on a positive-probability event. If a positive measure of types overbid, then by symmetry there is positive probability that all second-price participants have values below some cutoff but bids above it; the winner then pays more than her value, while truthful bidding would avoid this loss. If a positive measure of types underbid, then with positive probability all second-price participants have values above some cutoff but bids below it; at least one bidder then loses even though a truthful bid would win at a price below her value. Thus any non-truthful second-price bid used on a positive-measure set would admit a strictly profitable deviation to the truthful bid, contradicting equilibrium. Hence, conditional on participating in the second-price auction, bidders bid truthfully almost surely. ◀

[4] A natural question is whether the unraveling results in this and the following section follow more directly from revenue equivalence. Revenue equivalence determines the expected payment in any efficient symmetric mechanism with zero surplus at the lowest type, but it presupposes a fixed allocation rule. In our setting, the allocation rule is itself an equilibrium object of interest. In principle, the good can be allocated inefficiently – for example, if a high-value bidder selects themselves into the second-price auction but is outbid by a lower-value bidder in the first-price auction. We therefore cannot invoke revenue equivalence to short-cut the analysis.

▶ **Lemma 3.** *If two bidders with types $v, w \in (\underline{v}, \overline{v}]$ participate in the first-price auction with positive probability, then $v > w$ implies $b(v) > b(w)$.*

Proof of the theorem. For notational simplicity, normalize the support to $[0, 1]$. Let μ be the distribution associated with F. Suppose by contradiction that there exists a symmetric equilibrium with positive participation in the second-price auction. Let $E \subseteq [0, 1]$ denote the set of types who participate in the second-price auction.

We first show that $\alpha := \operatorname{ess\,inf} E > 0$. A type with valuation v, by Lemmas 2 and 3, obtains from the second-price auction payoff

$$U_2(v) := \int_{[0,v]\cap E} (v - t) \sum_{k=1}^{n-1} \binom{n-1}{k} k \cdot \mu(b^{-1}([0, t]) \cap E^c)^{n-1-k} \mu([0, t] \cap E)^{k-1} \, d\mu(t).$$

Here $\binom{n-1}{k}$ samples k other participants of the second-price auction, and the following factor k selects one participant to have value t.

Suppose that the function

$$t \mapsto (v - t) \sum_{k=1}^{n-1} \binom{n-1}{k} k \cdot \mu(b^{-1}([0, t]) \cap E^c)^{n-1-k} \mu([0, t] \cap E)^{k-1}$$

is maximized at some $t = t^*$. Then

$$U_2(v) \le \mu([0, v] \cap E)(v - t^*) \sum_{k=1}^{n-1} \binom{n-1}{k} k \mu([0, t^*] \cap E)^{k-1} \mu(b^{-1}([0, t^*]) \cap E^c)^{n-1-k}. \quad (1)$$

If the type chooses to bid t^* in the first-price auction instead, her utility payoff is at least

$$\begin{aligned}&(v - t^*)\mu(b^{-1}([0, t^*]) \cap E^c)^{n-1} \\ &\quad + (v - t^*) \sum_{k=1}^{n-1} \binom{n-1}{k} \mu(E) \mu([0, t^*] \cap E)^{k-1} \mu(b^{-1}([0, t^*]) \cap E^c)^{n-1-k}.\end{aligned} \quad (2)$$

Here $\binom{n-1}{k}\mu(E)\mu([0, t^*] \cap E)^{k-1}$ comes from first sampling k participants in the second-price auction, letting one participant bid arbitrarily, and having the remaining $k - 1$ participants bid less than t^*.

By comparing (2) with (1) term by term, as long as

$$(n - 1)\mu([0, v] \cap E) < \mu(E),$$

we have (2) $>$ (1). This is true when v is small enough. In other words, when v is small enough, v prefers the first-price auction to the second-price auction, and so $\alpha > 0$.

Let $\varepsilon > 0$ be small such that $\alpha + \varepsilon \in E$. When participating in the second-price auction, bidder $\alpha + \varepsilon$ has a chance of winning bounded above by

$$(n - 1)\mu([\alpha, \alpha + \varepsilon]).$$

So her utility payoff is at most

$$\varepsilon(n - 1)\mu([\alpha, \alpha + \varepsilon]).$$

However, if she bids α in the first-price auction, she wins whenever all other participants have value smaller than α, so her utility payoff is at least

$$\varepsilon\mu([0, \alpha])^{n-1}.$$

Thus, taking ε sufficiently small,

$$\mu([0,\alpha])^{n-1} > (n-1)\mu([\alpha,\alpha+\varepsilon]),$$

and so, bidder $\alpha+\varepsilon$ prefers the first-price auction, a contradiction. ◀

Theorem 1 implies that, under single-homing, the sealed second-price relay does not attract participation: in the unique symmetric equilibrium, all bidders bid in the first-price auction, and the item is allocated through that channel. In the Ethereum interpretation, the block is therefore allocated through the native direct-communication channel rather than through the relay. This conclusion, however, relies on the relay's ability to enforce single-homing. In a permissionless environment, a relay may restrict the identities that participate in its auction, but it cannot generally verify that an identity corresponds to a unique economic bidder. A bidder who wants to bid in both channels can use Sybil identities, submitting one bid through the relay and another through the native auction while appearing as distinct participants. Thus, single-homing has bite only to the extent that such deviations can be detected. When Sybil participation is feasible, bidders can circumvent the restriction by splitting bids across identities, so the effective environment is one of multi-homing. We analyze that case next.

3.2 Multi-Homing: First-Price vs Second-Price Auction

In this section, we prove that for the multi-plexing case with competing first-price and second-price auctions, there is an essentially unique symmetric pure-strategy Nash equilibrium in which the second-price auction unravels. The precise statement is the following.

▶ **Theorem 4** (Essential uniqueness). *Let b^{FP} be the unique symmetric pure-strategy equilibrium bidding function of the first-price auction. Every symmetric pure-strategy Nash equilibrium $v \mapsto (b(v), w(v))$ such that b is a.e. strictly increasing whenever the agent bids in the first-price auction is of the form $v \mapsto (b^{\mathrm{FP}}(v), w(v))$ with $w(v) \le b^{\mathrm{FP}}(v)$.*

Under any such equilibrium, the item is always allocated through the first-price auction. Hence the second coordinate w is payoff-irrelevant and may, without loss of generality, be taken to be identically zero.

Throughout this subsection, fix a symmetric pure-strategy equilibrium $v \mapsto (b(v), w(v))$ of the two-auction game, and define

$$P := \{v : w(v) > b(v)\}.$$

We identify strategies that differ only on an F-null set. We also use the standard observation that first-price overbidding is weakly dominated, so we may take $b(v) < v$ for every $v > \underline{v}$.

▶ **Lemma 5.** *Fix a type v and a first-price bid $x \le v$. Then, for every second-price bid $s > x$ and $s \neq v$, the action (x, v) weakly dominates (x, s) and $w(v) = v$ for almost every $v \in P$.*

▶ **Lemma 6.** *For almost all types v, the agent participates in the first-price auction.*

Proof. Observe that not participating in the first-price auction is payoff equivalent to bidding $\underline{v}$, and so wlog, we consider

$$P_0 := \{v \in P : b(v) = \underline{v}\}$$

as the set of agents that do not participate in the first-price auction. We first show that $\Pr[V \in P_0] = 0$. By Lemma 5, we may take $w(v) = v$ for almost every $v \in P$.

Suppose, towards a contradiction, that $\Pr[V \in P_0] > 0$, and let $\alpha := \operatorname{ess\,inf} P$. Define

$$\varphi(x) := \Pr[b(V) \le x,\ w(V) \le x].$$

Thus $\varphi(x)^{n-1}$ is the probability that all opponents submit both bids below x.

Fix $v \in P$. Since $w(v) = v$, the payoff from the equilibrium action is bounded above by

$$U(v) \le (v - b(v))\varphi(b(v))^{n-1} + \int_{[b(v),v]\cap P} (v-t)(n-1)\varphi(t)^{n-2}\, dF(t).$$

Let $t^* \in [b(v), v]$ maximize $(v-t)\varphi(t)^{n-2}$. Then

$$U(v) \le (v - t^*)\varphi(t^*)^{n-1} + (v-t^*)(n-1)\varphi(t^*)^{n-2}\Pr[V \in P \cap [b(v), v]].$$

Now let type v deviate by bidding t^* in the first-price auction and by making her second-price bid irrelevant. This deviation gives payoff at least

$$(v - t^*)\varphi(t^*)^{n-1} + (v-t^*)(n-1)\varphi(t^*)^{n-2}\Pr[V \in P_0 \cap [t^*, \infty)].$$

Since $t^* \le v$, we have

$$\Pr[V \in P_0 \cap [t^*, \infty)] \ge \Pr[V \in P_0 \cap [v, \infty)].$$

For $v \in P$ sufficiently close to α, the term $\Pr[V \in P \cap [b(v), v]]$ is arbitrarily small, whereas $\Pr[V \in P_0 \cap [v, \infty)]$ is bounded away from zero. Hence the deviation is strictly profitable, a contradiction. Therefore

$$\Pr[V \in P_0] = 0.$$

It remains to rule out types with $b(v) = \underline{v}$ and $w(v) \le b(v)$. Such types obtain zero payoff. But any type $v > \underline{v}$ can bid $(x, 0)$ for some $x \in (\underline{v}, v)$ and obtain strictly positive payoff on the event that all opponents' values are below x. Hence this cannot be a best response. Therefore $b(v) > \underline{v}$ for almost every v. ◀

Proof of Theorem 4. By the previous lemma and by assumption b is globally strictly increasing. We will show that $P = \emptyset$. Suppose, for contradiction, that there exists $v \in P$. By Lemma 5, $(b(v), v)$ is a best-reply. Now, consider the tuple-action (x, v) for an agent i with valuation v. His interim payoff is

$$\Pi(v, x) = (v - x)\Pr[B \le x, W \le x] + \mathbb{E}[(v - W)\mathbf{1}\{W > \max\{B, x\},\ W \le v\}],$$

where

$$W = \max_{j \ne i}\{w(v_j)\} \qquad \text{and} \qquad B = \max_{j \ne i}\{b(v_j)\}.$$

Now, we assume that Π is differentiable at $(v, b(v))$[5]. The derivative of Π with respect to x is

$$\frac{\partial \Pi(v,x)}{\partial x} = -\Pr[B \le x, W \le x] + (v - x) f_B(x)\Pr[W \le x \mid B = x].$$

[5] This assumption can be dropped by taking the left and right limits of Π and computing them directly. For simplicity, we do not include this argument.

Conditional on $B = b(v)$, since b is strictly increasing, we must have that there exists another $j \neq i$ such that $v_j = v$, so $W > b(v)$ and $\Pr[W \leq b(v) \mid B = b(v)] = 0$. Therefore,

$$\frac{\partial \Pi(v, b(v))}{\partial x} = -\Pr\left[B \leq b(v), W \leq b(v)\right].$$

However, $\Pr\left[B \leq b(v), W \leq b(v)\right] \geq \Pr[\max_{j \neq i}\{v_j\} \leq b(v)] = F(b(v))^{n-1}$ and since b is strictly increasing, $F(b(v))^{n-1} > 0$. Therefore, $\frac{\partial \Pi(v, b(v))}{\partial x} < 0$. This contradicts the fact that bidding $b(v)$ in the FP is a best-reply by first-order conditions, so there is no $v \in P$. ◀

When agents are allowed to use Sybil identities, the equilibrium analysis becomes more nuanced. The first-price only equilibrium is still an equilibrium even with Sybil-strategies. However, uniqueness can generally fail: For instance, an agent can submit the same bid through two identities in a second-price auction. If those two identities are the highest bidders, the agent pays her own bid, so the outcome effectively replicates a first-price bid.

3.3 Sealed and Open Bidding

Next, we look into the question whether some form of open bidding could emerge in an ePBS world. There are two fundamental and related questions to be answered:

1. In the presence of a credible sealed bid first-price bidding channel, can an alternative auction format with open bidding survive?
2. If in the first-price bidding channel (generally we think here of the in-protocol channel), the seller could choose to leak received bids, would he do so?

The first question is the English auction version of the question we studied in the last section. The analysis in this section applies to both single-homing and multi-homing environments; the participation distinction is orthogonal to the disclosure question. The answer in the symmetric case with equally fast bidders is that the English auction will unravel (or all bidders will bid at the "last" possible moment, essentially turning the auction into a sealed-bid first-price auction). In the asymmetric case, we will observe partial un-raveling: the slow bidder will self-select into the sealed-bidding channel, to avoid leaking information to bidders with a last look. The single- vs multi-homing distinction does not matter for these kinds of results. The conclusion from these results is that in the presence of a credible sealed-bid first-price bidding channel, the overall market is pushed in one way or another to a sealed-bid first-price auction architecture.

The answer to the second question will add important qualifications to the conclusion drawn from the answer to the first question: whether or not the seller has an incentive to keep the first-price bidding channel sealed depends on how many bidders have a last look advantage. In the symmetric case, as well as in the case of one fast bidder with last look, the seller has no incentive to leak bids. In the case of at least two fast bidders, it generally has an incentive to leak bids. Practically, we could imagine different ways of seller bid leakage in ePBS, e.g. the proposer could forward bids to a relay that integrates them into its bid stream.

However, if the seller can credibly commit to not leak information (e.g. through reputation, TEE commitments or delegation to an intermediary such as a possible "first-price sealed-bid relay") it is optimal for him to do so.

We formalize the unraveling of open-bidding relays in Propositions 7 and 14, and the seller's disclosure incentives (without and with commitment) in Propositions 8 and 13 and Theorem 9.

3.3.1 No Last Look

We begin with a deliberately relay-friendly idealization. The relay is allowed to run a fully converged English auction among its participants, but the openness of the auction is purely internal: after the relay selects its representative, the relay representative and the native-channel builders submit proposer-facing bids without observing each other's final bids. Thus the model removes any cross-channel last-look advantage. The purpose of this benchmark is to ask whether an English relay can be stable even in this favorable case. Proposition 7 shows that it cannot: in large markets, all builders joining the English relay is not an equilibrium.

Fix a first-stage participation profile. Let S be the set of builders who choose the relay, and let $A = [n] \setminus S$ be the set of builders who choose the native first-price channel. Write $s = |S|$ and $m = |A|$.

If $s \geq 2$, the relay runs an internal English knockout auction. Let Y_S be the highest value among relay participants and let Z_S be the second-highest value. The internal auction selects the builder with value Y_S and sets the internal English-auction price equal to Z_S. The relay winner then chooses a proposer-facing bid b_S. Since bidding below Z_S would not be consistent with the internal English outcome, while bidding above Y_S is weakly dominated, we restrict attention to bids satisfying

$$Z_S \leq b_S \leq Y_S.$$

If $s = 1$, the relay has no internal competition and therefore generates no English-auction information; the sole relay participant simply chooses a proposer-facing bid subject to the usual no-overbidding constraint.

Each builder $j \in A$ submits a native first-price bid b_j to the proposer. The proposer compares the relay bid b_S, when present, with the native bids $(b_j)_{j \in A}$ and accepts the highest bid. The key modeling restriction is that the relay representative chooses b_S without observing the native bids, and native builders bid without observing the internal relay state. Hence the English auction reveals information only within the relay and creates no last-look opportunity across channels.

To test stability of the all-English profile, let $U_E^n(F)$ be a builder's ex-ante payoff when all n builders choose the relay, and let $U_{\text{out}}^n(F)$ be the ex-ante payoff of a single builder who deviates to the native first-price channel while the other $n-1$ builders remain in the relay. The all-English profile can be a first-stage equilibrium only if

$$U_E^n(F) \geq U_{\text{out}}^n(F).$$

The all-first-price profile is a Bayes–Nash equilibrium of the two-stage game: a unilateral deviation to the relay creates no internal relay competition and hence no informational advantage, so the deviation is payoff-equivalent to remaining in the native first-price channel. The all-English profile is less robust. For compactly supported distributions with non-flat upper tails, full participation in the English relay is unstable once the number of builders is sufficiently large.

▶ **Proposition 7** (Asymptotic instability of all-English). *There exists $N(F) < \infty$ such that, for every $n \geq N(F)$, the all-English profile is not a first-stage equilibrium.*

Therefore, even in the idealized no-last-look benchmark, the relay cannot rely on single-homing alone to make an English format stable against a native sealed first-price alternative. The next question is what happens when the relay or proposer can create a genuine last-look opportunity by disclosing early bids.

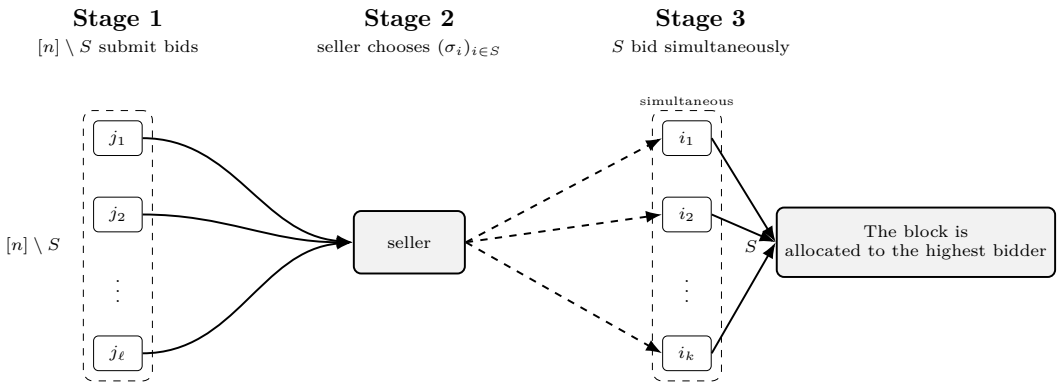


Figure 1 Last look vs First-price auction game representation.

3.3.2 Last Look

We now move to the more realistic case in which a subset of builders is latency-advantaged. These builders can submit a final response after observing information released close to the auction deadline; in this sense, they have the technological ability to obtain *last look*. Last look is not automatic, however. It matters only if the seller, the relay, or the auction format reveals relevant bid information before the deadline.

We first consider the case in which the seller, interpreted here as the block proposer, cannot credibly commit ex-ante to a disclosure strategy. As highlighted in Footnote 3, this does not exclude the possibility that the seller chooses their strategy ex-ante, but it forces them to choose a (Nash) equilibrium strategy. Let $L \subseteq [n]$ denote the set of builders who are technologically able to respond to late information before the auction closes. A builder in L does not automatically have last look; it obtains last look only if relevant information is revealed before the deadline. In the model below, we write $S \subseteq L$ for the builders who may receive such disclosures.

The game has three stages. First, the builders in $[n] \setminus S$ submit bids to the seller. Second, after observing these bids, the seller chooses a verifiable message σ_i to send to each bidder $i \in S$, where σ_i consists of a collection of bid pairs $\{(j, b_j)\}$ corresponding to an arbitrary subset of $[n] \setminus S$. Third, the bidders in S bid simultaneously. The block is allocated to the highest bidder under a pay-as-bid rule, with ties broken lexicographically in favor of bidders in S.[6]

Figure 1 summarizes the three-stage game.

We say that an auction is *k-leakage-resistant* if, for every set S with $|S| = k$, in every sequential Bayes–Nash equilibrium the seller reveals no information in the second stage. We say that it is *k-leaking* if for every set S with $|S| = k$, and every sequential Bayes–Nash equilibrium, the seller leaks information with probability one.

▶ **Proposition 8.** *The first-price auction is* 1*-leakage-resistant.*

Proof. Let $S = \{i\}$ and let r denote the highest bid submitted in stage 1. If the seller reveals r, then bidder i can condition on it and, whenever $v_i \geq r$, win by bidding exactly r, since ties are broken in favor of S. Thus leakage yields revenue exactly r. If instead the

[6] We could formulate a more general version of the game by adding a fourth stage in which the seller chooses which bid to accept. However, in any sequential equilibrium, the seller would choose the highest bid. Hence, without loss of generality, we restrict attention to the three-stage game.

seller reveals nothing, bidder i must choose his bid without observing the realized value of r. Hence, with positive probability, his equilibrium bid is strictly above r, and on that event the seller receives strictly more than r. Therefore the seller's expected revenue is strictly higher under no leakage, so in any sequential equilibrium the seller does not leak. ◀

With a unique latency-advantaged builder, the seller's incentive is compatible with no disclosure: the first-price allocation is preserved, builders are ex-ante symmetric, and the mechanism weakens incentives to compete purely on latency rather than on order-flow acquisition or block construction. However, with multiple latency-advantaged builders this is no longer true.

▶ **Theorem 9.** *The first-price auction is* k-leaking *for all* $2 \le k \le n-1$.

Proof of the theorem. Let us fix a sequential Bayes–Nash equilibrium and write $s = |S|$. Let R be the random variable

$$R := \max\{b_j : j \in [n] \setminus S\},$$

that is, the highest bid among the bidders in the first stage. Let M denote the seller's message in the second stage, with $M = \emptyset$ denoting the event that the seller reveals no information at that stage. Define

$$\overline{r} = \operatorname{ess\,sup}(R \mid M = \emptyset).$$

Observe that first-stage bids are bounded away from $\overline{v}$. Fix $a \in (\underline{v}, \overline{v})$, let $q = F(a)^{n-1}$, and choose $\eta > 0$ such that $(\overline{v} - \eta - a)q > \eta$. A bidder with value $v \ge \overline{v} - \eta$ can bid a and win whenever all other values are below a, obtaining a payoff at least $(\overline{v} - \eta - a)q > \eta$. On the other hand, any bid $x > \overline{v} - \eta$ gives $v - x < \eta$. Hence no high type bids above $\overline{v} - \eta$, while lower types do not do so by no-overbidding. Therefore $R \le \overline{v} - \eta$ almost surely, and so $\overline{r} < \overline{v}$.

▶ **Lemma 10.** *Fix the continuation game among the bidders in* S *after stage 2. Consider the two cases in which the seller either reveals the realized highest first-stage bid* r *to all bidders in* S*, or reveals nothing. In both cases, every equilibrium outcome is outcome-equivalent to a symmetric Bayes–Nash equilibrium. Moreover, every symmetric mixed Bayes–Nash equilibrium is outcome-equivalent to a pure symmetric Bayes–Nash equilibrium.*

Proof. Conditional on either public message, the bidders in S are ex-ante identical, their values are still i.i.d. according to F, and the continuation game is a continuous first-price auction game with a stochastic or static reserve price. Hence the symmetrization argument in [5] applies, so we may restrict attention to symmetric equilibria. The same result also implies that any symmetric mixed equilibrium is outcome-equivalent to a pure symmetric equilibrium. Since the seller's continuation payoff depends only on the induced distribution of bids and payments, this replacement is without loss of generality. ◀

By Lemma 10, we may assume without loss of generality that the continuation bidding strategies are symmetric and pure.

We compare the continuation game after no disclosure with the continuation game in which the seller reveals the realized highest bid to all bidders in S. Let the corresponding symmetric pure bidding strategies be $b_\emptyset$ and b_r.

When the seller reveals $M = r$ to all bidders in S, the continuation game is the standard first-price auction with reserve r. Its symmetric equilibrium is unique, and each player has bidding strategy

$$b_r(v) = \begin{cases} v - \dfrac{\int_r^v F(t)^{s-1}dt}{F(v)^{s-1}}, & v \ge r, \\ 0, & v < r. \end{cases}$$

Now consider the continuation game after no disclosure. Let

$$H(x) = \Pr[R \le x \mid M = \emptyset]$$

be the cdf of R conditional on $M = \emptyset$ and $b_\emptyset$ the bid function of the symmetric Bayes-Nash equilibrium with $M = \emptyset$. Define

$$v^\star = \inf\{v : b_\emptyset(v) \ge \overline{r}\}.$$

A first observation is that $v^\star > \overline{r}$. Indeed, in a first-price auction bidding above one's value is weakly dominated, so $b_\emptyset(v) \le v$ for every v, which already implies $v^\star \ge \overline{r}$. If $v^\star = \overline{r}$, then by continuity we would have $b_\emptyset(\overline{r}) = \overline{r}$. But then a bidder of type $\overline{r}$ obtains payoff 0 from bidding $\overline{r}$, while by shading slightly below $\overline{r}$ she still beats R with positive probability under $M = \emptyset$ and wins against the other bidders in S with positive probability, yielding strictly positive payoff. This contradicts optimality, so $v^\star > \overline{r}$.

Case 1: $v^\star < +\infty$. For all $v \ge v^\star$, equilibrium bids satisfy $b_\emptyset(v) \ge \overline{r}$, so such bids beat the first-stage maximum whenever $M = \emptyset$. Hence, if a bidder of type v deviates to the bid of a type $z \ge v^\star$, her payoff is

$$(v - b_\emptyset(z))F(z)^{s-1}.$$

Optimality at $z = v$ gives

$$\frac{d}{dz}\Big((v - b_\emptyset(z))F(z)^{s-1}\Big)\Big|_{z=v} = 0,$$

that is,

$$-b'_\emptyset(v)F(v)^{s-1} + (v - b_\emptyset(v))(s-1)F(v)^{s-2}f(v) = 0.$$

Equivalently,

$$\frac{d}{dv}\Big((v - b_\emptyset(v))F(v)^{s-1}\Big) = F(v)^{s-1}.$$

Using the boundary condition $b_\emptyset(v^\star) = \overline{r}$, we obtain

$$b_\emptyset(v) = v - \frac{(v^\star - \overline{r})F(v^\star)^{s-1} + \int_{v^\star}^{v} F(t)^{s-1}dt}{F(v)^{s-1}}.$$

We now compare the two bid functions. For every $v \ge v^\star$,

$$b_{\overline{r}}(v) - b_\emptyset(v) = \frac{(v^\star - \overline{r})F(v^\star)^{s-1} - \int_{\overline{r}}^{v^\star} F(t)^{s-1}dt}{F(v)^{s-1}}.$$

Since F is strictly increasing, we have $F(t)^{s-1} < F(v^\star)^{s-1}$ for all $t \in [\overline{r}, v^\star)$, and therefore

$$\int_{\overline{r}}^{v^\star} F(t)^{s-1}dt < (v^\star - \overline{r})F(v^\star)^{s-1}.$$

Hence

$$b_{\overline{r}}(v) > b_\emptyset(v) \qquad \text{for all } v \ge v^\star.$$

Also, if $v \in (\overline{r}, v^\star)$, then by definition $b_\emptyset(v) < \overline{r}$, whereas

$$b_{\overline{r}}(v) = v - \frac{\int_{\overline{r}}^{v} F(t)^{s-1}dt}{F(v)^{s-1}} > \overline{r}.$$

Thus

$$b_{\overline{r}}(v) > b_\emptyset(v) \qquad \text{for all } v > \overline{r}.$$

Case 2: $v^\star = +\infty$. In this case, for all $v \in [\underline{v}, \overline{v}]$, $b_\emptyset(v) < \overline{r}$. However, for $v \in [\overline{r}, \overline{v}]$, $b_{\overline{r}}(v) \geq \overline{r}$, and therefore $b_{\overline{r}}(v) > b_\emptyset(v)$.

Let v_i denote the valuation of bidder $i \in S$. The seller's continuation revenue after revealing r is

$$\Pi_{\text{rev}}(r) = \mathbb{E}\Big[\max\{r, \max_{i\in S} b_r(v_i)\}\Big],$$

whereas after no disclosure it is

$$\Pi_\emptyset(r) = \mathbb{E}\Big[\max\{r, \max_{i\in S} b_\emptyset(v_i)\}\Big].$$

From the pointwise comparison above, for every valuation profile $(v_i)_{i\in S}$,

$$\max\{\overline{r}, \max_{i\in S} b_{\overline{r}}(v_i)\} \geq \max\{\overline{r}, \max_{i\in S} b_\emptyset(v_i)\},$$

and the inequality is strict whenever $\max_{i\in S} v_i > \overline{r}$. Since $\overline{r} < \overline{v}$,

$$\Pr\Big[\max_{i\in S} v_i > \overline{r}\Big] = 1 - F(\overline{r})^s > 0.$$

Therefore

$$\Pi_{\text{rev}}(\overline{r}) > \Pi_\emptyset(\overline{r}).$$

The seller's continuation revenue is continuous in r in both cases. Hence there exists $\delta > 0$ such that

$$\Pi_{\text{rev}}(r) > \Pi_\emptyset(r) \qquad \text{for all } r \in [\overline{r} - \delta, \overline{r}].$$

By definition of $\overline{r}$ as the essential supremum of R conditional on $M = \emptyset$,

$$\Pr[R \in [\overline{r} - \delta, \overline{r}] \mid M = \emptyset] > 0.$$

Hence, on a positive-probability set of histories at which the equilibrium prescribes no disclosure, the seller would strictly prefer to reveal the realized highest bid. This contradicts sequential optimality. Therefore the seller cannot choose $M = \emptyset$ with positive probability. ◀

The previous theorem shows that the allocation induced by the three-stage game is not equivalent to the allocation in the standard first-price auction. Since the first-price auction is welfare-optimal with symmetric i.i.d. bidders, the three-stage game is not welfare-optimal. Moreover, when value distributions are regular, the first-price auction is revenue-optimal within the class of auctions without a reserve price (see Proposition 13). Thus allowing the seller to leak information cannot improve on the no-disclosure benchmark; the theorem implies that revenue is strictly lower. Also, information disclosure can have an adverse effect on competition in the block-building market by increasing market concentration. As we show in the following proposition, disclosure raises the payoff of agents with an informational advantage, thereby strengthening their position relative to less-informed agents.

▶ **Proposition 11** (Last-look rents). *Suppose the seller reveals the highest first-stage bid to every bidder in S. Then, in any symmetric equilibrium, every bidder in S obtains a strictly larger ex-ante utility and probability of winning than every bidder in $[n] \setminus S$.*

Proof. As in Lemma 10, the symmetrization argument in [5] applies to the bidders in $A := [n] \setminus S$. Let $m := |A|$ and $s := |S|$, so that $m + s = n$. Therefore, we can reduce the analysis to the case in which bidders in A use a monotone bidding strategy b_A and bidders in S use a monotone bidding strategy b_S. Let $X_A(v)$ and $X_S(v)$ denote the interim winning probabilities of a bidder with value v in groups A and S, respectively.

Almost surely, every player bids strictly below her valuation. In particular, $b_A(v) < v$. A bidder in A with value v wins only if the other bidders in A have values below v and the bidders in S have values below $b_A(v)$. Hence

$$X_A(v) = F(v)^{m-1} F(b_A(v))^s \le F(v)^{m-1} F(v)^s = F(v)^{n-1},$$

with strict inequality on a positive-probability set.

By contrast, a bidder in S with value v wins whenever the other bidders in S have values below v and the highest first-stage bid is at most v. Therefore

$$X_S(v) = F(v)^{s-1} \Pr\Big[\max_{j \in A} b_A(v_j) \le v\Big].$$

Since $b_A(t) \le t$, the event that all bidders in A have values below v implies $\max_{j \in A} b_A(v_j) \le v$. Hence

$$X_S(v) \ge F(v)^{s-1} F(v)^m = F(v)^{n-1}.$$

Thus $X_S(v) > X_A(v)$ on a positive-measure set. By the envelope theorem,

$$u_S(v) = \int_{\underline{v}}^{v} X_S(t)\, dt \ge \int_{\underline{v}}^{v} X_A(t)\, dt = u_A(v),$$

with strict inequality on a positive-measure set. Taking expectations over $v \sim F$ gives the result. ◀

3.3.3 Last-Look with Seller Commitment

The previous section studied the case in which the seller could not commit to a disclosure policy. In that environment, disclosure is chosen history by history after early bids have arrived. We showed that this creates an ex-post incentive to leak information when there are at least two latency-advantaged bidders: after observing the early bids, the seller can raise continuation revenue by revealing information to the bidders who are still able to respond.

We now ask a different question. Suppose instead that the seller can commit ex-ante to a particular auction rule, or equivalently can choose among relays that commit to different rules (and make it credible that they will ignore or not-leak direct bids sent to them). One relay may run a sealed first-price auction, another may run an open English-style auction, and another may commit to giving last look to a specified set of builders. In this sense, the seller's decision is over which auction format they prefer to pick ex-ante and commit to.

This distinction matters. Ex-post, after early bids have been observed, leaking information can be profitable. Ex-ante, however, the proposer must take into account that last look changes bidders' incentives and shifts surplus toward the latency-advantaged builders. The result below shows that, when values are i.i.d. from a regular distribution, this ex-ante effect dominates: among committed last-look rules, the proposer weakly prefers the sealed first-price auction.

We assume that the seller can choose among the following auction formats. For a set $S \subseteq [n]$, define the auction that reveals bids from $[n] \setminus S$ to bidders in S as follows:

1. The set of builders $A := [n] \setminus S$ submits bids simultaneously.
2. The bids submitted by A are revealed to the builders in S.
3. The builders in S then submit bids simultaneously.
4. The right to decide the content of the next block is allocated to the highest bidder under a pay-as-you-bid payment rule.

Subsequently, we use the short-hand notation $\texttt{Show}(S)$ to denote the auction that reveals bids from $[n] \setminus S$ to bidders in S. Observe that the cases $S = \emptyset$ and $S = [n]$ are equivalent to the first-price auction. For the set of latency-advantaged parties $L \subseteq [n]$, one can view the case $S = L$ as the current status quo, i.e. an open first-price auction with last-look advantage for a set of parties.

First, we assume that the block proposer can *commit* to a specific auction format; that is, once they announce to whom they reveal bids, they cannot deviate from it during its execution.

▶ **Definition 12.** *A distribution F is* regular *if its virtual valuation function*

$$\phi(v) = v - \frac{1 - F(v)}{f(v)}$$

is non-decreasing in v, where f is the density of F.

▶ **Proposition 13.** *If F is regular, then for any proper set $S \subseteq [n]$ and any Bayes–Nash equilibrium of $\texttt{Show}(S)$,*

$$\mathrm{Rev}(\mathrm{FP}) \geq \mathrm{Rev}(\texttt{Show}(S)).$$

In other words, under commitment the block proposer weakly prefers the sealed first-price auction.

Suppose, finally, that the proposer commits not to leak bids submitted through the sealed first-price channel, but still accepts bids coming from an open-bid relay. In this case, a high-latency builder has no reason to send her bid through the open relay. If she sends a bid b through the relay, low-latency builders may observe it and still have time to react. If she sends the same bid b through the sealed channel, the proposer compares it in the same way, but the bid is not revealed before the deadline. Hence, whenever the builder has a chance of winning, submitting through the relay can only reduce that chance of winning.

▶ **Proposition 14** (Open relays under no leakage)**.** *Suppose the proposer commits not to leak bids submitted through the sealed first-price channel, while still accepting bids from an open-bid relay. Then high-latency builders bid through the sealed channel. Low-latency builders may still use or observe the open relay, but the relay no longer gives them a last-look advantage over sealed bids. The outcome is equivalent to the standard sealed first-price auction.*

4 Conclusion

We have analyzed competing auctions in intermediated markets where a single seller can route a single sale through multiple parallel mechanisms, with proposer-builder separation as the leading application. We found that second-price intermediary auctions unravel fully against a sealed first-price principal auction (Theorem 1 under single-homing, Theorem 4 under multi-plexing), open-bidding intermediaries unravel partially – collapsing into first-price

under symmetric latency, sorting fast bidders to the intermediary under asymmetric latency (Propositions 7 and 14) – and Sybil bidding can restore the second-price intermediary by allowing bidders to convert it into a first-price-equivalent mechanism. On the disclosure side, a first-price auction is leakage-resistant against a single privileged bidder (Proposition 8) but leaks in equilibrium against two or more (Theorem 9), and the optimal commitment under regularity is to share no information at all (Proposition 13).

The institutional reading of these results is that ePBS will likely reshape the relay market: An in-protocol sealed-bid first-price channel limits the design space for relays going forward. Relays running second-price or open-bidding mechanisms cannot sustain participation against the sealed first-price alternative. However, this prediction relies on the in-protocol channel being credibly sealed; the assumption of a non-leaking in-protocol bidding channel is itself fragile: the disclosure analysis identifies that a proposer receiving direct bids has an incentive to leak bids in several scenarios. Thus, proposers forwarding bids to a relay's open-bid feed is a possible outcome. On the other hand, there is value for proposers from a credible commitment not to leak bids. This could be created through reputation mechanisms or solutions such as TEEs, in the case of stakers large enough to provide individual commitments (Lido and similar institutional stakers), or through intermediation in the form of "sealed-bid" relays. Thus, there are several factors pushing relays to effectively move to sealed-bid first-price auction formats post ePBS. Still, there is a potential role for them as credible intermediaries and for services such as fast bid and block propagation as well as non-collateralized bidding. There is also, in principle, the narrower path to maintain an open-bid relaying service to which proposers forward bids. However, the viability of this latter path depends crucially on the nature of competition in the builder market.

Several limitations of our analysis are worth flagging. We do not model collusion among builders, which is a standard concern for sealed-bid auctions in classical settings but plausibly less binding in blockchain environments where on-chain transparency and programmable coordination weaken the open-vs-sealed distinction as a collusion-suppression tool. We do not model the searcher-builder pipeline upstream of the auction; our values are private at the builder level, which is the relevant abstraction for the auction itself but abstracts from common-value dynamics among searchers. We treat each slot in isolation rather than as part of a repeated game; reputational mechanisms that sustain relay commitment are mentioned but not formally modeled. The analysis also takes the set of available mechanisms and intermediaries as given; the entry and design problem for relays – what mechanism a new relay should propose, given the equilibrium predictions above – is a natural follow-up question.

References

1 Mark Armstrong. Competition in two-sided markets. *The RAND journal of economics*, 37(3):668–691, 2006.

2 Dirk Bergemann and Martin Pesendorfer. Information structures in optimal auctions. *Journal of economic theory*, 137(1):580–609, 2007. doi:10.1016/J.JET.2007.02.001.

3 Roberto Burguet and József Sákovics. Imperfect competition in auction designs. *International Economic Review*, 40(1):231–247, 1999.

4 Bernard Caillaud and Bruno Jullien. Chicken & egg: Competition among intermediation service providers. *RAND journal of Economics*, pages 309–328, 2003.

5 Shuchi Chawla and Jason D Hartline. Auctions with unique equilibria. In *Proceedings of the fourteenth ACM conference on Electronic commerce*, pages 181–196, 2013. doi:10.1145/2492002.2483188.

6 Francesco D'Amato, Nico Flaig, Barnabé Monnot, Michael Neuder, Potuz, Justin Traglia, and Terence Tsao. EIP-7732: Enshrined Proposer-Builder Separation. `https://eips.ethereum.org/EIPS/eip-7732`, 2024. Draft EIP, accessed April 2026.

7 Glenn Ellison, Drew Fudenberg, and Markus Möbius. Competing auctions. *Journal of the European Economic Association*, 2(1):30–66, 2004.

8 Thierry Foucault, Marco Pagano, and Ailsa Röell. *Market liquidity: theory, evidence, and policy*. Oxford University Press, 2023.

9 Bruno Mazorra, Burak Öz, Christoph Schlegel, and Fei Wu. The free option problem of epbs. *arXiv preprint arXiv:2509.24849*, 2025. `doi:10.48550/arXiv.2509.24849`.

10 R Preston McAfee. Mechanism design by competing sellers. *Econometrica: Journal of the econometric society*, pages 1281–1312, 1993.

11 Paul Milgrom and Robert J Weber. The value of information in a sealed-bid auction. *Journal of Mathematical Economics*, 10(1):105–114, 1982.

12 Maureen O'Hara and Mao Ye. Is market fragmentation harming market quality? *Journal of Financial Economics*, 100(3):459–474, 2011.

13 Roel Oomen. Last look. *Quantitative Finance*, 17(7):1057–1070, 2017.

14 Mallesh Pai and Max Resnick. Structural advantages for integrated builders in mev-boost. In *International Conference on Financial Cryptography and Data Security*, pages 128–132. Springer, 2024. `doi:10.1007/978-3-031-69231-4_9`.

15 Michael Peters and Sergei Severinov. Competition among sellers who offer auctions instead of prices. *Journal of Economic Theory*, 75(1):141–179, 1997.

16 Jean-Charles Rochet and Jean Tirole. Two-sided markets: a progress report. *The RAND journal of economics*, 37(3):645–667, 2006.

Just-In-Time Resale in an Ahead-Of-Time Auction for Faster Execution

Burak Öz
Flashbots, Berlin, Germany

Akaki Mamageishvili
Offchain Labs, Zurich, Switzerland

Christoph Schlegel
Flashbots, Zurich, Switzerland

Ali Taslimi
Entropy Advisors, London, UK

Abstract

We study Arbitrum's Timeboost auction, an ahead-of-time mechanism that sells a 200 ms ordering advantage in an otherwise first-come, first-served transaction ordering policy. The market naturally divides into two phases: a *competition phase*, in which the dominant searchers compete directly in the primary auction, and a *coordination phase*, in which they source fast-lane access through Kairos, a Just-in-Time resale intermediary. We use auction bids, time-boosted transactions, and on-chain payment traces to study how well ahead-of-time bids predict realized CEX–DEX arbitrage profits and how the emergence of resale changes surplus allocation.

We find that ahead-of-time bids are noisy predictors of short-horizon arbitrage profits. During the competition phase, bid–profit correlations are statistically significant but economically modest at the round level, while correlations increase when profits are aggregated over longer horizons. This suggests that bidders can identify favorable market conditions but face substantial uncertainty about the realized value of any individual one-minute fast-lane interval. After the Kairos transition, competition in the primary auction weakens sharply: in our focused transition-analysis window, paid bids fall from 62.7% of the top bid before the transition to 14.8% thereafter, while total searcher profits remain broadly similar. The resulting dynamics are most consistent with coordination through a common intermediary rather than direct competition in the primary auction. More broadly, our findings suggest that ahead-of-time allocation mechanisms can be vulnerable to secondary-market intermediation when competition among dominant participants is weak.

2012 ACM Subject Classification Applied computing → Online auctions

Keywords and phrases ahead-of-time auctions, secondary markets, Timeboost, arbitrage

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.20

Related Version *Full Version*: `https://arxiv.org/abs/2603.20175` [26]

1 Introduction

Many blockchains use auctions as part of their transaction ordering mechanism. In Ethereum Layer-1 (L1), *Proposer Builder Separation* (PBS) implements a builder auction in which specialized parties called *block builders* bid for the right to determine the content of the next proposed block. In most Layer-2 (L2) rollups, transactions are sorted in decreasing order of priority fees per gas. This mechanism is commonly called *Priority Ordering* or *Priority Gas Auction*, since it amounts to an auction for earlier execution. Despite significant differences – in Ethereum, builders bid for the right to order an entire block while in rollups the auction concerns relative ordering of individual transactions, and Ethereum has a public mempool while rollups do not – these formats share a common feature: they are *Just-in-Time* (JIT), meaning participants can submit bids until immediately before block content is finalized.

© Burak Öz, Akaki Mamageishvili, Christoph Schlegel, and Ali Taslimi;
licensed under Creative Commons License CC-BY 4.0

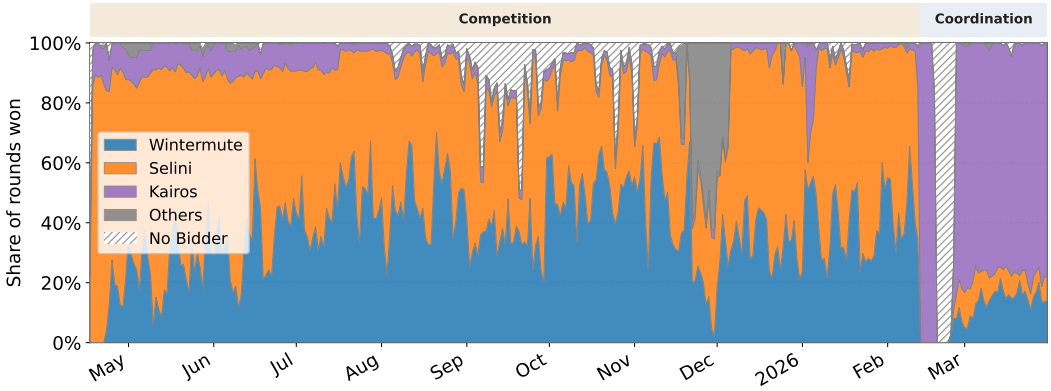


Figure 1 Daily shares of Timeboost auction rounds won by each participant (April 2025–March 2026). The header strip delineates the *competition phase,* in which Wintermute and Selini win the majority of rounds, from the *coordination phase*, in which Kairos takes over as the dominant winner. Hatched regions indicate rounds with no active bidders.

An alternative design, *ahead-of-time* (AOT) auctions, introduces a substantial time gap between the auction and the determination of transaction order. In the Ethereum context, such mechanisms have been discussed as slot auctions: a bidder wins the right to propose the next block at auction time but commits to no specific block content until later (typically 12 seconds later) [14]. In the rollup ecosystem, Arbitrum's Timeboost [2] is the first major deployed instance of a transaction ordering policy with an AOT component.

Prior work on Ethereum slot auctions has argued [15, 22] that selling the right to propose blocks in advance may lead to secondary markets for proposal rights that benefit certain searchers. It has also been argued that predicting block value far in advance – as AOT bidders must – is inherently difficult and may lead to inefficient value capture, which can itself be the catalyst for such secondary markets to emerge. As the only major deployed AOT arbitrage auction for transaction ordering, Arbitrum's Timeboost provides a natural empirical case study for evaluating these claims.

Since its launch in April 2025, the Timeboost auction has been dominated by two searchers, Wintermute and Selini Capital, who specialize in non-atomic arbitrage and have won the vast majority of auction rounds (see Figure 1). Kairos [10], a service that participates in the primary auction and resells fast-lane access at the transaction level JIT, attracted little traction for most of Timeboost's existence. This changed sharply in mid-February 2026, when Wintermute and Selini began sourcing fast-lane access through Kairos' resale market and reduced or ceased primary-auction bidding, after which Kairos began winning the majority of auction rounds.

More precisely, we observe the following sequence of events. On February 12, Wintermute and Selini stopped participating in the primary auction or significantly reduced their bids, leaving Kairos winning nearly all auction rounds at prices close to the reserve. In response, Arbitrum substantially increased the reserve price on February 18, after which no participant was willing to pay it for some period. Bidding later resumed in a small fraction of rounds before the market settled into a new stable regime when Arbitrum lowered the reserve price on February 25. In this regime, Kairos wins approximately 78% of auction rounds, with occasional but limited participation from Wintermute and Selini.

This naturally divides our study into two phases: a *competition phase* prior to February 12, which lets us assess AOT auction effectiveness in the absence of significant JIT resale, and a *coordination phase* thereafter, in which dominant searchers source fast-lane access through a

common JIT intermediary rather than competing directly in the primary auction. We use *coordination* in this descriptive sense and do not mean to imply that we observe an explicit agreement among the participants.

We find that, during the competition phase, bids are only weakly correlated with estimated markout arbitrage profit and loss (PnL), consistent with the difficulty of predicting short-horizon value in an AOT setting. We observe that the two dominant searchers route through Kairos's JIT resale market, allowing them to obtain fast-lane access without directly committing to price-uncertain forward bids. Because the associated off-chain payments are unobserved, however, we cannot determine whether this shift increased their net payoffs. In our focused transition-analysis window, paid bids in the primary auction fall to 14.8% of the top bid after the transition, down from 62.7% in the February 1–12 pre-transition segment, while estimated arbitrage PnL remain broadly similar across regimes. This indicates that the primary auction captures a smaller share of the value reflected in submitted bids, although the final allocation of surplus between Kairos and the searchers remains unobserved. The observed dynamics are consistent with a shift from direct competition in the primary auction toward routing through a common intermediary. More broadly, our findings suggest that AOT allocation mechanisms may be particularly susceptible to secondary-market intermediation when competition among dominant participants is weak.

1.1 Related Work

In Ethereum, the next block proposer sorts transactions unilaterally. This allows the proposer to extract value from the power to order transactions, commonly referred to as Maximum Extractable Value (MEV) [5]. Transaction ordering through auctions is theoretically analyzed in [23]. An early version of Timeboost without an AOT auction was proposed in [16]. Spam levels before and after Timeboost are studied in [4], which finds that spam decreased after its introduction. The time advantage of centralized exchange (CEX) to decentralized exchange (DEX) arbitrage under a First-Come-First-Serve (FCFS) ordering policy is studied in [9], with comparisons to PGA auctions and pure FCFS. Latency advantage in a PBS setting are examined in [21, 24]. Possible downsides of AOT auctions are identified in [22] for settings where a secondary market allows JIT resale of AOT-acquired rights.

For methodology, we follow [25] in using markouts at CEX mid-prices to estimate CEX–DEX arbitrage profits. Theoretical predictions of such profits are developed in [20]. Our theoretical background for auctions with common-value component draws on the classical work of [19].

1.2 Background

Timeboost. Timeboost has been deployed on the Arbitrum rollup since April 17, 2025. Under this mechanism, the protocol sells fast-lane access in a permissionless second-price auction with a reserve price. The fast-lane integrates with Arbitrum's FCFS transaction ordering policy and gives the auction winner a 200 ms advantage for one minute. The auction for the next minute begins at second 0 of each round and lasts until second 45 [1]. The remaining 15 seconds are used for on-chain auction settlement. The rule is enforced by the sequencer, which executes regular transactions with a 200 ms delay, while fast-lane transactions from the Timeboost winner are executed immediately upon arrival. In effect,

[1] Each round corresponds to a minute shifted by 9 seconds relative to the wall-clock minute.

the sequencer merges the two FCFS queues: one for time-boosted transactions and one for regular transactions. Empirically, most fast-lane activity and extracted value can be attributed to CEX–DEX arbitrage [18]. In particular, Wintermute exclusively conducts such arbitrage, while Selini occasionally uses fast-lane for other forms of value extraction methods.

Kairos. After winning a Timeboost round, Kairos organizes fast-lane "sub auctions" with bidding phases of approximately 100 ms. Transactions within each 100 ms batch are ordered by profitability, and the batch is submitted to the Arbitrum sequencer via the fast-lane, thereby giving all transactions in the batch a time advantage of at most 100 ms over regular transactions. The exact advantage depends on the latency between Kairos's servers and the sequencer. As long as this latency remains below 100 ms, a competing transaction in the regular FCFS queue cannot be executed before the Kairos batch.

Payments to Kairos are typically made on-chain as explicit ETH transfers embedded in time-boosted transactions and sent to a designated contract address[2]. An important exception is Kairos's parallel lane [11], which is designed for CEX–DEX arbitrage searchers and operates via a subscription model with off-chain payments. This design bypasses the simulation step in which searchers' transactions are continuously re-submitted under FCFS, thereby reducing latency and lowering the risk of losing priority to regular transactions. Since these transactions constitute a substantial share of all time-boosted transactions in our dataset, the absence of observable on-chain payments prevents us from tracking transfers to Kairos by the two main searchers, as we discuss further below.

2 Conceptual Framework and Empirical Predictions

We develop a conceptual framework to organize and guide our empirical analysis. We first use a standard interdependent-values auction model to discuss how common- and bidder-specific value components affect AOT bidding. We then relate expected arbitrage value to short-run price volatility. Finally, we identify several mechanisms through which a JIT resale market such as Kairos may emerge and derive their distinct empirical implications. The framework is intended to provide qualitative benchmarks rather than a complete structural model of the Timeboost market.

2.1 Common-Value Second-Price Auctions

The Timeboost auction has a clear common-value component: the market-wide arbitrage opportunities arising during a Timeboost interval are, in principle, available to all bidders. At the same time, valuations may contain bidder-specific components. Bidders may face private inventory constraints, hold accumulated positions that they want to unwind, or incur heterogeneous trading costs on external markets. For bidders that are also market makers on external exchanges, the trades and inventory accumulated through their market-making activity may further affect their value for winning the auction.

A standard framework for auctions with common- and private-value components is the interdependent-values model of [19]. Bidders $i = 1, \ldots, n$ receive private signals X_i that are i.i.d., and the value of winning the auction for bidder i is a function of all signals $v_i(X_1, \ldots, X_n)$, which is assumed to be symmetric in opponents' signals, strictly increasing in

[2] Kairos payment address: `0x60E6a31591392f926e627ED871e670C3e81f1AB8`

own signal, and nondecreasing in the others'. Let $Y_i := \max_{j \neq i} X_j$ denote the highest signal among bidder i's competitors. Under risk neutrality, the symmetric equilibrium bidding strategy in a second-price auction is

$$b(x) \;=\; \mathbb{E}[\, v_i(X_i, X_{-i}) \mid X_i = x,\; Y_i = x \,],$$

the expected value of winning conditional on the pivotal event in which the bidder's signal is exactly tied with that of the strongest competitor.

In contrast to a pure private-value auction, bidders should not simply bid their expected value given their own signal, $\mathbb{E}[v_i(X_i, X_{-i}) \mid X_i = x]$: because the value of winning depends on rivals' signals, winning itself is informative – it reveals that all competing signals were lower – and a bidder who ignores this would systematically overpay. Equilibrium bidding corrects for this winner's curse by evaluating the expected value in the pivotal event where the bidder is exactly tied with the strongest rival.

This framework motivates an empirical comparison between bids and realized arbitrage value. In equilibrium, the price paid equals b evaluated at the second-highest signal, while the top bid reflects the highest signal. If the paid bid is more closely associated with our measure of realized common arbitrage value than the top bid, this is consistent with the common-value component being an important driver of bidder valuations, since the paid bid reflects the signal of the strongest losing bidder, which is informative about realized value only to the extent that values are interdependent. Conversely, if valuations are dominated by private idiosyncratic components, we would expect bids to be less closely associated with realized common arbitrage value. We emphasize that this comparison is suggestive rather than a sharp test: the theory does not imply in general that the second-highest signal must predict realized common value better than the highest, as the relative predictive power of the two order statistics depends on the joint distribution of signals and values.

2.2 Arbitrage Value and Price Volatility

The theory of arbitrage in automated market makers (AMM) [20] can inform how much arbitrage value searchers can expect to extract over a fast-lane interval. While the theory applies to the case where bidders do not face uncertainty about the amount of arbitrage their individual trades will extract [3], quantitatively, we should expect similar results. The theory predicts that arbitrage value is proportional to price variance, i.e., expected arbitrage gains over a window of length T scale with price variance over the same window.

Cryptocurrency price volatility is typically non-stationary and is best modeled by a mean-reverting stochastic volatility process [13]. Therefore, this framework motivates the following empirical expectations. First, if price variance is linear in time, then arbitrage gains are also linear in the length of the Timeboost interval. Second, longer run averages of arbitrage gains should be easier to predict than realized gains in any short interval. This makes value prediction particularly difficult in an AOT auction, where bidders must price access before observing the exact opportunities that will arise.

[3] Under the Timeboost policy, fast-lane transactions do not know the exact state of the AMMs, as unobserved slow-lane transaction might change it compared to the state that the arbitrageur last observed.

2.3 Resale Market

We identify three main mechanisms through which a JIT resale market such as Kairos could be successful.

1. **Unbundling**: The Timeboost auction wholesales a 200 ms time advantage for an entire minute. Reselling this advantage over shorter time intervals or on a per transaction basis could, in principle, enhance efficiency by allowing different searchers to obtain time-boosts within the same minute. Ideally, this would allocate the time advantage to the searcher that can extract the most value at a given moment, rather than to a single party that extracts the most value on average over the full minute. Similar dynamics arise in L1 block-building auctions [8], where builders resell blockspace to many different searchers. In this case, additional surplus from resale would come from improved market efficiency.
2. **Non-Competitive Behavior**: In the Timeboost setting, the presence of only two dominant searchers makes the market vulnerable to tacit coordination or collusive behavior. Kairos may effectively, though not necessarily deliberately, operate as a hub in a hub-and-spoke coordination structure: the two major searchers may abstain from bidding, or bid less aggressively than they otherwise would, allowing Kairos to acquire the fast-lane at a lower price than under competitive bidding. In this case, the combined surplus of the two searchers and the reseller could exceed that in the competitive benchmark, not because more value is created, but because less revenue is paid to the seller in the primary auction, namely Arbitrum.
3. **Information**: Bidding AOT for the right to use the fast-lane over an entire minute gives bidders less information about the value they will ultimately be able to extract than bidding JIT for an immediately submitted transaction. This matters when some opportunities are searcher-specific, e.g., because a searcher identifies or can exploit opportunities that others cannot. In such cases, a near-JIT auction can generate higher expected revenue[4]. A related but separate issue arises if the reseller is also a searcher themselves, for which there is no indication that Kairos qualifies[5]. In that case, the reseller could strategically choose to resell the time-boost only after receiving an unfavorable signal about extractable value, thereby capturing even more value from operating the resale market. In this setting, additional surplus from resale would again arise through improved allocation efficiency.

These mechanisms imply different empirical patterns. The unbundling mechanism predicts a broader and more differentiated set of active searchers under Kairos, together with higher total value extraction through time-boosted transactions. The weakened-competition mechanism predicts that the same dominant searchers remain active and total value extraction stays broadly similar, while revenue in the primary Timeboost auction declines. It may also imply that the dominant searchers stop bidding competitively against each other in the primary auction. The information mechanism predicts higher value extraction from time-boosted transactions, together with a significant spread between revenue captured in the primary auction and revenue generated through Kairos.

[4] The mathematical intuition is simple: in the extreme case of a pure private-value auction, the mean of i.i.d. random variables is smaller than their expected maximum, so allocating closer to the point of realization can increase expected value extraction.

[5] [22] focus particularly on this case and argue that it can be a driver of market centralization in the searcher/builder market.

As we show below, the strongest empirical support is for the weakened-competition mechanism. After the transition to Kairos, the same searchers remain central, total value extraction changes only modestly, and primary auction revenue declines. By contrast, we find more limited evidence for unbundling or information-based efficiency gains.

3 Methodology

We now describe the datasets used in our analysis and the methodology for estimating arbitrage profits. Our primary data source is Dune Analytics [7].[6]

3.1 Datasets

Our datasets cover two complementary windows. The main auction and time-boosted transaction datasets span August 2025 through March 12, 2026. For the analysis of the Kairos transition, regular non-time-boosted CEX–DEX transactions and Kairos payments are collected over a focused February 1–March 12, 2026 window.

Timeboost auctions. We collect Timeboost auction data from August 2025 until March 12, 2026. The dataset contains 298,530 auction rounds with at least one bid. We obtain detailed bidding data from public endpoints provided by Arbitrum [1]. For each round, we observe the start and end time of the one-minute interval during which the auction winner holds fast-lane access. We use this interval to assign time-boosted transactions to auction rounds based on block time.

Transactions. In total, we identify 30,310,245 time-boosted transactions executed during our study period. Of these, 27,399,487 (90.4%) target contracts belonging to Wintermute[7] and Selini[8]. Wintermute accounts for 22,383,187 such transactions, while Selini accounts for 5,016,300. Since these contracts are associated with non-atomic CEX–DEX arbitrage, we apply a heuristic to identify transactions likely executing such strategies. We classify a transaction as CEX–DEX arbitrage if it targets these contracts, emits only one swap-related event, and trades a known liquid pair involving WETH, WBTC, USDC, or USDT. Using this heuristic, we label 11,320,741 Wintermute transactions and 3,399,555 Selini transactions as CEX–DEX arbitrage. These correspond to 50.6% and 67.8% of their respective time-boosted contract interactions. Applying the same heuristic to regular, non-time-boosted transactions targeting these contracts, we identify an additional 1,290,594 Wintermute and 316,927 Selini CEX–DEX arbitrage transactions during the February 1–March 12 analysis window.

Kairos Payments. Using Arbitrum transaction traces, we identify all transactions during the February 1–March 12 analysis window in which Kairos's payment contract receives an explicit ETH transfer with positive value. In total, we identify 39,521 such payments.

[6] The queries used in the analysis are publicly accessible: Timeboost auctions, `https://dune.com/queries/6219197`; CEX–DEX transactions, `https://dune.com/queries/6219209`; and Kairos payments, `https://dune.com/queries/6813491`.

[7] Wintermute CEX–DEX contracts: `0xcb43d843f6cadf4f4844f3f57032468aadd9b95c` and `0x27920e8039d2b6e93e36f5d5f53b998e2e631a70`.

[8] Selini CEX–DEX contract: `0xee2e7bbb67676292af2e31dffd1fea2276d6c7ba`.

3.2 Markouts and Profits

To estimate the PnL of Wintermute and Selini, we measure the realized value of their CEX–DEX arbitrage trades using markouts. The searchers' exact off-chain trading strategies are unobserved, including when and how positions accumulated on DEXes are offset on CEXes. Consider an arbitrage trade i executed on-chain at time t that purchases x units of token A for y units of token B on a DEX. Assuming the position is offloaded on a CEX after a markout horizon m at prices $P_A(t+m)$ and $P_B(t+m)$, the profit is defined as

$$\Pi(i)_{t,m} := x\,P_A(t+m) - y\,P_B(t+m) - \text{fees}.$$

Unless otherwise stated, our empirical analyses involving markouts restrict attention to trades with positive estimated PnL, thereby focusing on realized profitable arbitrage activity. For the markout-horizon sensitivity exercise, however, we report net PnL over a fixed sample of all identified trades, since applying the positive-PnL filter separately at each horizon would change the composition of the sample.

For our baseline specification, we use a markout horizon of $m = 5$ seconds. This choice reflects a trade-off: very short horizons may value the DEX leg before prices have sufficiently adjusted across on-chain and off-chain markets, whereas longer horizons are increasingly likely to incorporate price movements unrelated to the original arbitrage opportunity.

To assess the sensitivity of the markout valuation to the chosen horizon, we recompute net markout PnL over a fixed sample of all identified time-boosted trades in August 2025 using horizons of $m \in \{1, 2, 5, 10, 30\}$ seconds. As shown in the appendix of the full version [26], estimated PnL increases between one and five seconds but is comparatively stable thereafter. Relative to the five-second baseline, total estimated PnL is only 1.2% higher at ten seconds and 5.0% higher at thirty seconds. At ten seconds, the corresponding differences are 1.8% for Wintermute and 0.7% for Selini. We therefore select five seconds as the shortest horizon in this relatively stable range, allowing time for cross-market price adjustment while limiting contamination from subsequent market movements.

We obtain price data from the Binance historical data repository [3], using USDT trading pairs sampled at one-second resolution granularity. Following [25], we compute markouts using CEX mid-prices.

3.3 Limitations

Our methodology has several limitations.

1. The heuristic used to detect CEX–DEX arbitrage captures only simple swaps involving a small set of liquid assets known to trade on CEXes. In practice, non-atomic arbitrage searchers may execute more complex strategies involving a long tail of assets [25], as well as over-the-counter markets or venues on other chains [27].
2. Profit estimation assumes that the entire inventory is unwound in a single trade at the chosen markout horizon. In reality, traders may unwind positions gradually to manage price risk, or may choose to retain inventory exposure.
3. Trade coverage is limited by the DEX protocol data available on Dune Analytics.
4. The observability of payments to Kairos is limited to on-chain transfers and does not include payments made through its CEX–DEX subscription model.
5. We assume 1USDT = 1USDC = 1USD, which may not always hold in practice.

However, because the methodology is applied consistently across regimes, any resulting over- or underestimation of profits should affect all periods similarly and still allow meaningful comparisons across regimes.

Table 1 Pearson and Spearman correlations between per-round Timeboost auction bids and total positive-PnL time-boosted searcher profits during the competition phase. Auction bids are denominated in ETH, while searcher profits are estimated using a 5 s markout. n denotes the number of auction rounds.

Winner	Pearson		Spearman		n
	Top Bid	Paid Bid	Top Bid	Paid Bid	
Wintermute	0.321	0.348	0.449	0.458	108,489
Selini	0.115	0.109	0.337	0.305	93,246

4 Empirical Results

We now present the empirical results. We first evaluate auction effectiveness during the pre-transition competition phase, before turning to the emergence of Kairos-mediated JIT resale and its effects on bidding behavior, auction revenue, and surplus allocation.

4.1 Auction Effectiveness in the Competition Phase

We first assess the effectiveness of the AOT auction during the competition phase, before the Kairos transition. To do so, we measure the relationship between auction bids and estimated searcher PnL at the round level. Table 1 shows that bids are noisy predictors of arbitrage profits. During the competition phase, the Pearson correlations between per-round profits and both the top bid and the paid bid are statistically significant but economically modest. Spearman correlations are higher for both searchers, suggesting that bidders could directionally identify higher-value intervals, but had limited ability to predict the magnitude of realized value.

One way to interpret this pattern is through the relationship between arbitrage profits and return variance. Return variance is a moderately informative predictor of estimated arbitrage profits for the relevant trading pairs, but the fit is substantially stronger over longer horizons than over the one-minute intervals used in the Timeboost auction. Consistent with this, both the top bid and the paid bid are noisy predictors of short-run markout profits, but become more informative when profits are aggregated over longer forward windows. This suggests that bidders are not simply unable to identify valuable market conditions; rather, the main difficulty is pricing the realized value of a specific one-minute fast-lane interval ahead of time. This pattern is summarized in Table 2.

Consistent with this interpretation, auction bids exhibit strong and slowly decaying autocorrelation during the competition phase, as shown in Figure 2. This implies that bids reflect persistent market conditions rather than only round-specific realized opportunities. By contrast, per-round markout PnL is much less persistent. The all-round markout series exhibits little autocorrelation beyond the first two lags, while positive-markout rounds show only limited short-run clustering. This difference is consistent with the idea that bidders can track broader market conditions, but face substantial uncertainty about the realized value of any individual one-minute interval.

The intra-round timing distribution motivates a shorter-window analysis. As shown in Figure 3, time-boosted CEX–DEX transactions are not uniformly distributed within the round. Transaction activity initially declines and then exhibits a pronounced spike around second 10 of the Timeboost interval, which corresponds approximately to the beginning of the next wall-clock minute. Because we do not observe the searchers' trading logic or off-chain opportunity signals, we cannot identify the cause of this spike directly.

Table 2 Bid–PnL correlations at multiple horizons during the competition phase before the Kairos transition. Correlations are computed over forward windows of length W, where one auction round is approximately one minute. For each window, the bid variable is the mean top or paid bid over the window, and PnL is the total time-boosted PnL over the same window. The sample contains $n = 201{,}737$ round-level observations.

Bid type	5 min	10 min	15 min	30 min	1 hour
Panel A: Spearman correlations					
Top bid	+0.681	+0.743	+0.768	+0.794	+0.804
Paid bid	+0.650	+0.714	+0.742	+0.774	+0.792
Panel B: Pearson correlations					
Top bid	+0.195	+0.245	+0.278	+0.342	+0.401
Paid bid	+0.178	+0.215	+0.236	+0.316	+0.378

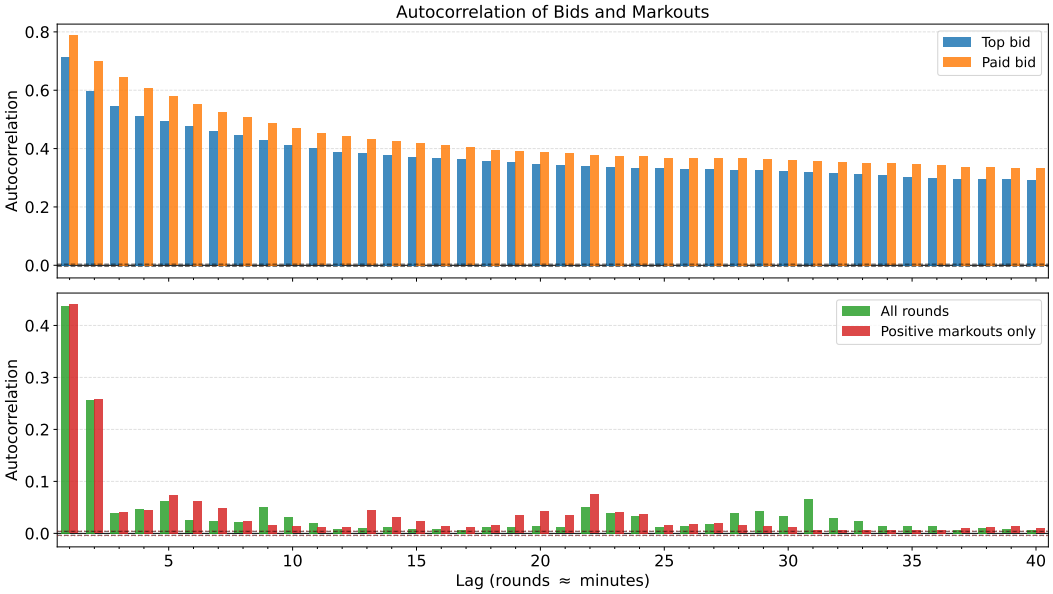


Figure 2 Autocorrelation functions of auction bids and per-round markout PnL during the competition phase before the Kairos transition. *Top:* Autocorrelation of the top and paid bids up to 40 lags, where one lag corresponds approximately to one auction round, or one minute. *Bottom:* Autocorrelation of per-round time-boosted PnL for all rounds and for the positive-markout subsample. Dashed and dotted lines indicate 95% confidence bands.

One possible interpretation is clock-time scheduling. Hasbrouck and Saar [12] document periodicities in Nasdaq message activity associated with automated strategies that access the market at fixed wall-clock intervals. The alignment of the spike with the minute boundary is consistent with a similar mechanism. However, it could also reflect periodic updates on external venues, protocol-specific timing effects, or common reactions by the dominant searchers to the same market signals. We therefore interpret the timing pattern as suggestive rather than causal.

Because activity is concentrated later in the interval, the first seconds of each round contribute disproportionately little to total realized PnL, which is relevant when interpreting correlations measured over shorter markout windows. We therefore repeat the round-level correlation exercise using shorter markout windows. Instead of measuring PnL over the full

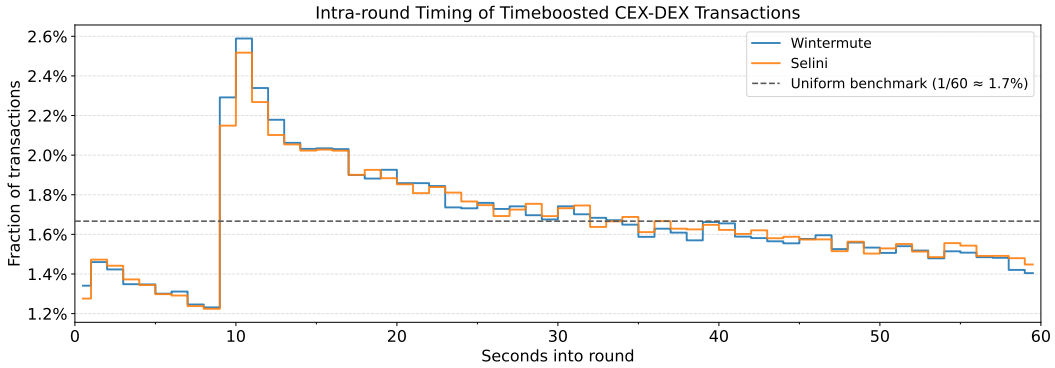


Figure 3 Intra-round timing distribution of time-boosted CEX–DEX transactions by Wintermute and Selini during the competition phase. Each line represents the fraction of transactions submitted in a given second of the one-minute Timeboost round. Dashed line indicates the uniform distribution benchmark.

Table 3 Correlations between auction bids and positive-PnL time-boosted markouts over shorter horizons during the competition phase before the Kairos transition. Markouts are computed over the first 15 and 30 seconds of the one-minute Timeboost interval. n denotes the number of auction rounds.

Winner	15 seconds		30 seconds		n
	Top bid	Paid bid	Top bid	Paid bid	
Panel A: Pearson correlations					
Wintermute	+0.240	+0.276	+0.282	+0.314	116,679
Selini	+0.087	+0.098	+0.101	+0.110	133,774
Panel B: Spearman correlations					
Wintermute	+0.385	+0.386	+0.421	+0.424	116,679
Selini	+0.292	+0.277	+0.315	+0.302	133,774

one-minute Timeboost interval, we compute markouts over the first 15 and 30 seconds of the interval. This test captures competing forces. On the one hand, bidders may be better informed about more immediate opportunities, which could increase the correlation between bids and realized PnL. On the other hand, shorter windows contain less aggregation, so realized profits are likely to be noisier than over the full minute. Moreover, because activity is concentrated later in the round, shorter windows also mechanically capture less of the realized opportunity, which works against finding strong correlations. The results in Table 3 support the latter interpretations: correlations remain positive but economically modest, and are generally weaker than for longer aggregation windows.

4.2 Bid Coordination and Just-in-Time Resale

In this section, we analyze the transition from direct competition in the primary Timeboost auction to Kairos-mediated fast-lane access. We focus on how this shift reshapes bidding behavior, primary auction revenue, and the distribution of surplus.

Table 4 Share of auction rounds won by each entity per period.

Entity	Competition	Kairos	Res. Price Adapt.	Steady State	Overall
Wintermute	45.0% (7,682/17,065)	0.0% (1/8,593)	72.7% (141/194)	11.3% (2,478/21,837)	21.6% (10,302/47,689)
Selini	53.4% (9,109/17,065)	0.2% (18/8,593)	3.6% (7/194)	9.8% (2,134/21,837)	23.6% (11,268/47,689)
Kairos	1.4% (244/17,065)	99.6% (8,559/8,593)	23.7% (46/194)	78.4% (17,119/21,837)	54.5% (25,968/47,689)

4.2.1 Timeline

Within our study period, we identify three key on-chain events that delineate the observed market dynamics:

1. **2026-02-12 20:31:51 UTC** – Wintermute and Selini cease competitive bidding in the primary Timeboost auction and begin sourcing fast-lane access through Kairos.
2. **2026-02-18 20:01:51 UTC** – Arbitrum raises the Timeboost reserve price from 0.001 ETH to 0.0075 ETH.
3. **2026-02-25 19:49:51 UTC** – Arbitrum lowers the reserve price back to 0.001 ETH.

These events divide the February 1–March 12, 2026 transition-analysis window into four sub-periods: *Competition* (pre-transition February), *Kairos* (February 12–18), *Reserve Price Adaptation* (February 18–25), and *Steady State* (after February 25). The label *Competition* refers specifically to the February segment of the broader pre-transition competition phase. We use this shorter pre-transition window because auction dynamics in February are representative of the full pre-transition period: the August 2025–February 2026 data exhibit a stable structure (see Figure 1).

4.2.2 Bidding Behavior

During the coordination phase, we observe 47,689 Timeboost auctions, of which Kairos (54.5%), Selini (23.6%), and Wintermute (21.6%) together account for 99.7% of all wins (see Table 4). In total, 117,146 bids were placed, with an average of 2.46 bids per round across 14 distinct bidding addresses. Selini accounts for the highest share of bids (38.7%), followed by Wintermute (32.7%) and Kairos (25.0%); together these three entities are responsible for 96.4% of all bids. We summarize auction participation and active periods for each bidding address in the appendix of the full version [26]. We interpret these as distinct on-chain bidding accounts and do not assume that unattributed addresses necessarily correspond to distinct economic entities.

Looking only at the number of bids per round might suggest that the competitiveness of the auction was not significantly affected by the adoption of Kairos. Table 5 shows that the number of rounds in which all three entities placed a bid is highest after this transition. However, examining the gap between the winning bid (top bid) and the paid bid reveals a clear decline in auction competitiveness.

Figure 4 shows that during the *Competition* phase, both the top bid and the paid bid move closely with volatility in ETH prices[9]. This relationship is confirmed by the correlation measures reported in the appendix of the full version [26], which show strong and statistically significant linear relationships between volatility and both paid and top bids. The absolute bid gap also increases with volatility, indicating that while both bids respond to more valuable trading opportunities, the top bid tends to rise more strongly. During this period, the median daily relative bid gap is 37.3%.

[9] We compute volatility as the standard deviation of log returns from consecutive one-second ETH/USDT mid-price observations.

Table 5 Share of auction rounds by bidder participation combination in each period. WM = Wintermute, Sel = Selini, Kai = Kairos.

Combination	Competition	Kairos	Res. Price Adapt.	Steady State	Overall
WM + Sel + Kai	1.5% (262/17,065)	10.9% (935/8,593)	1.5% (3/194)	85.3% (18,621/21,837)	41.6% (19,821/47,689)
WM + Sel	88.8% (15,161/17,065)	0.0% (0/8,593)	4.1% (8/194)	5.1% (1,118/21,837)	34.2% (16,287/47,689)
WM + Kai	0.0% (4/17,065)	0.2% (14/8,593)	10.3% (20/194)	6.7% (1,471/21,837)	3.2% (1,509/47,689)
Sel + Kai	0.2% (41/17,065)	87.1% (7,485/8,593)	0.5% (1/194)	1.2% (257/21,837)	16.3% (7,784/47,689)
Only WM	1.2% (198/17,065)	0.0% (0/8,593)	67.5% (131/194)	1.4% (314/21,837)	1.3% (643/47,689)
Only Sel	8.2% (1,399/17,065)	0.0% (4/8,593)	3.1% (6/194)	0.2% (35/21,837)	3.0% (1,444/47,689)
Only Kai	0.0% (0/17,065)	1.8% (155/8,593)	12.9% (25/194)	0.1% (20/21,837)	0.4% (200/47,689)
None	0.0% (0/17,065)	0.0% (0/8,593)	0.0% (0/194)	0.0% (1/21,837)	0.0% (1/47,689)

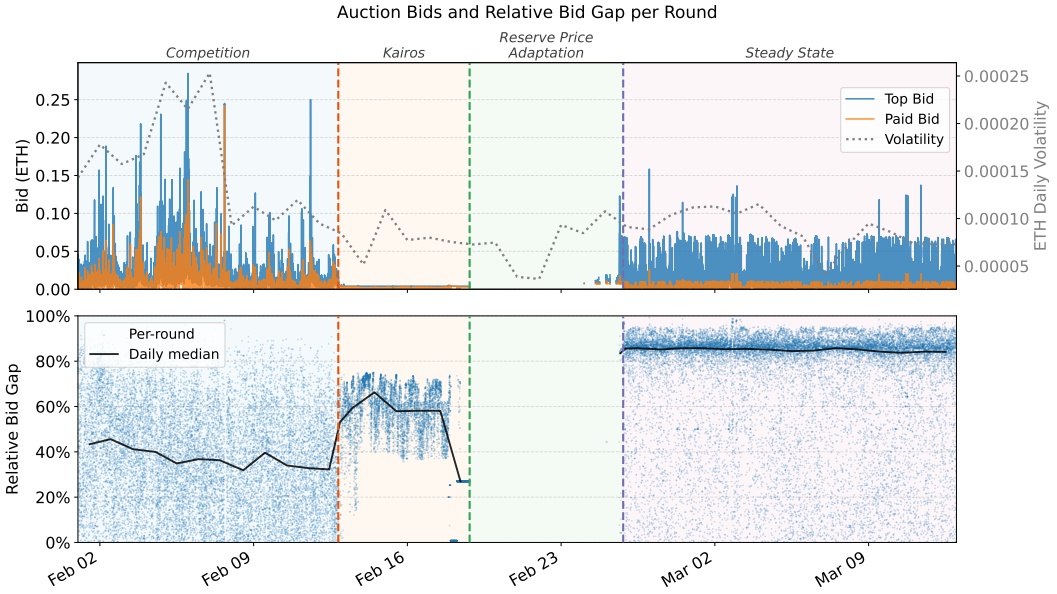


Figure 4 Timeboost auction bidding dynamics during the transition-analysis window. *Top:* Top bid and paid bid per auction round, measured in ETH. The dotted line shows daily ETH volatility on the right axis. *Bottom:* Relative bid gap between the top bid and the paid bid for each auction round, shown as points, together with the daily median, shown as a line. Breaks in the daily-median line indicate periods with no auction rounds receiving bids.

Once Wintermute and Selini retreat from the auction and begin sourcing fast-lane access through Kairos (the *Kairos* period), the median daily relative bid gap increases to 57.5%. At the same time, the paid bid only weakly tracks price volatility, while the top bid shows no statistically significant relationship. During this period, Kairos wins 99.6% of the rounds and predominantly submits a fixed bid of 0.004 ETH, while the paid price typically remains only marginally above the reserve price, usually submitted by Selini (see Table 5). As a result, the bid gap becomes negatively correlated with volatility, which can be explained by the paid price adjusting slightly with volatility while the top bid remains largely fixed. Consequently, auction pricing in this regime remains largely compressed despite changing market conditions.

One notable deviation during the *Kairos* period occurs in the final 24 hours before the reserve-price increase. At 20:00:51 UTC on February 17, the previously inactive and unattributed address `0x3141592601bf441700f150bef82db6040f4fe373` begins submitting several different bid amounts. After approximately two hours, it switches to a fixed bid

of 0.003970 ETH, only 0.75% below Kairos's predominant fixed bid of 0.004 ETH, and submits this amount in every round from 23:00 on February 17 until approximately 06:00 on February 18. Because Timeboost is a second-price auction, the address becomes the price-setting second-highest bidder in most of these rounds, sharply reducing the relative bid gap and increasing the price paid by Kairos.

From 06:00 until the reserve-price update approximately 14 hours later, the address instead submits a fixed bid of 0.002921 ETH, producing the roughly 27% relative-bid-gap plateau visible in Figure 4. The address wins only 13 rounds, approximately 1% of the rounds in which it participates, but sets the paid bid in 1,238 rounds, or 96% of its active rounds. In the remaining rounds, another address submits the second-highest bid. The address is listed in the appendix of the full version [26], but we find no public attribution for it.

During the *Reserve Price Adaptation* period, the reserve price increases from 0.001 ETH to 0.0075 ETH and bidding activity nearly disappears. Only 194 rounds receive at least one bid, the majority of which (131 rounds) involve Wintermute as the sole bidder. Among these active rounds, bidding activity is again positively correlated with price volatility, consistent with Wintermute deriving value from external price movements as a CEX–DEX searcher. The apparent increase in the daily median relative bid gap in Figure 4 should therefore not be interpreted as a gradual change in bidding behavior, as it is based on a small number of active rounds. In rounds with only one bidder, the paid price is determined by the reserve, so the relative bid gap reflects the difference between Wintermute's bid and the elevated reserve price. Bidding resumes more consistently only after the reserve price returns to 0.001 ETH.

Finally, the market enters what we refer to as the *Steady State* once Arbitrum lowers the reserve price back to its original value of 0.001 ETH. Despite having the same reserve price as the earlier *Kairos* period, bidding behavior differs substantially. All three entities submit bids in 85.3% of rounds, while Kairos wins 78.4% of rounds. Both the top bid and the paid bid again exhibit strong positive correlations with volatility, as does the absolute bid gap. However, the paid bid remains effectively capped in most rounds, as reflected in Figure 4. As a result, the top bid rises more strongly than the paid bid when trading opportunities become more valuable, and the median relative bid gap stabilizes at 85.2%. Thus, the paid bid amounts to only approximately 14.8% of the top bid, indicating that the primary auction captures only a limited share of the value reflected in submitted bids.

Unlike in the preceding *Kairos* period, Kairos no longer submits a predominantly fixed bid or wins nearly every auction. Across the 21,837 rounds with at least one bid, Kairos participates in 93.3% and wins 84.0%. The median relative gap between Kairos's winning bid and the second-highest bid is approximately 86.0%, whereas it falls to 32.0% in rounds that Kairos loses. The difference is even more pronounced in absolute terms: the median gap is approximately 0.015 ETH when Kairos wins, compared with 0.001 ETH when another bidder wins.

This asymmetry is also reflected in the bid distributions shown in Figure 5. During the *Competition* period, median bids are relatively close across Wintermute, Selini, and Kairos (0.0075, 0.0062, and 0.0069 ETH, respectively). In the *Steady State*, they diverge substantially: Kairos's median bid rises to 0.0182 ETH, whereas the corresponding medians for Wintermute and Selini fall to 0.0027 and 0.0021 ETH. The same pattern appears in the upper tail of the distributions (p99: Wintermute 0.0096 ETH, Selini 0.0101 ETH, and Kairos 0.0703 ETH). A more detailed distribution is reported in the appendix of the full version [26]. These patterns show that the renewed participation of Wintermute and Selini does not constitute a return to the competitive bidding observed before the adoption of Kairos. Kairos bids substantially more aggressively, while the other two searchers generally submit bids close to the reserve.

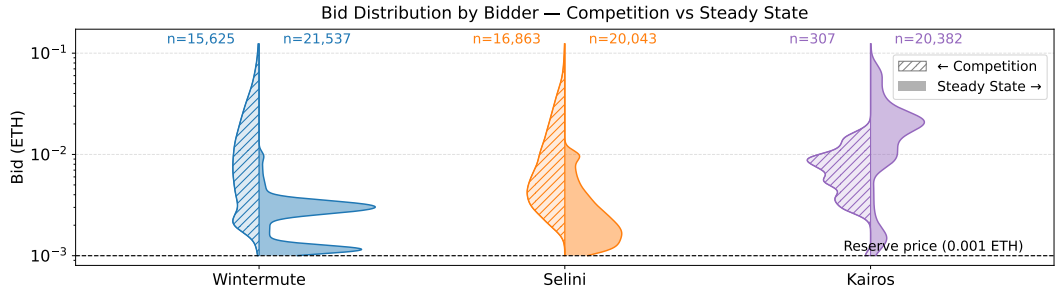


Figure 5 Bid distributions for Wintermute, Selini, and Kairos. The left half of each violin corresponds to the *Competition* period, while the right half corresponds to the *Steady State*. The dashed horizontal line indicates the reserve price (0.001 ETH).

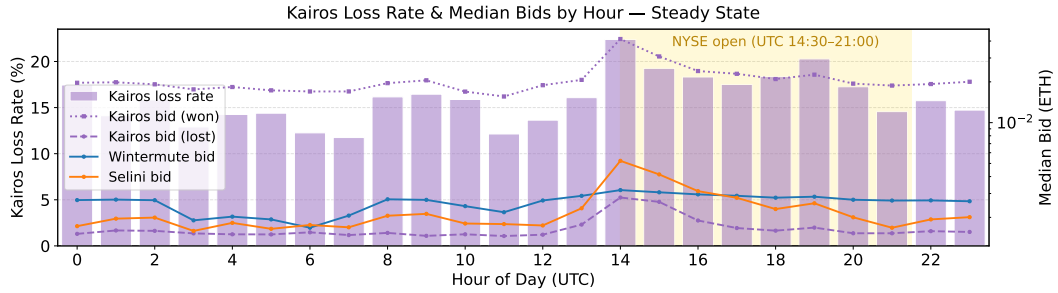


Figure 6 Kairos loss rate and median bids by hour of day in the *Steady State* period. The shaded region indicates UTC hour bins overlapping the regular NYSE session (14:30–21:00 UTC during EST).

To better understand when Kairos loses in the *Steady State*, we examine the temporal distribution of its losses. As shown in Figure 6, Kairos loses more frequently during U.S. market hours, particularly around the NYSE opening: its loss rate is 19.0% during NYSE hours, compared with 14.5% outside those hours. Selini, in particular, submits somewhat higher bids during these periods. Nevertheless, Kairos's median bid in rounds it loses remains substantially below its median winning bid. Heightened market activity therefore explains only part of the variation in Kairos's losses. More generally, losses tend to occur when Kairos submits a relatively low bid or does not participate, rather than when Wintermute or Selini submit bids comparable to Kairos's usual winning bids.

One possible interpretation is that the temporary reserve-price increase altered participants' expectations about future protocol responses, making a complete withdrawal from the primary auction less attractive after the reserve returned to 0.001 ETH. This could help explain why all three entities maintain visible bidding activity in the *Steady State*, while Kairos submits higher and more variable bids to preserve its access to the fast-lane. Because participants' expectations and off-chain arrangements are unobserved, we cannot identify the cause of this shift directly. We therefore interpret the *Steady State* as a modified coordination regime with limited residual competition, rather than a return to the earlier competitive environment.

4.2.3 Searcher Activity

We next examine how the changing auction dynamics affect searcher activity and the use of the Timeboost fast-lane versus the regular-lane. The top-left and top-right panels in Figure 7 show the number of time-boosted and regular CEX–DEX transactions submitted over time by Wintermute and Selini, respectively. The bottom-left panel reports daily PnL, while the bottom-right panel shows the average PnL per trade per day for time-boosted and regular transactions. Consistent with our baseline PnL specification, we restrict attention to trades with positive estimated PnL. Results are summarized in the appendix of the full version [26].

Before the adoption of Kairos, in the *Competition* phase, searchers submit CEX–DEX transactions through both the Timeboost fast-lane and the regular-lane when they do not control the fast-lane. Their usage patterns differ meaningfully: Wintermute earns more total PnL from time-boosted trades and submits relatively more of them, whereas Selini earns more total PnL from regular-lane trades and also achieves higher average profit per trade there. More broadly, although Wintermute trades more frequently and earns higher total profits overall, Selini tends to realize higher average profit per trade. These differences remain largely stable across auction regimes and suggest that the two searchers follow somewhat different trading models. At the same time, there are periods in which average profit per trade is similar across lanes, and sometimes even higher in the regular-lane for both searchers, indicating that Timeboost is not reserved only for the most profitable opportunities but also helps secure smaller trades that might otherwise be lost to latency competition. For both searchers, PnL from both time-boosted and regular-lane trades remains positively correlated with volatility (see the appendix of the full version [26]), consistent with the idea that higher volatility creates more opportunities to exploit stale DEX prices relative to rapidly moving CEX prices.

Once Wintermute and Selini begin using Kairos, both almost completely stop submitting transactions through the regular-lane, as they can now access the fast-lane via Kairos. This indicates that Kairos effectively substitutes for direct participation in the Timeboost auction, allowing searchers to retain fast-lane access without bidding competitively in the primary auction. During this *Kairos* phase, the PnL from time-boosted trades remains strongly correlated with volatility for both searchers, whereas the relationship is considerably weaker for regular-lane trades, suggesting a clear shift in how the two submission lanes are used.

During the *Reserve Price Adaptation* phase, when bidding activity was minimal and resumed only toward the end, both searchers mainly continued operating through the regular transaction lane. Their average PnL remains similar to the *Kairos* period, but the volatility relationships appear to reverse: regular-lane transactions now show a strong correlation with volatility for both searchers, while the much less frequent time-boosted trades exhibit a weaker relationship.

Finally, once the reserve price returns to its original level, both searchers move back to the fast-lane while submitting relatively few regular-lane transactions. In this *Steady State* period, both execute significantly more time-boosted trades relative to regular-lane trades than in the *Competition* period, reflecting improved access to the fast-lane through Kairos.

Overall, the adoption of Kairos allows searchers to access the fast-lane almost continuously, largely eliminating the need to rely on the slower regular-lane while time-boosted activity continues to track price movements closely. However, consistent with the weaker bidding competition documented above, this increased reliance on the fast-lane does not translate into a proportional increase in primary-auction revenue.

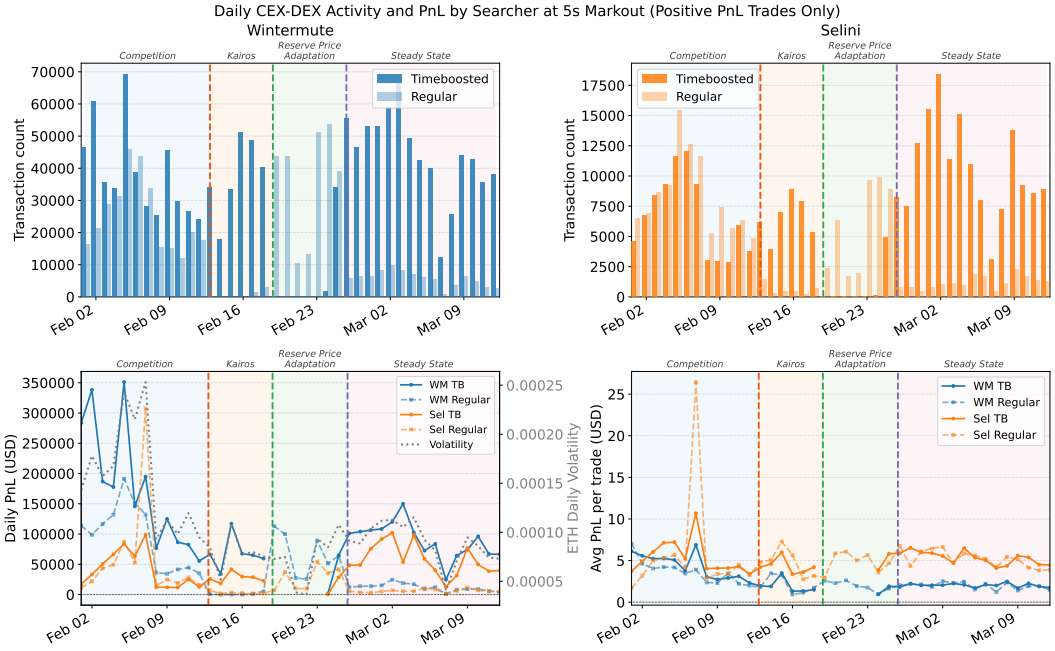


Figure 7 CEX–DEX arbitrage activity by Wintermute and Selini during the transition-analysis window. *Top-left:* time-boosted and regular transaction count by Wintermute. *Top-right:* time-boosted and regular transaction count by Selini. *Bottom-left:* daily PnL from time-boosted and regular transactions for Wintermute and Selini. The dotted line shows daily ETH volatility (right axis). *Bottom-right:* average PnL per trade for time-boosted and regular transactions for Wintermute and Selini.

4.2.4 Surplus

Lastly, we examine how the changing auction dynamics affect the total surplus generated and its distribution across the involved parties. We define net surplus as

$$\text{Net Surplus} := \sum_{\text{Wintermute, Selini}} \text{PnL} - \text{transaction fees} - \text{auction bids paid},$$

where transaction fees and auction bids paid together represent revenue captured by Arbitrum, while total searcher PnL captures the value generated by CEX–DEX arbitrage opportunities and ultimately shared among participants in the Timeboost ecosystem.

Figure 8 presents the evolution of total surplus over time (top panel) and the share of searcher PnL captured by Arbitrum through auction bids and transaction fees (bottom panel). Throughout the study period, searcher PnL is strongly positively correlated with daily ETH volatility ($r = +0.96$, $p < 0.001$), with visible spikes during the *Competition* phase when volatility was highest. Importantly, total searcher PnL remains of similar magnitude across regimes, suggesting that the observed changes in primary-auction competition mainly reflect a redistribution of surplus rather than an increase in total value generated.

We find that bids paid correspond on average to 7.4% of total searcher PnL during the *Competition* phase[10], but this declines following the adoption of the Kairos secondary market (*Kairos*: 5.8%, *Reserve Price Adaptation*: 0.4%, *Steady State*: 6.0%). This decline is even

[10] This share is likely understated, as we restrict attention to positive-PnL trades. Including loss-making trades would reduce net searcher PnL and thus mechanically increase the captured share.

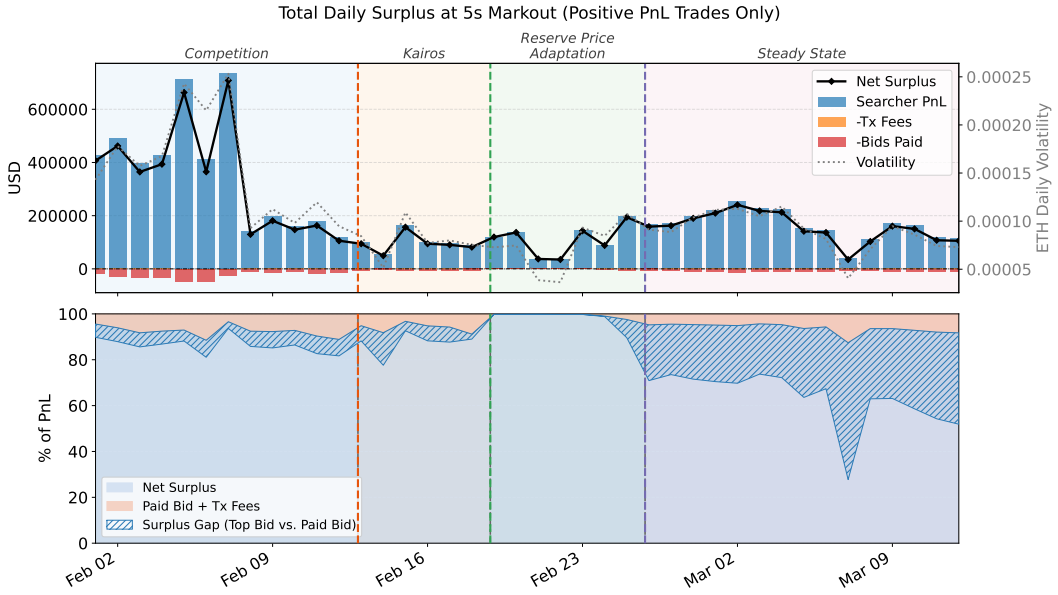


Figure 8 Surplus dynamics during the February 1–March 12, 2026 transition-analysis window. *Top:* daily surplus decomposition into searcher PnL, transaction fees, and Timeboost auction bids paid. The black solid line shows the net surplus, while dotted line shows daily ETH volatility (right axis). *Bottom:* share of searcher PnL retained as net surplus versus captured by Arbitrum through transaction fees and paid auction bids. The hatched area shows the difference between the share of surplus reflected in the top bid and the share captured through the paid bid.

more pronounced when focusing only on time-boosted surplus and excluding regular-lane transactions (*Competition*: 12.5%, *Steady State*: 6.7%). This pattern is consistent with increasing reliance on the fast-lane via Kairos together with weaker direct competition in the primary auction.

Although bids in the Timeboost auction need not directly reveal true valuations because of the common-value component and strategic shading, stronger competition in the *Competition* period likely allowed a larger share of surplus to be captured through the auction than in the *Steady State*, where Kairos is often the only competitive participant with a high value for winning the lane. Consistent with this interpretation, the top bid corresponds to a substantially larger share of surplus in the final regime (*Competition*: 13.6%, *Steady State*: 36.3%), while the gap between the surplus reflected in the top bid and the surplus captured through the paid bid also widens markedly (bottom panel of Figure 8). The key change, therefore, is not that bidding ceases to respond to valuable trading opportunities, but that weaker competition causes a smaller fraction of the underlying value to be captured through the paid bid. Whether this reflects more aggressive shading, coordination, or lower standalone value from controlling the lane for bidders other than Kairos, the implication is the same: weaker competition increases the disconnect between value and revenue.

We now examine payments to Kairos to better understand how surplus may be distributed outside the primary auction. Figure 9 compares the total daily bids paid by Kairos in the Timeboost auction with the on-chain payments it receives from transactions using its service. While Kairos's auction payments increase substantially once it becomes the dominant auction participant, the observed on-chain payments it receives do not increase proportionally.

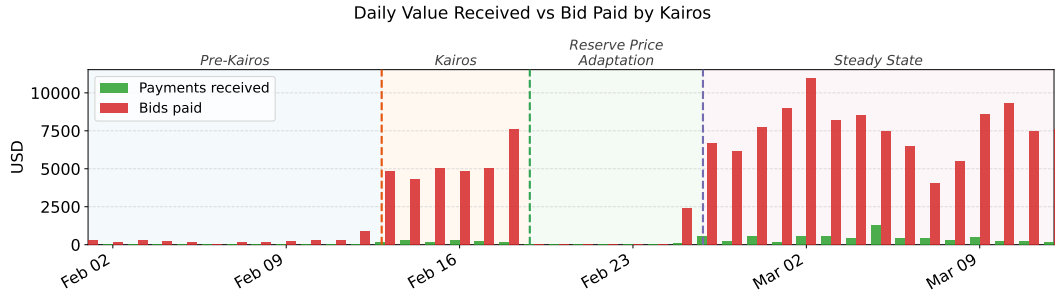


Figure 9 Kairos primary auction payments and resale revenue during the February 1–March 12, 2026 transition-analysis window. The figure compares the daily Timeboost auction bids paid by Kairos to the on-chain payments Kairos receives from transactions using its service.

Over the study period, Kairos pays a total of 151,302 USD in auction bids, while the on-chain payments we observe sum to only 8,001 USD. The resulting gap of more than 143,000 USD suggests that a substantial portion of the payments to Kairos likely occurs through alternative mechanisms that are not visible on-chain, such as its subscription model for CEX–DEX arbitrage searchers.

During periods when Kairos controls the fast-lane, Wintermute and Selini submit a combined total of 1,004,876 time-boosted CEX–DEX transactions with positive PnL, generating approximately 2,541,351 USD. However, none of the identified on-chain payments to Kairos originate from their transactions, suggesting that these entities primarily rely on Kairos's off-chain payment structure.

Taken together, these findings are consistent with the evidence above that, once Kairos becomes dominant, a smaller share of the value generated around Timeboost is captured through the primary auction. However, because payments to Kairos are only partially observable on-chain, the ultimate division of the remaining surplus between the intermediary and the searchers remains unclear. Hence, we cannot establish the extent using Kairos benefits Wintermute and Selini as the final surplus allocation is unobservable on-chain.

5 Concluding Discussion

We conclude by discussing what our empirical results imply for value extraction in AOT auctions. We first consider the effectiveness of AOT auctions when access is allocated through the primary auction alone, then examine how JIT resale changes revenue capture and surplus allocation. Finally, we discuss broader implications for Ethereum.

5.1 Value Extraction Through Ahead-of-Time Auctions

Analyzing Timeboost auction data in the competition phase shows that estimating immediate future arbitrage value is difficult. Bids are reasonably informative about longer-horizon average arbitrage profits, but they are noisy predictors of one-minute or shorter-horizon profits. This finding casts doubt on whether AOT auctions are effective MEV extraction mechanisms for rollups or other blockchains, even when no JIT resale market forms.

In a pure common-value auction with risk-neutral bidders, noisy ex-ante signals do not necessarily imply low expected seller revenue: bidders should condition on the winner's curse and shade their bids relative to their private signals, but competition can still transfer a substantial share of expected value to the seller. In practice, however, the risk-neutrality

assumption might not hold perfectly. Bidders may be risk averse, face capital or inventory constraints, or differ in their private ability to exploit opportunities. Such private-value components weaken the link between common arbitrage value and bids. They also imply that selling ordering rights AOT may fail to allocate the fast-lane to the bidder with the highest realized value for it.

5.2 Value Extraction if There is a Just-in-Time Resale Market

The adoption of the Kairos resale market effectively dampens competition in Arbitrum's Timeboost auction, consistent with the weakened-competition channel being the dominant mechanism – though we find some limited supporting evidence for un-bundling and information effects as well. As a result, the protocol appears to capture a substantially smaller share of the value generated around Timeboost, while a larger share remains outside the primary auction. However, the exact distribution of this additional surplus between the searchers and Kairos remains unclear, as the payments associated with the resale mechanism are only partially observable on-chain.

The success of the Kairos resale market is also reflected in the increasing volume of time-boosted transactions once Kairos wins most auction rounds. However, this effect is not clearly visible in PnL, which is a noisier signal. As a result, the unbundling effect appears to be second-order relative to the impact of weaker competition on auction outcomes. This channel may become more prominent if Kairos attracts additional searchers in the future, particularly those employing more diverse strategies such as atomic arbitrage.

To improve revenue capture, Arbitrum could pursue several directions. Beyond adopting an entirely different mechanism or market structure, the current auction format itself could be modified to better align payments with the value reflected in bids.

First, switching to a first-price auction could make anti-competitive bidding more difficult [17] and may reduce the gap between value and payments. Second, the auction could respond more effectively to changing demand by adjusting the reserve price dynamically rather than relying on a fixed reserve. In particular, incorporating external price volatility – which appears to be a key driver of searcher activity and auction values – into the reserve-price design could push bids closer to bidders' willingness to pay.

We illustrate this with a simple volatility-based reserve-price rule. Specifically, we set the reserve price equal to $c\sigma_t^2$, where c is a constant calibrated using bid data from the *Competition* period and σ_t denotes the standard deviation of log returns, $\log(p_{i+1}/p_i)$, computed over a window of length t before the auction ends. Using this rule, we recover at least 80% of the revenue benchmarked against the sum of top bids during the period from August 2025 to January 2026. Figure 10 shows the performance of this dynamic reserve-price rule across different choices of t and c.

5.3 Lessons for Ethereum L1

The lessons from Timeboost's AOT auction may also be relevant for Ethereum L1, subject to two caveats. First, Timeboost is mainly used for CEX–DEX arbitrage, which captures only part of Ethereum MEV, albeit an important and highly latency-sensitive part. A rough analogue on Ethereum would be an AOT auction for top-of-block rights over several consecutive blocks. Second, while revenue capture through Timeboost is a primary concern for Arbitrum, protocol-level MEV capture on Ethereum – for example through MEV-burn-type designs [6] – is arguably less central than preserving broader system properties.

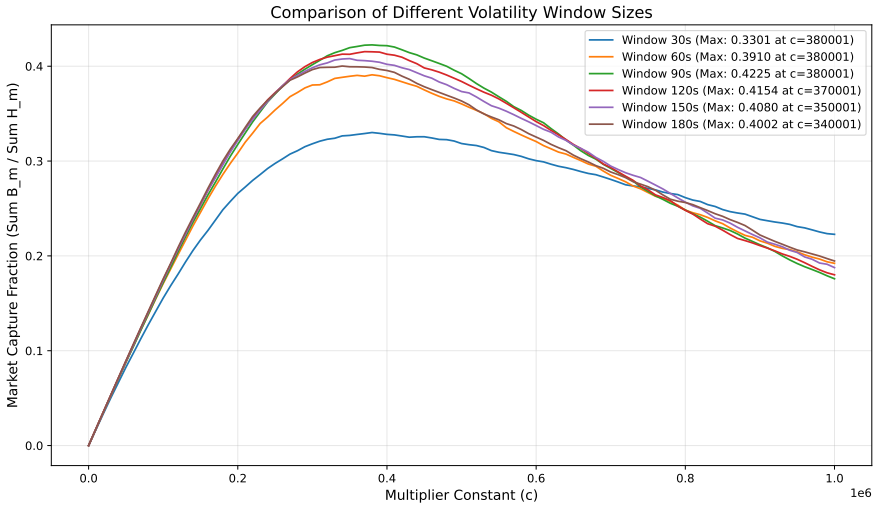


Figure 10 Performance of the dynamic reserve-price rule for different choices of t and c, measured relative to the sum of top bids.

With these caveats in mind, two lessons appear particularly relevant. First, on Timeboost the population of dominant searchers is narrow, and the value they extract from the fast-lane is largely concentrated in the same type of opportunity: CEX–DEX arbitrage on a few liquid pairs. On Ethereum L1, MEV opportunities are more heterogeneous and accrue to different searcher classes at different times. An AOT auction allocates ordering rights before this realization is known and is therefore more likely to misallocate them relative to a JIT mechanism. We would expect this allocative inefficiency to be larger on L1, even before any secondary market emerges.

Second, limited competition among dominant searchers can create favorable conditions for secondary markets that move value outside the primary auction. Ethereum's searcher and builder markets remain more competitive than Arbitrum's and are not currently as concentrated as the duopolistic structure we observe here. Still, the searcher market on Ethereum has become more concentrated over time, and similar resale dynamics could plausibly emerge if competition weakens substantially.

More broadly, the Arbitrum case suggests that AOT allocation mechanisms may be particularly vulnerable to secondary-market intermediation once competition among dominant participants becomes weak. This vulnerability may itself be amplified by AOT design. In a JIT auction, valuations are likely to be more dispersed across auctions, allowing participants who are usually not competitive to occasionally outbid dominant players when they identify especially valuable opportunities. Such episodic entry can help preserve some competitive pressure. When ordering rights are sold ahead of time, this source of occasional competitiveness is weaker, potentially making uncompetitive equilibria even easier to sustain.

References

1 Arbitrum Foundation. How to use timeboost: How to view historical bid data. `https://docs.arbitrum.io/how-arbitrum-works/timeboost/how-to-use-timeboost#how-to-view-historical-bid-data`, 2026. Accessed: 2026-03-11.

2 Arbitrum Foundation. Timeboost: A gentle introduction. https://docs.arbitrum.io/how-arbitrum-works/timeboost/gentle-introduction, 2026. Accessed: 2026-03-11.

3 Binance. Binance market data. https://data.binance.vision, 2026. Accessed: 2026-03-11.

4 Agostino Capponi and Brian Zhu. Does timeboost reduce mev-related spam? theory and evidence from layer-2 transactions. *CoRR*, abs/2512.10094, 2025. doi:10.48550/arXiv.2512.10094.

5 Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 910–927. IEEE, 2020. doi:10.1109/SP40000.2020.00040.

6 Justin Drake. Mev burn—a simple design. https://ethresear.ch/t/mev-burn-a-simple-design/15590, 2023.

7 Dune Analytics. Dune: Blockchain data analytics platform. https://dune.com/, 2026. Accessed: 2026-03-11.

8 Flashbots. What is mev-boost? https://docs.flashbots.net/flashbots-mev-boost/introduction, 2026. Accessed: 2026-03-18.

9 Robin Fritsch, Maria Inês Silva, Akaki Mamageishvili, Benjamin Livshits, and Edward W. Felten. MEV capture through time-advantaged arbitrage. *CoRR*, abs/2410.10797, 2024. doi:10.48550/arXiv.2410.10797.

10 Gattaca. Kairos timeboost documentation. https://docs.kairos-timeboost.xyz/, 2026. Accessed: 2026-03-11.

11 Gattaca. Parallel kairos lanes. https://docs.kairos-timeboost.xyz/parallel-kairos-lanes, 2026. Accessed: 2026-03-12.

12 Joel Hasbrouck and Gideon Saar. Low-latency trading. doi:10.2139/ssrn.1695460.

13 Boyi Li and Weixuan Xia. Crypto inverse-power options and fractional stochastic volatility. *Quantitative Finance*, 25(7):1073–1099, 2025. doi:10.1080/14697688.2025.2518252.

14 Julian Ma. Block vs. Slot Auction PBS. *Paragraph*, 2022. Available at https://paragraph.com/@julianma/block-vs-slot-auction-pbs.

15 Julian Ma. Trusted advantage in slot auction epbs. https://ethresear.ch/t/trusted-advantage-in-slot-auction-epbs/20456, 2024.

16 Akaki Mamageishvili, Mahimna Kelkar, Jan Christoph Schlegel, and Edward W. Felten. Buying Time: Latency Racing vs. Bidding for Transaction Ordering. In Joseph Bonneau and S. Matthew Weinberg, editors, *5th Conference on Advances in Financial Technologies (AFT 2023)*, volume 282 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:22, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.AFT.2023.23.

17 Robert C. Marshall and Leslie M. Marx. Bidder collusion. *Journal of Economic Theory*, 133(1):374–402, March 2007. URL: https://ideas.repec.org/a/eee/jetheo/v133y2007i1p374-402.html.

18 Johnnatan Messias and Christof Ferreira Torres. The express lane to spam and centralization: An empirical analysis of arbitrum's timeboost. *CoRR*, 2025. doi:10.48550/arXiv.2509.22143.

19 Paul R. Milgrom and Robert J. Weber. A theory of auctions and competitive bidding. *Econometrica*, 50, 1982.

20 Jason Milionis, Ciamac C Moallemi, Tim Roughgarden, and Anthony Lee Zhang. Automated market making and loss-versus-rebalancing. *arXiv preprint*, 2022. arXiv:2208.06046.

21 Ciamac C. Moallemi, Mallesh M. Pai, and Dan Robinson. Latency advantages in common-value auctions. *CoRR*, abs/2504.02077, 2025. arXiv:2504.02077.

22 Mallesh Pai and Max Resnick. Centralization in attester-proposer separation. https://arxiv.org/abs/2408.03116, 2024. doi:10.48550/arXiv.2408.03116.

23 Christoph Schlegel. Transaction ordering auctions. In *International Conference on Financial Cryptography and Data Security*, pages 91–104. Springer, 2024. doi:10.1007/978-3-031-69231-4_7.

24 Caspar Schwarz-Schilling, Fahad Saleh, Thomas Thiery, Jennifer Pan, Nihar Shah, and Barnabé Monnot. Time is money: Strategic timing games in proof-of-stake protocols. In Joseph Bonneau and S. Matthew Weinberg, editors, *5th Conference on Advances in Financial Technologies, AFT 2023, October 23-25, 2023, Princeton, NJ, USA*, volume 282 of *LIPIcs*, pages 30:1–30:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.AFT.2023.30.

25 Fei Wu, Danning Sui, Thomas Thiery, and Mallesh Pai. Measuring CEX-DEX extracted value and searcher profitability: The darkest of the MEV dark forest. *CoRR*, abs/2507.13023, 2025. doi:10.48550/arXiv.2507.13023.

26 Burak Öz, Akaki Mamageishvili, Christoph Schlegel, and Ali Taslimi. Just-in-time resale in an ahead-of-time auction for faster execution, 2026. arXiv:2603.20175.

27 Burak Öz, Christof Ferreira Torres, Christoph Schlegel, Bruno Mazorra, Jonas Gebele, Filip Rezabek, and Florian Matthes. Cross-chain arbitrage: The next frontier of mev in decentralized finance. https://arxiv.org/abs/2501.17335, 2025. arXiv:2501.17335.

Towards Decentralized Searcher Competition in MEV Markets

Roozbeh Sarenche
COSIC, KU Leuven, Belgium

Yunwen Liu
COSIC, KU Leuven, Belgium

Abstract

Centralization in maximal extractable value (MEV) markets is a significant concern for blockchain systems, as persistent concentration of economic power can weaken competition, reduce openness, and undermine the decentralization goals of permissionless protocols. While much of the existing analysis has focused on builders, validators, and block-building markets, this paper studies centralization from the perspective of searcher competition. We develop a heterogeneous model in which searchers differ in opportunity coverage and execution efficiency, and we analyze how auction design affects fairness, decentralization, and security among searchers competing for the same MEV opportunity.

To evaluate searcher competition, we introduce two metrics: a Shapley-weighted Jain fairness index, which measures whether rewards are proportional to searchers' marginal contributions, and an expected-reward Herfindahl–Hirschman Index (HHI), which measures concentration in long-run searcher rewards. Using these metrics, we first analyze the standard first-price, winner-take-all auction as a benchmark. Our analysis shows that, under searcher heterogeneity, first-price competition can reward rank dominance rather than marginal contribution, leading to concentrated rewards and weaker contribution-adjusted fairness.

Motivated by these limitations, we propose an entry-filtered Shapley-capped auction mechanism that distributes searcher rewards more fairly and broadly among admitted high-quality submissions. Designing such a mechanism in a permissionless blockchain environment is challenging: searchers may create Sybil identities by submitting copied or degraded versions of the same execution strategy, and validators may collude with searchers to increase joint payoff. We address these concerns through Bayesian security constraints for copied-code Sybil deviations and validator–searcher coalition deviations, and show that suitable parameter choices make the Shapley-capped mechanism secure against both.

We complement the theoretical analysis with numerical benchmarks and Ethereum on-chain data. The theoretical benchmarks show that the proposed mechanism is especially effective in centralized opportunity classes, where first-price competition concentrates rewards among a small set of searchers. They further indicate that the proposed mechanism is a potential improvement over first-price competition in empirically motivated opportunity types.

2012 ACM Subject Classification Theory of computation → Algorithmic game theory and mechanism design; Security and privacy → Economics of security and privacy

Keywords and phrases Maximal extractable value, MEV, searcher competition, mechanism design, Sybil resistance, Shapley value, reward decentralization

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.21

Related Version *Full Version*: `https://arxiv.org/abs/2608.05011` [20]

Supplementary Material *Software (Source Code)*: `https://gitlab.esat.kuleuven.be/mev/searcher_game`

Funding *Roozbeh Sarenche*: Supported by the Flemish Government through the Cybersecurity Research Program Flanders with grant number: VOEWICS02.
Yunwen Liu: Partially supported by KU Leuven STG/25/039 and CTVG R&D grant (QJW-410118-LIU-CTVG).



© Roozbeh Sarenche and Yunwen Liu;
licensed under Creative Commons License CC-BY 4.0



1 Introduction

In Ethereum's Maximal Extractable Value (MEV) supply chain, *searchers* are specialized actors who identify profitable execution opportunities and submit transactions or bundles to capture them [4, 5]. Their submissions enter auction-like competition for block inclusion, where searchers bid part of the generated value to block producers – builders and validators.

Prior work suggests that searcher-level MEV extraction is far from a symmetric market. Searchers differ in opportunity access, latency, information, capital, and execution quality, while specialized bot behavior and heterogeneous competitive pressure further amplify these differences [10, 26]. For atomic MEV strategies, including sandwiching and decentralized-exchange arbitrage, early studies already documented sophisticated bot specialization and intense competition [4]. More recent evidence shows that concentration can be severe: in centralized–decentralized exchange arbitrage, Heimbach et al. [9] find that eleven searchers account for more than 80% of identified arbitrage volume, with the two largest searchers alone accounting for 49.2%.

These observations suggest that searcher markets can become highly skewed in the presence of players with specialized infrastructure, private information, capital, or privileged access. This skewness, together with current searcher-auction design, has consequences beyond searcher profits and can become a bottleneck for blockchain decentralization. Persistent reward concentration weakens open competition and raises effective barriers to entry. This concern is amplified on high-throughput chains where MEV competition often takes the form of large-scale speculative transaction submission [25]. It can also reinforce builder-side concentration when valuable searcher flow is routed privately or integrated with particular builders [9, 12, 15]. Searcher concentration can then propagate through the whole MEV supply chain: dominant searchers become preferred counterparties, builders with access to their flow gain an advantage, and validators face a block-building market shaped by privileged order-flow channels.

Most work on MEV concentration has focused on the block-building layer, analyzing how private order flow, vertical integration, latency advantages, and strategic bidding drive builder concentration under proposer-builder separation [2, 8, 9, 12, 14, 15, 24, 28, 29, 30, 31, 32]. Within this view, searchers are typically modeled merely as sources of transaction bundles or private flow that confer competitive advantages to builders. Consequently, the internal mechanics of the searcher auction market have received considerably less attention. While recent studies have begun to examine searcher mechanism design and bidding strategies [11], the specific implications of these auction formats for fairness and decentralization in the searcher market remain largely unexplored. This motivates our central question: how fair and decentralized is the current first-price, winner-take-all searcher auction, and can an alternative mechanism design improve reward decentralization while preserving the rigorous security guarantees required for permissionless MEV competition?

Decentralizing rewards in a permissionless environment is fundamentally difficult. If a mechanism pays multiple submissions, a searcher may split a single execution strategy into multiple identities, submitting copied or intentionally degraded versions of the same code. Consequently, any mechanism that naively rewards participation inherently invites Sybil attacks. Moreover, validators and searchers have strong incentives to coordinate outside the protocol if identity splitting can increase their joint payoff. Therefore, a decentralized searcher auction must solve two problems simultaneously: it must distribute rewards more broadly and fairly, while strictly preserving the security guarantees required for permissionless MEV execution.

This paper studies that tradeoff. We develop a model of heterogeneous searcher competition, analyze the standard first-price auction, and introduce a Shapley-based mechanism as a more decentralized alternative. The main message is that the current winner-take-all design is secure against copied-code deviations, but can be economically centralizing under realistic searcher heterogeneity. By contrast, a carefully capped and burned Shapley-based mechanism can improve contribution-adjusted fairness and reward decentralization, while controlling Sybil and validator–searcher coalition incentives through explicit Bayesian security constraints.

1.1 Our Contribution

Our contribution includes:

System model and evaluation criteria. We introduce a model of searcher competition in which searchers differ in both opportunity coverage and execution quality. For each opportunity class, a searcher may or may not discover and submit for the opportunity, and conditional on participation its submitted code has a certified normalized net value. This lets us separate two important sources of searcher advantage: seeing more opportunities and executing a given opportunity more efficiently. We then define economic and security criteria for evaluating searcher auctions. The economic criteria are a Shapley-weighted Jain fairness index, which measures whether rewards are aligned with marginal contribution, and an expected-reward HHI, which measures reward concentration. The security criteria are Bayesian copied-code Sybil resistance and Bayesian validator–searcher coalition resistance.

Analysis of the first-price auction. We analyze the first-price auction as the benchmark for current winner-take-all searcher competition. Under monotone searcher-specific bidding, copied or degraded versions of a searcher's own code cannot improve either the searcher's payoff or the joint payoff of the searcher and validator. Thus, the first-price auction has strong copied-code security properties. Economically, however, its rewards are governed by rank dominance: searchers are rewarded for being the highest-bidding active participant, not for their marginal contribution. We show that this distinction becomes important under heterogeneity. When execution quality is unequal, first-price rewards can concentrate on a small dominant class; in the limiting case of a single dominant searcher, the effective number of rewarded searchers approaches one.

Shapley-Capped Auction. We propose the Shapley-Capped Auction as an alternative design. The mechanism still executes the highest certified-value submission, but when the number of admitted submissions is below a cap, it distributes retained rewards according to the Shapley value of the validator-dependent max game. When the number of admitted submissions exceeds the cap, it switches to a conservative fallback branch and burns the residual. The cap and burn parameters are used to control the incentives created by reward sharing. We prove budget feasibility and provide a Bayesian security characterization showing that it is sufficient to consider a bounded range of Sybil sizes while optimizing over all feasible copied or degraded submissions. This yields a feasible region for the security parameters under which the mechanism satisfies copied-code Sybil resistance and validator–searcher coalition resistance.

Real-world data and empirical calibration. We analyze Ethereum arbitrage data to connect the model to observed searcher behavior. The goal is to estimate practical patterns of searcher participation, profit concentration, and opportunity coverage, and then use these

patterns to evaluate the mechanism under realistic parameters. This empirical component is not meant to fully identify every real-world searcher entity, since a single entity may control multiple contracts, but it provides a data-driven basis for calibrating the participation and execution-quality primitives used in the theoretical and numerical analysis.

1.2 Related work

Searcher's MEV activities. To produce competitive bundles, searchers rely on a combination of advantageous resources: efficient algorithms and heuristics to detect MEV opportunities, substantial capital to execute transactions across domains, and information asymmetries to shield lucrative opportunities from competitors. The strategic landscape has grown considerably more complex over recent years. Early MEV extraction was focused on *atomic* strategies, where all transactions constituting an attack execute within a single chain or block, guaranteeing all-or-nothing execution. Prominent examples include cyclic arbitrage across decentralized exchanges (DEX) and sandwich attacks on pending user trades [4, 17]. Later, the proliferation of Layer-2 systems and the frictions between heterogeneous market domains have increasingly pushed searchers toward *non-atomic* strategies, giving rise to arbitrage opportunities spanning multiple layers, chains, and off-chain venues such as centralized exchanges (CEX) [7, 13, 22].

Centralization in the MEV market. Block building on Ethereum has become highly concentrated: the leading builder, Titan, alone accounts for around 50% of blocks in May 2026. A prominent study explaining builder centralization is that of Öz et al. [12], which links builders' market share to order-flow diversity and their profitability to access to exclusive order-flow providers. Recent measurement studies on block builders also suggest that builder dominance is shaped not only by builder-side strategies, but more importantly by access to valuable searcher-generated bundles [9, 27, 31]. These bundles may derive their value from private information [14], more effective search and bidding strategies [10], or vertical integration between searchers and builders [15].

Compared with the extensive literature on block builders, systematic studies of searchers remain relatively recent. Searchers are heterogeneous in both strategy and market access. At the strategy level, searchers specialize in different forms of MEV extraction, including arbitrage, sandwich, liquidation, and mixed strategies [10]. At the access level, searchers may rely on public mempool transactions, private or exclusive order flow, MEV-Share signals[1], and off-chain information.

Moreover, searcher competition is unevenly distributed across opportunities and actors. Mamageishvili et al. [11] showed that validator revenue increases with searcher competition: when many searchers identify similar opportunities, competition pushes most of the value to the validator, while searchers retain more value when competition is weak. Empirical studies further confirmed that the searcher market is highly skewed toward large or specialized actors. For non-atomic arbitrage, Heimbach et al. [9] and Wu et al. [27] found that a small number of searchers account for most of the identified arbitrage volume and value.

These results suggest that searchers do not compete in a single homogeneous market – some MEV opportunities are highly contested, whereas others are repeatedly captured by searchers with persistent advantages. From a game-theoretic perspective, Mamageishvili et al. [11] provide an important foundation by modeling searcher competition as a cooperative

[1] `https://github.com/flashbots/mev-share`

submodular game, where the value function maps each coalition of searcher bundles to the highest block value achievable from that coalition. What remains less understood is how to model searcher competition in a way that captures heterogeneous opportunity access, variation in execution quality, and measurable competition intensity across the searcher market.

Game-Theoretic Approaches to MEV Auction Design. Recent game-theoretic analyses of MEV auctions increasingly emphasize the tension between allocative efficiency, decentralization, and resilience against strategic manipulation [19, 6]. While traditional winner-take-all procurement auctions inherently drive market concentration, alternative mechanisms that penalize monopoly allocations must strictly account for Sybil attacks in permissionless environments [6]. Pan et al. [16] demonstrate that the second-price auction with symmetric tie-breaking is the only non-wasteful, incentive-compatible, and Sybil-proof direct mechanism in single-parameter settings, severely limiting the design space for secure alternative distributions. To circumvent these constraints and achieve fair surplus allocation without compromising security, recent work applies cooperative game theory directly to order flow auctions (OFAs) [11]. Specifically, Shapley value-based frameworks have been proposed to equitably redistribute matchmaking revenues among transaction creators according to their marginal contributions, utilizing randomized approximation to handle the computational complexity of the MEV supply chain [18]. Complementing these static allocation approaches, Braga et al. [3] propose a dynamic MEV-sharing mechanism that adjusts the protocol-level extraction rate over time to balance participation incentives between users and MEV extractors. Together, this literature highlights a critical shift from purely competitive block building toward robust, Sybil-proof mechanisms and cooperative revenue-sharing models.

2 Model, Metrics, and Mechanisms

This section introduces a heterogeneous model of MEV searcher competition. Searchers differ in both opportunity coverage and execution efficiency. We evaluate mechanisms along two economic dimensions – contribution-adjusted fairness and reward decentralization – and two security dimensions – copied-code Sybil resistance and validator–searcher coalition resistance.

2.1 System Model

We model searcher competition for MEV opportunities in a setting with heterogeneous opportunity discovery, heterogeneous execution quality, and permissionless submissions. We focus on on-chain competition and abstract away from exogenous off-chain or cross-domain factors, such as centralized-exchange price movements and MEV opportunities originating from off-chain systems.

Entities and opportunity classes. There is one validator (proposer) and a finite set of potential searchers $\mathcal{S} = \{1, \ldots, n\}$. MEV opportunities are indexed by $o \in \mathcal{O}$. Each opportunity belongs to a class $\tau(o) \in \mathcal{C}$[2].

[2] A class may represent a broad MEV type, such as arbitrage, liquidation, sandwiching, or a finer subclass, such as arbitrage over a particular family of AMMs or routes.

Opportunity coverage. Searcher $i \in \mathcal{S}$ can participate in opportunity o only if it discovers the opportunity and can submit *execution code* before the deadline. We call such searchers *active searchers* for opportunity o. Let $D_{i,o} \sim \text{Bernoulli}(\alpha_{i,\tau(o)})$ denote this event. The parameter $\alpha_{i,\tau(o)} \in [0,1]$ is searcher i's coverage probability for class $\tau(o)$. The realized active set for opportunity o is

$$\mathcal{S}_{\text{act}}(o) = \{i \in \mathcal{S} : D_{i,o} = 1\}, \qquad N_o = |\mathcal{S}_{\text{act}}(o)|.$$

How opportunities are clustered into classes, and how $\mathcal{S}_{\text{act}}(o)$ is empirically identified, is outside the formal model.[3]

Execution code, certified values, and normalization. Any execution code is identified by two certified quantities under the canonical simulation rule: its gross realized value and its execution cost. Let $\tilde{v}_{i,o}$ denote the gross certified value generated by searcher i's code, and let $\tilde{c}_{i,o}$ denote its certified execution cost. The cost $\tilde{c}_{i,o}$ captures execution losses that are objectively computable from the submitted code, such as base-fee gas costs, realized slippage, and flash-loan fees; it does not include transfers to the validator. We define the certified net value of this code as $\tilde{z}_{i,o} = \tilde{v}_{i,o} - \tilde{c}_{i,o}$.

For each opportunity o, let $V_o > 0$ denote a public opportunity-scale parameter, known before the submission deadline. This scale represents the commonly observable or commonly estimated size of the opportunity and is used to normalize certified values across opportunities. We normalize these quantities by V_o: $v_{i,o} = \frac{\tilde{v}_{i,o}}{V_o}$, $\quad c_{i,o} = \frac{\tilde{c}_{i,o}}{V_o}$, $\quad z_{i,o} = \frac{\tilde{z}_{i,o}}{V_o}$. We assume V_o is chosen so that $v_{i,o} \le 1$ and $z_{i,o} \le 1$.

Submissions and copied-code deviations. For opportunity o, only active searchers with $D_{i,o} = 1$ can submit. A submission b consists of execution code and, depending on the auction format, may also include a mechanism-specific message m_b, such as a transfer offer. Each submission has a certified net value $\tilde{z}_b$ and certified normalized net value $z_b = \tilde{z}_b / V_o$, computed under the canonical simulation rule. In a permissionless setting, a searcher may attempt to increase its payoff by representing one execution code through multiple submissions. We model this through *copied-code Sybil deviations* and denote the finite set of submissions made by active searcher i by $\mathcal{B}_i(o)$.

▶ **Definition 1** (Copied-code Sybil deviation)**.** *Fix an opportunity o, an active searcher $i \in \mathcal{S}_{\text{act}}(o)$, and a submission $b \in \mathcal{B}_i(o)$ with certified normalized net value z_b. A k-submission copied-code Sybil deviation from b, with $k \ge 2$, is a collection of submissions $\sigma_{i,o} = (\hat{b}_1, \dots, \hat{b}_k)$ by the same real searcher i, with certified normalized net values $(z_{\hat{b}_1}, \dots, z_{\hat{b}_k})$, such that $z_{\hat{b}_\ell} \le z_b$ for every $\ell = 1, \dots, k$. The reference submission b is called the original submission.*

We denote by $\mathcal{B}(o)$ the set of all submissions for opportunity o, including both original submissions and Sybil submissions: $\mathcal{B}(o) = \bigcup_{i \in \mathcal{S}_{\text{act}}(o)} \mathcal{B}_i(o)$, and $t_o = |\mathcal{B}(o)|$. The validator can include at most one submitted execution code for opportunity o.

Bayesian information structure. Before the submission deadline, an active searcher i observes its own original execution code and its certified normalized net value. It does not observe the realized participation decisions, execution codes, certified values, or mechanism-

[3] One practical approach is to place submissions in the same active set $\mathcal{S}_{\text{act}}$ when executing the highest-net-value submission would reduce the residual profitability of the remaining submissions below a specified threshold.

specific messages of other searchers[4]. It only knows the corresponding distributional primitives. For each $j \neq i$, $D_{j,o} \sim \text{Bernoulli}(\alpha_{j,\tau(o)})$. Conditional on $D_{j,o} = 1$, the certified normalized net value of searcher j's original code for opportunity o, denoted by $z^\star_{j,o}$, is distributed according to the cumulative distribution function $F_{j,\tau(o)}$: $z^\star_{j,o} \sim F_{j,\tau(o)}, \quad z^\star_{j,o} \in [\underline{z}_{j,\tau(o)}, \overline{z}_{j,\tau(o)}] \subset (0,1)$. The distribution $F_{j,\tau(o)}$ is searcher- and class-specific: it describes searcher j's normalized performance across opportunities of class $\tau(o)$. After the submission deadline, the set of submissions is fixed. The mechanism then evaluates the submitted execution codes under the canonical simulation rule and observes their realized certified normalized net values. This interim information structure is the basis for the Bayesian-form security properties discussed later in the paper.

Auction mechanism. For each opportunity o, a searcher auction mechanism $\mathcal{M}$ takes the submitted execution codes, their certified net values, and any mechanism-specific messages as input. It selects one submission for execution and assigns payoffs to the real searchers and the validator. For each submission $b \in \mathcal{B}(o)$, let $\tilde{z}_b = \tilde{v}_b - \tilde{c}_b$ denote its certified net value. Let m_b denote its mechanism-specific message. Write $\tilde{z}_o = (\tilde{z}_b)_{b\in\mathcal{B}(o)}$ and $m_o = (m_b)_{b\in\mathcal{B}(o)}$. The mechanism specifies a winner-selection rule $\text{Win}^{\mathcal{M}}_o(\tilde{z}_o, m_o) \in \mathcal{B}(o)$, which selects a single winning submission. The validator includes the execution code associated with the selected submission. The mechanism also specifies payoff rules $U^{\mathcal{M}}_i(\tilde{z}_o, m_o)$ for each $i \in \mathcal{S}_{\text{act}}(o)$, and $U^{\mathcal{M}}_{\text{val}}(\tilde{z}_o, m_o)$ for the validator. The payoff $U^{\mathcal{M}}_i$ is the total payoff assigned to real searcher i, aggregating over all submissions in $\mathcal{B}_i(o)$.

2.2 Evaluation Criteria

We evaluate a searcher auction mechanism $\mathcal{M}$ along two dimensions: economic performance and security. The economic metrics measure whether rewards are fairly and broadly distributed among real searchers. The security properties require that these rewards cannot be profitably manipulated through copied-code Sybil submissions or validator–searcher coalitions. Let $U^{\mathcal{M}}_i$ denote searcher i's realized payoff and $U^{\mathcal{M}}_{\text{val}}$ the validator's realized payoff. Define searcher i's expected reward under $\mathcal{M}$ as $R^{\mathcal{M}}_i = \mathbb{E}[U^{\mathcal{M}}_i]$.

2.2.1 Economic metrics

The first economic metric measures whether expected rewards are aligned with searchers' contributions. We use the Shapley value as the contribution benchmark. The Shapley value is a standard solution concept from cooperative game theory: it assigns to each participant its average marginal contribution over all possible orders in which participants could join a coalition. In our setting, this provides a natural way to measure how much each searcher contributes to the expected surplus generated by competition for an opportunity.

Let $\mathcal{N} = \mathcal{S} \cup \{\text{val}\}$. For $A \subseteq \mathcal{N}$, define the expected value of coalition A in the validator–searcher game as

$$W^{\text{VS}}(A) = \begin{cases} \mathbb{E}\left[V_o \max_{i\in A\cap\mathcal{S}: D_{i,o}=1} z^\star_{i,o}\right], & \text{val} \in A, \\ 0, & \text{val} \notin A, \end{cases}$$

[4] This information structure can be implemented, for example, by an encrypted mempool or commit–reveal protocol in which validators and searchers do not observe other submissions before the epoch deadline for decryption or reveal.

with the convention that the maximum over an empty set is zero. Searcher i's validator–searcher Shapley contribution [21] is

$$\phi_i^{\mathrm{VS}} = \sum_{A \subseteq \mathcal{N} \setminus \{i\}} \frac{|A|!(n-|A|)!}{(n+1)!} \big(W^{\mathrm{VS}}(A \cup \{i\}) - W^{\mathrm{VS}}(A) \big).$$

Assume $\phi_i^{\mathrm{VS}} > 0$ for all searchers included in the metric. Define the reward-to-contribution ratio $r_i^{\mathcal{M}} = R_i^{\mathcal{M}} / \phi_i^{\mathrm{VS}}$. We then measure searcher-side fairness by applying Jain's index to these ratios: a mechanism is fair when searchers receive similar rewards per unit of validator-dependent Shapley contribution.

▶ **Definition 2** (Shapley-weighted Jain fairness). *The Shapley-weighted Jain fairness index of mechanism $\mathcal{M}$ is*

$$\mathcal{J}_{\mathrm{Sh}}^{\mathcal{M}} = \frac{\left(\sum_{i=1}^n r_i^{\mathcal{M}}\right)^2}{n \sum_{i=1}^n (r_i^{\mathcal{M}})^2} = \frac{\left(\sum_{i=1}^n R_i^{\mathcal{M}} / \phi_i^{\mathrm{VS}}\right)^2}{n \sum_{i=1}^n (R_i^{\mathcal{M}} / \phi_i^{\mathrm{VS}})^2}.$$

The index satisfies $1/n \le \mathcal{J}_{\mathrm{Sh}}^{\mathcal{M}} \le 1$, provided that at least one searcher receives positive expected reward. It equals 1 in a fully contribution-fair allocation, where all searchers receive the same reward per unit of Shapley contribution; equivalently, $R_i^{\mathcal{M}} = \lambda \phi_i^{\mathrm{VS}}$ for all i and some $\lambda > 0$. At the other extreme, it equals $1/n$ when only one searcher has a positive reward-to-contribution ratio, meaning that rewards are maximally concentrated relative to contribution. Note Jain's original index measures equality among the quantities to which it is applied. Since we apply it to $R_i^{\mathcal{M}} / \phi_i^{\mathrm{VS}}$, this metric measures equality of contribution-adjusted rewards.

The second economic metric measures how concentrated long-run expected rewards are among searchers. Whenever $\sum_{j=1}^n R_j^M > 0$, let $s_i^{\mathcal{M}} = \frac{R_i^{\mathcal{M}}}{\sum_{j=1}^n R_j^{\mathcal{M}}}$ be searcher i's expected reward share.

▶ **Definition 3** (Effective reward decentralization). *The expected-reward Herfindahl–Hirschman Index (HHI) and effective number of rewarded searchers are*

$$\text{ER-HHI}^{\mathcal{M}} = \sum_{i=1}^n (s_i^{\mathcal{M}})^2, \qquad N_{\mathrm{eff}}^{\mathcal{M}} = \frac{1}{\text{ER-HHI}^{\mathcal{M}}}.$$

The Herfindahl–Hirschman Index (HHI) is a standard concentration measure. Here we apply it to expected reward shares rather than market shares, so $\text{ER-HHI}^{\mathcal{M}}$ measures concentration in searcher rewards. It ranges from $1/n$, when all searchers receive equal expected reward shares, to 1, when one searcher receives all expected rewards. The reciprocal $N_{\mathrm{eff}}^{\mathcal{M}}$ gives the effective number of rewarded searchers: it equals n under equal reward shares and equals 1 when rewards are fully concentrated on a single searcher. Unlike $\mathcal{J}_{\mathrm{Sh}}^{\mathcal{M}}$, this metric does not adjust for contribution; it measures reward decentralization directly.

2.2.2 Security properties

The first security property concerns identity splitting by a searcher that can only copy or degrade its own execution code. For the security definitions, write $\mathcal{B}_{-i}(o) = \bigcup_{j \neq i} \mathcal{B}_j(o)$. We use $U_i^{\mathcal{M}}(\{b\} \cup \mathcal{B}_{-i}(o))$ to denote the payoff of real searcher i when it submits only the original submission b, and $U_i^{\mathcal{M}}(\sigma_i \cup \mathcal{B}_{-i}(o))$ to denote its payoff when it replaces b by a copied-code Sybil deviation σ_i. The payoff $U_i^{\mathcal{M}}$ aggregates over all submissions controlled by real searcher i.

A strong benchmark is ex-post copied-code Sybil resistance [16]. This property requires that copied-code identity splitting is never profitable for any realized profile of rival submissions.

▶ **Definition 4** (Ex-post copied-code Sybil resistance). *A mechanism $\mathcal{M}$ is ex-post copied-code Sybil-resistant if, for every opportunity o, every real searcher i, every original submission $b \in \mathcal{B}_i(o)$, every realized rival submission set $\mathcal{B}_{-i}(o)$, and every copied-code Sybil deviation σ_i from b, $U_i^{\mathcal{M}}(\{b\} \cup \mathcal{B}_{-i}(o)) \geq U_i^{\mathcal{M}}(\sigma_i \cup \mathcal{B}_{-i}(o))$.*

This security requirement states that, once the rival submissions $\mathcal{B}_{-i}(o)$ are fixed, a real searcher cannot increase its total payoff by replacing one original submission b with any feasible collection of copied-code submissions σ_i.

We next introduce a weaker security requirement that evaluates copied-code deviations in expectation rather than state by state [16]. Under this requirement, a copied-code Sybil deviation is ruled out if it does not increase the deviating searcher's expected payoff conditional on its own participation and certified normalized net value.

▶ **Definition 5** (Bayesian copied-code Sybil resistance). *A mechanism $\mathcal{M}$ is Bayesian copied-code Sybil-resistant if, for every real searcher i, every $z \in [\underline{z}_{i,\tau(o)}, \overline{z}_{i,\tau(o)}]$, every original submission b with $z_b = z$, and every copied-code Sybil deviation σ_i from b,*

$$\mathbb{E}\big[U_i^{\mathcal{M}}(\{b\} \cup \mathcal{B}_{-i}(o)) \mid D_{i,o} = 1,\ z^\star_{i,o} = z\big] \geq \mathbb{E}\big[U_i^{\mathcal{M}}(\sigma_i \cup \mathcal{B}_{-i}(o)) \mid D_{i,o} = 1,\ z^\star_{i,o} = z\big]\,.$$

This weaker Bayesian definition is appropriate for our introduced deadline-based system model (see Sect. 2.1). Before the submission deadline, a deviating searcher observes its own execution code and certified normalized net value, but cannot condition its copied-code Sybil strategy on the realized submissions of its competitors.

The second security property extends copied-code Sybil resistance to validator–searcher coalitions. It requires that replacing an original submission with copied-code submissions cannot increase the combined payoff of the deviating searcher and the validator. This property is important because, if a copied-code deviation can increase joint surplus, the searcher and validator may have a direct incentive to coordinate the deviation and share the gain. As above, we define both an ex-post version and a Bayesian version of this requirement.

▶ **Definition 6** (Ex-post validator–searcher coalition resistance). *A mechanism $\mathcal{M}$ is ex-post validator–searcher coalition-resistant under copied-code deviations if, for every opportunity o, every real searcher i, every original submission $b \in \mathcal{B}_i(o)$, every realized rival submission set $\mathcal{B}_{-i}(o)$, and every feasible copied-code Sybil deviation σ_i from b,*

$$U_i^{\mathcal{M}}(\{b\} \cup \mathcal{B}_{-i}(o)) + U_{\mathrm{val}}^{\mathcal{M}}(\{b\} \cup \mathcal{B}_{-i}(o)) \geq U_i^{\mathcal{M}}(\sigma_i \cup \mathcal{B}_{-i}(o)) + U_{\mathrm{val}}^{\mathcal{M}}(\sigma_i \cup \mathcal{B}_{-i}(o)).$$

This security requirement rules out copied-code deviations that increase the joint payoff of the deviating searcher and the validator, evaluated state by state for each realized rival submission set.

▶ **Definition 7** (Bayesian validator–searcher coalition resistance). *A mechanism $\mathcal{M}$ is Bayesian validator–searcher coalition-resistant under copied-code deviations if, for every real searcher i, every $z \in [\underline{z}_{i,\tau(o)}, \overline{z}_{i,\tau(o)}]$, every original submission b with $z_b = z$, and every feasible copied-code Sybil deviation σ_i from b,*

$$\begin{aligned}&\mathbb{E}\big[U_i^{\mathcal{M}}(\{b\} \cup \mathcal{B}_{-i}(o)) + U_{\mathrm{val}}^{\mathcal{M}}(\{b\} \cup \mathcal{B}_{-i}(o)) \mid D_{i,o} = 1,\ z^\star_{i,o} = z\big] \\ &\quad \geq \mathbb{E}\big[U_i^{\mathcal{M}}(\sigma_i \cup \mathcal{B}_{-i}(o)) + U_{\mathrm{val}}^{\mathcal{M}}(\sigma_i \cup \mathcal{B}_{-i}(o)) \mid D_{i,o} = 1,\ z^\star_{i,o} = z\big]\,.\end{aligned}$$

The Bayesian version applies the same joint-payoff comparison from the interim perspective of the deviating searcher. It rules out copied-code deviations that increase the expected combined payoff of the searcher–validator coalition under the deadline-based information structure.

3 First-Price Auction

We first analyze the sealed first-price auction (FPA), which is the natural benchmark for current private MEV bundle markets. In this mechanism, each active searcher submits a transfer offer to the validator; the highest offer wins, and the winner pays its own offer.

3.1 FPA Mechanism

For the FPA, the mechanism-specific message is the unnormalized transfer offer to the validator, $m_b = \widetilde{\mathsf{bid}}_b$. We write $\widetilde{\mathsf{bid}}_o = (\widetilde{\mathsf{bid}}_b)_{b\in\mathcal{B}(o)}$ for the vector of submitted transfer offers. The first-price auction selects the submission with the highest transfer offer, breaking ties by a fixed priority order, such as encrypted-mempool arrival order, fixed before reveal[5]:

$$\mathrm{Win}_o^{\mathrm{FPA}}(\tilde{z}_o, \widetilde{\mathsf{bid}}_o) \in \arg\max_{b\in\mathcal{B}(o)} \widetilde{\mathsf{bid}}_b.$$

When the argmax contains multiple submissions, $\mathrm{Win}_o^{\mathrm{FPA}}$ is the highest-priority submission among them. If submission $b^\star = \mathrm{Win}_o^{\mathrm{FPA}}(\tilde{z}_o, \widetilde{\mathsf{bid}}_o)$ wins, the validator includes the execution code associated with $b^\star$. The owner $s(b^\star)$ receives payoff $U_{s(b^\star)}^{\mathrm{FPA}}(\tilde{z}_o, \widetilde{\mathsf{bid}}_o) = \tilde{z}_{b^\star} - \widetilde{\mathsf{bid}}_{b^\star}$, and the validator receives $U_{\mathrm{val}}^{\mathrm{FPA}}(\tilde{z}_o, \widetilde{\mathsf{bid}}_o) = \widetilde{\mathsf{bid}}_{b^\star}$. All losing searchers receive zero.

For the FPA, each submission $b \in \mathcal{B}_i(o)$ made by searcher i has normalized certified net value z_b and submits the normalized transfer offer $\mathsf{bid}_b = \beta_{i,\tau(o)}(z_b)$, where $\beta_{i,\tau(o)}$ is searcher i's bidding function for opportunities of class $\tau(o)$. The corresponding unnormalized transfer offer is $\widetilde{\mathsf{bid}}_b = V_o \mathsf{bid}_b = V_o \beta_{i,\tau(o)}(z_b)$.

The formal security analysis of the FPA is deferred to Appendix A of the full version [20]. There we prove both copied-code Sybil resistance and validator–searcher coalition resistance in the strong ex-post sense; the corresponding Bayesian guarantees follow immediately.

3.2 Expected Rewards and Economic Properties under FPA

This section characterizes the economic allocation induced by the FPA. The key feature is that rewards are assigned only to the submission with the highest transfer offer. Hence, under monotone bidding, the FPA rewards the event of being the highest-bidding active searcher rather than marginal contribution to the opportunity.

Let $F_i = F_{i,\tau(o)}$ and $\alpha_i = \alpha_{i,\tau(o)}$. Suppose first that the value distributions are continuous and the bidding functions $\beta_{i,\tau(o)}$ are strictly increasing, so that, for any transfer offer bid in the range of $\beta_{i,\tau(o)}$, the inverse bidding function $\beta_{i,\tau(o)}^{-1}(\mathsf{bid})$ is well defined. Conditional on searcher i being active and having normalized certified net value z, its interim win probability is

$$\Pi_i^{\mathrm{win}}(z) = \prod_{j\neq i} \left(1 - \alpha_j + \alpha_j F_j\left(\beta_{j,\tau(o)}^{-1}(\beta_{i,\tau(o)}(z))\right)\right).$$

[5] For copied-code deviations, copied submissions cannot obtain higher tie-breaking priority than the original submission they copy.

Therefore searcher i's expected reward under the FPA is

$$R_i^{\mathrm{FPA}} = \mathbb{E}[U_i^{\mathrm{FPA}}] = \alpha_i V_o \int \big(z - \beta_{i,\tau(o)}(z)\big)\, \Pi_i^{\mathrm{win}}(z)\, dF_i(z),$$

and the validator's expected payoff is

$$R_{\mathrm{val}}^{\mathrm{FPA}} = \mathbb{E}[U_{\mathrm{val}}^{\mathrm{FPA}}] = V_o \sum_{i=1}^{n} \alpha_i \int \beta_{i,\tau(o)}(z) \Pi_i^{\mathrm{win}}(z)\, dF_i(z).$$

The economic metrics $\mathcal{J}_{\mathrm{Sh}}^{\mathrm{FPA}}$, ER-HHI^{FPA}, and $N_{\mathrm{eff}}^{\mathrm{FPA}}$ are obtained by substituting R_i^{FPA} into the definitions in Sect. 2.2. To obtain sharper comparisons, we use the proportional-bidding benchmark $\beta_{i,\tau(o)}(z) = \rho z$ for all searchers and opportunity classes, where $\rho \in (0,1)$. Then the FPA ranks submissions by certified normalized net value, while the winner keeps fraction $1-\rho$ of its certified net value. In this benchmark,

$$R_i^{\mathrm{FPA}} = V_o(1-\rho)\Theta_i, \qquad \Theta_i = \alpha_i \int z \prod_{j \neq i} (1 - \alpha_j + \alpha_j F_j(z))\, dF_i(z).$$

Thus all searcher reward shares, and hence ER-HHI^{FPA} and $N_{\mathrm{eff}}^{\mathrm{FPA}}$, are determined by the rank-dominance scores $(\Theta_i)_{i=1}^n$.

▶ **Proposition 8** (Symmetric setting). *Suppose $\alpha_i = \alpha$ and $F_i = F$ for all i. Under proportional bidding, $R_i^{\mathrm{FPA}} = R_j^{\mathrm{FPA}}$ for all i, j. Consequently,*

$$\text{ER-HHI}^{\mathrm{FPA}} = \frac{1}{n}, \qquad N_{\mathrm{eff}}^{\mathrm{FPA}} = n.$$

Moreover, since the Shapley contributions are also symmetric, $\mathcal{J}_{\mathrm{Sh}}^{\mathrm{FPA}} = 1$.

Proof. Symmetry gives $\Theta_i = \Theta_j$ for all i, j, hence equal expected rewards and equal reward shares. The HHI and effective-number claims follow immediately. The game $W^{\mathrm{VS}}(\cdot)$ is symmetric across searchers, so $\phi_i^{\mathrm{VS}} = \phi_j^{\mathrm{VS}}$ for all i, j, and therefore $R_i^{\mathrm{FPA}}/\phi_i^{\mathrm{VS}}$ is constant. ◀

The symmetric case provides a natural benchmark in which the FPA is decentralized and contribution-fair. Once searchers differ in coverage or execution quality, rewards become concentrated around the searchers most likely to win the rank competition.

▶ **Proposition 9** (Dominant execution-quality class with heterogeneous coverage). *Suppose $\alpha_i = \alpha_H \in (0,1)$ for every $i \leq m$ and $\alpha_j = \alpha_L \in (0,1)$ for every $j > m$, with $\alpha_H \geq \alpha_L$. For a distribution F, let $\mathrm{supp}(F)$ denote the set of normalized net values attainable with positive probability. Under proportional bidding, suppose there exist $m \in \{1, \ldots, n\}$ and $\underline{z}_H, \overline{z}_L \in (0,1)$, with $\underline{z}_H > \overline{z}_L$, such that $F_i = F_H$ and $\mathrm{supp}(F_H) \subseteq [\underline{z}_H, 1]$ for every $i \leq m$, while $\mathrm{supp}(F_j) \subseteq [0, \overline{z}_L]$ for every $j > m$. Then,*

$$N_{\mathrm{eff}}^{\mathrm{FPA}} \leq m \left(\frac{(1-(1-\alpha_H)^m)\underline{z}_H + (1-\alpha_H)^m (n-m)\alpha_L \overline{z}_L}{(1-(1-\alpha_H)^m)\underline{z}_H} \right)^2.$$

Consequently, as $\alpha_H \to 1$, $N_{\mathrm{eff}}^{\mathrm{FPA}} \to m$ and $\mathcal{J}_{\mathrm{Sh}}^{\mathrm{FPA}} \to m/n$.

Proof. Let $p_H = 1 - (1-\alpha_H)^m$ be the probability that at least one high-quality searcher is active. Since $\underline{z}_H > \overline{z}_L$, whenever at least one high-quality searcher is active, the FPA winner is in the high-quality class. Hence the total expected reward of the high-quality class is at least $V_o(1-\rho)p_H \underline{z}_H$. A low-quality searcher can win only when no high-quality searcher is active, which occurs with probability $(1-\alpha_H)^m$. Since each low-quality searcher is active

with probability α_L and has normalized net value at most $\overline{z}_L$, the total expected reward of the low-quality class is at most $V_o(1-\rho)(1-\alpha_H)^m(n-m)\alpha_L\overline{z}_L$. Therefore, if s_H denotes the total reward share of the high-quality class, then

$$s_H \geq \frac{p_H \underline{z}_H}{p_H \underline{z}_H + (1-\alpha_H)^m(n-m)\alpha_L\overline{z}_L}.$$

The m high-quality searchers are symmetric, since they have the same coverage probability and the same distribution F_H. Thus they receive equal expected rewards, so each has reward share s_H/m. Hence $\text{ER-HHI}^{\text{FPA}} = \sum_i (s_i^{\text{FPA}})^2 \geq m(s_H/m)^2 = s_H^2/m$. Since $N_{\text{eff}}^{\text{FPA}} = 1/\text{ER-HHI}^{\text{FPA}}$, we obtain

$$N_{\text{eff}}^{\text{FPA}} \leq m\left(\frac{p_H \underline{z}_H + (1-\alpha_H)^m(n-m)\alpha_L\overline{z}_L}{p_H \underline{z}_H}\right)^2.$$

As $\alpha_H \to 1$, we have $p_H \to 1$ and $(1-\alpha_H)^m \to 0$, so the upper bound converges to m. Low-quality reward shares converge to zero, while symmetry implies that the m high-quality searchers have equal limiting reward shares $1/m$. Hence $N_{\text{eff}}^{\text{FPA}} \to m$.

It remains to prove the Jain-index limit. For any low-quality searcher $j > m$, as $\alpha_H \to 1$,

$$R_j^{\text{FPA}} \leq (1-\rho)V_o\alpha_L(1-\alpha_H)^m\overline{z}_L \to 0.$$

On the other hand, using only the Shapley marginal contribution from the predecessor coalition $\{\text{val}\}$,

$$\phi_j^{\text{VS}} \geq \frac{1}{n(n+1)}\Big(W^{\text{VS}}(\{\text{val}, j\}) - W^{\text{VS}}(\{\text{val}\})\Big) = \frac{V_o\alpha_L\mathbb{E}_{F_j}[z]}{n(n+1)} > 0.$$

Hence $R_j^{\text{FPA}}/\phi_j^{\text{VS}} \to 0$ for every low-quality searcher $j > m$. For the m high-quality searchers, symmetry implies equal expected rewards and equal validator–searcher Shapley contributions; their common reward-to-contribution ratio has a positive limit. Therefore the Jain vector converges to one with m equal positive components and $n-m$ zero components. Its Jain index is $\frac{(mc)^2}{nmc^2} = \frac{m}{n}$, for any $c > 0$. Thus $\mathcal{J}_{\text{Sh}}^{\text{FPA}} \to m/n$. ◀

The proposition shows that FPA reward decentralization is governed by the size of the dominant class, not by the total number of searchers. When α_H is high, at least one dominant searcher is almost always active, so lower-quality searchers almost never win. Thus, the FPA concentrates rewards on the top class: on m searchers in general, and on a single effective monopolist when $m = 1$.

▶ **Corollary 10** (Single dominant searcher). *Under the assumptions of Proposition 9 with $m = 1$, if searcher 1 is the unique dominant searcher, then, as $\alpha_H \to 1$, $N_{\text{eff}}^{\text{FPA}} \to 1$, $\mathcal{J}_{\text{Sh}}^{\text{FPA}} \to \frac{1}{n}$.*

Figure 1 illustrates how increasing dominance affects reward concentration under the FPA. The horizontal axis varies the participation probability α_H of the high-quality class, while different curves correspond to different numbers m_H of elite searchers. As α_H increases, FPA rewards become increasingly concentrated within the dominant class: contribution-adjusted fairness declines toward m_H/n, and the effective number of rewarded searchers declines toward m_H.

The results show that strong asymmetry can substantially reduce the economic decentralization of the FPA. With coverage heterogeneity, FPA rewards concentrate around searchers who access opportunities more often. With execution-quality heterogeneity, concentration is sharper: rewards are captured by the dominant execution class, and when this class contains only a few searchers, the FPA converges toward monopolistic concentration. Thus, the winner-take-all structure of the FPA can amplify centralization in realistic MEV environments where both opportunity access and execution quality are unequal.

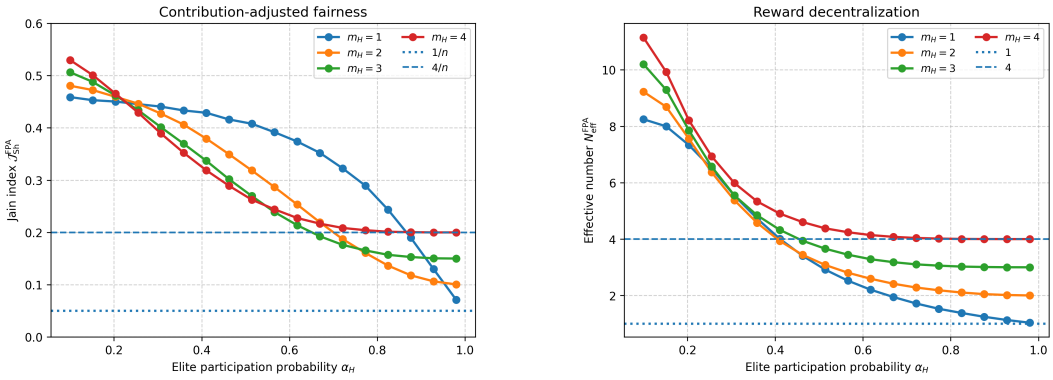


Figure 1 Effect of dominant-class participation on FPA fairness and reward decentralization.

4 The Entry-Filtered Shapley-Capped Auction

This section introduces a Shapley-based searcher auction mechanism designed to distribute MEV rewards more fairly and broadly among competing searchers. The goal is to align rewards with searchers' marginal contributions, reduce winner-take-all concentration, and maintain security against copied-code Sybil deviations and validator–searcher coalition deviations in a permissionless MEV environment.

4.1 Design Rationale

Our primary goal is to design a mechanism that, unlike a winner-take-all auction, distributes the searcher-side MEV reward among searchers that contribute to identifying or executing the same opportunity, rather than assigning all surplus to the top-ranked submission. The immediate difficulty is Sybil manipulation. Once rewards are distributed across multiple submissions, a searcher may submit copied or degraded versions of the same execution strategy to inflate its apparent contribution and capture a larger share. A natural solution is a Tullock-style contest [23], where proportional allocation with costly bids can be Sybil-proof because multiple bids can be aggregated into a single equivalent bid [23, 6]. However, this security relies on an all-pay structure: participants may obtain negative utility and must be willing to lose their bids. In an on-chain searcher auction, this would require deposits or enforceable losses, making the mechanism difficult to implement under current MEV infrastructure.

The broader obstruction is formal. Pan et al. show that any non-wasteful, symmetric, incentive-compatible, and ex-post Sybil-proof direct mechanism must reduce to a second-price auction with symmetric tie-breaking; equivalently, strong Sybil-proofness rules out nontrivial distributional objectives under those axioms [16]. Their Bayesian relaxation shows that this impossibility can be softened: an increasing-ticket-price mechanism can be Bayesian Sybil-proof by allocating the object through a lottery among threshold-clearing bidders whenever at least two such bidders exist, and otherwise falling back to a second-price auction [16]. Nevertheless, this construction is not well matched to MEV searcher competition. Its Bayesian Sybil deterrence relies on an ex-ante ticket threshold dictated by an upper-tail cutoff: the point beyond which the bid distribution's tail becomes no heavier than a uniform benchmark. While this condition is easily satisfied by distributions with flat or increasing right tails, it fails under the concentrated, declining upper tails typical of MEV signal models, such as the truncated log-normal distribution [1, 28]. In these cases, the required cutoff is

pushed to the maximum feasible valuation. Consequently, the ticket price also reaches this maximum, the lottery branch occurs with probability zero under continuous bids, and the mechanism entirely collapses back into a standard, winner-take-all second-price auction [16]. Appendix B of the full version [20] discusses Pan et al.'s characterization and the reduction of their mechanism to the second-price auction under our truncated-lognormal specification.

We therefore use the Shapley value as the economic allocation target, since it rewards searchers according to their marginal contribution to the opportunity. Shapley sharing alone, however, prevents neither copied-code Sybil manipulation nor validator–searcher collusion in a permissionless submission environment. The mechanism therefore combines Shapley-based rewards with residual caps and burn functions. The searcher-side burn is used to eliminate the expected gain from copied-code Sybil deviations, while the validator-side burn is calibrated to rule out profitable validator–searcher coalition deviations.

4.2 Entry-Filtered Shapley-Capped Auction Mechanism

The entry-filtered Shapley-capped auction mechanism is denoted by SCA_θ, where θ is the entry-filter threshold. Unlike the FPA, SCA_θ does not use transfer offers; its input is the set of valid submitted execution codes and their certified normalized net values. The mechanism first applies an entry filter: a submitted code is admitted only if its certified normalized net value is at least a protocol threshold $\theta \in [0,1]$. The threshold is chosen ex ante to control the expected number of admitted submissions. For opportunity class τ, one may choose $\theta = \theta_\tau$ so that $\mathbb{E}[N_o^\theta] = \sum_i \alpha_{i,\tau} \Pr_{F_{i,\tau}}(z^\star_{i,o} \geq \theta) \leq \bar{N}$, where $\bar{N}$ is the target upper bound on the expected admitted population.

Fix an opportunity o. Submissions with $z_b < \theta$ are rejected and receive zero. Let $\mathcal{B}^\theta(o) = \{b \in \mathcal{B}(o) : z_b \geq \theta\}$ be the set of admitted submissions, and let $t_\theta := |\mathcal{B}^\theta(o)|$. Their normalized and unnormalized certified net-value profiles are denoted by $z_o^\theta := (z_b)_{b\in\mathcal{B}^\theta(o)}$ and $\tilde{z}_o^\theta := (\tilde{z}_b)_{b\in\mathcal{B}^\theta(o)}$, respectively. If $t_\theta = 0$, no submission is executed; set $z^\theta_{(1)} := 0$, all searcher and validator payoffs to zero, and $B_o^{\mathrm{SCA}_\theta} := 0$. Thus, θ serves as a minimum certified-value requirement for execution.

Otherwise, order admitted submissions by $z^\theta_{(1)} \geq \cdots \geq z^\theta_{(t_\theta)}$, where $b^\theta_{(r)}$ is the rank-r admitted submission. Set $z^\theta_{(t_\theta+1)} = 0$ and $\Delta_k^\theta = z^\theta_{(k)} - z^\theta_{(k+1)}$. The executed submission is $\mathrm{Win}_o^{\mathrm{SCA}_\theta}(\tilde{z}_o^\theta) = b^\theta_{(1)}$, with ties broken by mempool arrival order.[6]

For any coalition $A \subseteq \mathcal{B}^\theta(o) \cup \{\mathrm{val}\}$, its realized value under SCA_θ is

$$W_o^\theta(A) = \begin{cases} V_o \max_{b\in A\setminus\{\mathrm{val}\}} z_b, & \mathrm{val} \in A,\ A \setminus \{\mathrm{val}\} \neq \varnothing, \\ 0, & \text{otherwise.} \end{cases}$$

Note that the validator is necessary for a coalition to generate value. For the admitted submissions $\mathcal{B}^\theta(o)$, the Shapley values of the validator and the rank-r admitted submission are

$$\Phi^\theta_{\mathrm{val}}(z_o^\theta) = V_o \sum_{k=1}^{t_\theta} \frac{k\Delta_k^\theta}{k+1}, \qquad \Phi^\theta_{(r)}(z_o^\theta) = V_o \sum_{k=r}^{t_\theta} \frac{\Delta_k^\theta}{k(k+1)}.$$

Hence $\Phi^\theta_{\mathrm{val}}(z_o^\theta) + \sum_{r=1}^{t_\theta} \Phi^\theta_{(r)}(z_o^\theta) = V_o z^\theta_{(1)}$.

[6] If two submissions from different identities have the same certified value, the earlier submission in the encrypted mempool is ranked first. The ordering is fixed before decryption or reveal, so copied submissions cannot improve tie-breaking after observing rivals.

The mechanism includes two branches: the decentralized Shapley-sharing branch and the capped fallback branch. The mechanism places a cap $H \geq 2$ on the number of admitted submissions that can be handled in the decentralized Shapley-sharing branch. This cap is chosen relative to the admitted population, so H should be moderately larger than $\mathbb{E}[N_o^\theta]$. Given retention parameters $\kappa_S, \kappa_V \geq 0$, define $\eta_{t_\theta}^V = e^{-\kappa_V(\min\{t_\theta, H\}-1)}$ and $\eta_{t_\theta}^S = e^{-\kappa_S(\min\{t_\theta, H\}-1)}$.

- If $t_\theta \leq H$, all admitted submissions are rewarded according to retained Shapley values:

$$U_{\mathrm{val}}^{\mathrm{SCA}_\theta}(\tilde{z}_o^\theta) = \eta_{t_\theta}^V \Phi_{\mathrm{val}}^\theta(z_o^\theta), \qquad P_{(r)}^{\mathrm{SCA}_\theta}(\tilde{z}_o^\theta) = \eta_{t_\theta}^S \Phi_{(r)}^\theta(z_o^\theta).$$

- If $t_\theta > H$, the number of admitted submissions exceeds the cap, and the mechanism switches to the capped fallback branch:

$$U_{\mathrm{val}}^{\mathrm{SCA}_\theta}(\tilde{z}_o^\theta) = \frac{H\eta_H^V}{H+1} V_o z_{(1)}^\theta, \qquad P_{(1)}^{\mathrm{SCA}_\theta}(\tilde{z}_o^\theta) = \frac{\eta_H^S}{H(t_\theta+1)} V_o z_{(1)}^\theta,$$

  and $P_{(r)}^{\mathrm{SCA}_\theta}(\tilde{z}_o^\theta) = 0$ for all $r \geq 2$.

Thus, when the number of admitted submissions is at most the cap, the mechanism uses Shapley sharing across admitted submissions. When the number of admitted submissions exceeds the cap, it switches to a fallback rule that pays only the top admitted submission on the searcher side while retaining a cap-dependent burn. For each real searcher i, its total payoff is the sum of the payments assigned to its admitted submissions, $U_i^{\mathrm{SCA}_\theta}(\tilde{z}_o^\theta) = \sum_{b \in \mathcal{B}_i(o) \cap \mathcal{B}^\theta(o)} P_b^{\mathrm{SCA}_\theta}(\tilde{z}_o^\theta)$. Any surplus remaining after paying the validator and all searchers is burned, $B_o^{\mathrm{SCA}_\theta}(\tilde{z}_o^\theta) = V_o z_{(1)}^\theta - U_{\mathrm{val}}^{\mathrm{SCA}_\theta}(\tilde{z}_o^\theta) - \sum_i U_i^{\mathrm{SCA}_\theta}(\tilde{z}_o^\theta)$.

▶ **Proposition 11** (Budget feasibility). *For every opportunity and every valid submitted profile,*

$$U_{\mathrm{val}}^{\mathrm{SCA}_\theta}(\tilde{z}_o^\theta) + \sum_{i \in \mathcal{S}_{\mathrm{act}}(o)} U_i^{\mathrm{SCA}_\theta}(\tilde{z}_o^\theta) + B_o^{\mathrm{SCA}_\theta}(\tilde{z}_o^\theta) = V_o z_{(1)}^\theta \leq V_o, \qquad B_o^{\mathrm{SCA}_\theta}(\tilde{z}_o^\theta) \geq 0.$$

Proof. If $t_\theta = 0$, the claim is immediate. If $1 \leq t_\theta \leq H$, Shapley budget balance gives $\Phi_{\mathrm{val}}^\theta + \sum_{r=1}^{t_\theta} \Phi_{(r)}^\theta = V_o z_{(1)}^\theta$, and $\eta_{t_\theta}^S, \eta_{t_\theta}^V \leq 1$ imply total payments at most $V_o z_{(1)}^\theta$. If $t_\theta > H$, total payments are at most

$$\left(\frac{H}{H+1} + \frac{1}{H(t_\theta+1)}\right) V_o z_{(1)}^\theta \leq \left(\frac{H}{H+1} + \frac{1}{H(H+2)}\right) V_o z_{(1)}^\theta < V_o z_{(1)}^\theta.$$

Thus the residual defining $B_o^{\mathrm{SCA}_\theta}$ is nonnegative, and the displayed identity follows by definition of the burn. ◀

The formal Bayesian security analysis of the entry-filtered SCA is given in Appendix C of the full version [20]. It shows that it is sufficient to consider copy sizes $q = 2, \ldots, H+1$, while optimizing over all feasible copied-value profiles. The parameters (κ_S, κ_V) are calibrated so that no such deviation increases the Bayesian interim payoff of either the searcher alone or the searcher–validator coalition. The calibration creates a security–efficiency tradeoff. Increasing κ_S strengthens searcher-side Sybil deterrence but also increases burn. Coalition resistance, by contrast, generally requires κ_V to lie within a feasible interval: identity-expansion deviations may impose a lower bound, while withdrawal deviations may impose an upper bound. When this security region is nonempty, $(\kappa_S^\star, \kappa_V^\star)$ can be chosen to minimize expected burn subject to both Bayesian security requirements. The calibration reported in Appendix D of the full version [20] gives priority to minimizing searcher-side burn.

4.3 Economic Properties of the Entry-Filtered SCA

We analyze the honest one-submission benchmark under the contribution-adjusted metrics defined in Sect. 2.2. The entry filter affects rewards by excluding submissions below the threshold from the reward-sharing game. Thus, for a searcher i, only states in which i is active and satisfies $z^\star_{i,o} \geq \theta$ can generate a positive SCA_θ payment.

Let $E^\theta_{i,o} := D_{i,o}\mathbf{1}\{z^\star_{i,o} \geq \theta\}$ indicate that searcher i is active and admitted, and let $N^\theta_o := \sum_i E^\theta_{i,o}$ be the admitted population. The cap H is chosen relative to N^θ_o, not the raw active count N_o. In particular, the threshold θ and the cap H should be calibrated jointly so that $\mathbb{E}[N^\theta_o]$ is controlled and H is moderately larger than the typical admitted population.

For a realized profile, let $\Phi^\theta_i(z^\theta_o)$ denote searcher i's realized Shapley value in the admitted max game, with $\Phi^\theta_i(z^\theta_o) = 0$ when i is not admitted. Let ϕ^{VS}_i denote searcher i's ex-ante Shapley contribution in the unfiltered validator–searcher game defined in Section 2.2.1. Define the admitted contribution coverage ratio $a_{i,\theta} := \frac{\mathbb{E}[\mathbf{1}\{E^\theta_{i,o}=1,\ N^\theta_o \leq H\}\Phi^\theta_i(z^\theta_o)]}{\phi^{\mathrm{VS}}_i}$, and the admitted contribution exposure ratio $b_{i,\theta} := \frac{\mathbb{E}[\mathbf{1}\{E^\theta_{i,o}=1\}\Phi^\theta_i(z^\theta_o)]}{\phi^{\mathrm{VS}}_i}$. The ratio $a_{i,\theta}$ is the share of searcher i's contribution denominator that is admitted and paid in the decentralized branch. The ratio $b_{i,\theta}$ is an upper exposure bound for the contribution available to i after thresholding. These ratios are defined for all searchers and are normalized by ϕ^{VS}_i. Unlike in the unfiltered case, $b_{i,\theta}$ need not be at most one, because removing ineligible rivals can increase an admitted searcher's Shapley value relative to its full-profile Shapley value.

Define the global lower and upper reward-ratio bounds $\lambda_\theta := e^{-\kappa_S(H-1)} \min_i a_{i,\theta}$ and $\Lambda_\theta := \max_i b_{i,\theta}$. Thus λ_θ and Λ_θ are bounds on the reward-to-contribution ratios $R^{\mathrm{SCA}_\theta}_i/\phi^{\mathrm{VS}}_i$ across all searchers.

▶ **Proposition 12** (Entry-filtered SCA reward bounds). *Under honest one-submission play, assume the entry threshold is chosen so that the admitted population is bounded by $t_{\max}$, and suppose the fallback searcher retention satisfies $\eta^S_H \leq H/t_{\max}$. Then, for every searcher i, $\lambda_\theta \leq \frac{R^{\mathrm{SCA}_\theta}_i}{\phi^{\mathrm{VS}}_i} \leq \Lambda_\theta$.*

Proof. On the decentralized branch $N^\theta_o \leq H$, the entry-filtered SCA pays $\eta^S_{N^\theta_o}\Phi^\theta_i(z^\theta_o)$. Since $\eta^S_{N^\theta_o} = e^{-\kappa_S(N^\theta_o-1)} \geq e^{-\kappa_S(H-1)}$, we have

$$R^{\mathrm{SCA}_\theta}_i \geq e^{-\kappa_S(H-1)}\mathbb{E}[\mathbf{1}\{E^\theta_{i,o}=1,\ N^\theta_o \leq H\}\Phi^\theta_i(z^\theta_o)] = e^{-\kappa_S(H-1)}a_{i,\theta}\phi^{\mathrm{VS}}_i.$$

Taking the minimum over i gives the lower bound.

For the upper bound, $U^{\mathrm{SCA}_\theta}_i \leq \Phi^\theta_i(z^\theta_o)$ pointwise. This is immediate in the decentralized branch. In fallback, non-winners receive zero, and the winner receives

$$\frac{\eta^S_H}{H(t_\theta+1)}V_o z^\theta_{(1)} \leq \frac{1}{t_\theta(t_\theta+1)}V_o z^\theta_{(1)} \leq \Phi^\theta_{(1)}(z^\theta_o),$$

where the first inequality uses $t_\theta \leq t_{\max}$ and $\eta^S_H \leq H/t_{\max}$. Hence $R^{\mathrm{SCA}_\theta}_i \leq \mathbb{E}[\mathbf{1}\{E^\theta_{i,o} = 1\}\Phi^\theta_i(z^\theta_o)] = b_{i,\theta}\phi^{\mathrm{VS}}_i$. Taking the maximum over i gives the upper bound. ◀

Define the contribution-side effective number by $N^{\phi,\mathrm{VS}}_{\mathrm{eff}} := \frac{\left(\sum_{i=1}^n \phi^{\mathrm{VS}}_i\right)^2}{\sum_{i=1}^n \left(\phi^{\mathrm{VS}}_i\right)^2}$, which measures how many economically meaningful searchers are represented by the validator–searcher Shapley contribution vector $(\phi^{\mathrm{VS}}_i)^n_{i=1}$.

Table 1 Asymmetric searcher populations. Scenarios become more centralized from S1 to S4.

Scenario	n	$\mathbb{E}[N_o]$	Class structure $m(\alpha, \tilde{c}, [\underline{c}, \overline{c}])$
S1: broad heterogeneity	20	9.60	$4(0.60, 0.15, [0.10, 0.25])$, $8(0.50, 0.30, [0.20, 0.45])$, $8(0.40, 0.52, [0.35, 0.70])$
S2: aligned coverage–quality advantage	20	10.10	$5(0.85, 0.14, [0.10, 0.25])$, $7(0.55, 0.30, [0.20, 0.45])$, $8(0.25, 0.55, [0.40, 0.70])$
S3: dominant high-quality class	20	8.00	$3(0.95, 0.10, [0.05, 0.20])$, $7(0.45, 0.30, [0.18, 0.45])$, $10(0.20, 0.52, [0.35, 0.70])$
S4: single elite searcher	20	9.43	$1(0.98, 0.10, [0.05, 0.20])$, $9(0.55, 0.30, [0.18, 0.45])$, $10(0.35, 0.52, [0.35, 0.70])$

▶ **Theorem 13** (Economic performance of the entry-filtered SCA). *Under the assumptions of Proposition 12, and assuming that* $\sum_{i=1}^{n} R_i^{\mathrm{SCA}_\theta} > 0$,

$$\mathcal{J}_{\mathrm{Sh}}^{\mathrm{SCA}_\theta} \geq \frac{4\lambda_\theta \Lambda_\theta}{(\lambda_\theta + \Lambda_\theta)^2}, \qquad N_{\mathrm{eff}}^{\mathrm{SCA}_\theta} \geq \left(\frac{\lambda_\theta}{\Lambda_\theta}\right)^2 N_{\mathrm{eff}}^{\phi,\mathrm{VS}}.$$

Proof. Let $q_i := R_i^{\mathrm{SCA}_\theta}/\phi_i^{\mathrm{VS}}$. By Proposition 12, $q_i \in [\lambda_\theta, \Lambda_\theta]$ for every i. Therefore, for any given i, $q_i^2 \leq (\lambda_\theta + \Lambda_\theta) q_i - \lambda_\theta \Lambda_\theta$, because $(q_i - \lambda_\theta)(q_i - \Lambda_\theta) \leq 0$. Summing over i gives $\sum_i q_i^2 \leq (\lambda_\theta + \Lambda_\theta) \sum_i q_i - n\lambda_\theta \Lambda_\theta$. For fixed bounds $[\lambda_\theta, \Lambda_\theta]$, the Jain index is minimized when the ratios are split between the two endpoints. Equivalently, the worst case satisfies $\mathcal{J}(q) = \frac{(\sum_i q_i)^2}{n \sum_i q_i^2} \geq \frac{4\lambda_\theta \Lambda_\theta}{(\lambda_\theta + \Lambda_\theta)^2}$. Hence $\mathcal{J}_{\mathrm{Sh}}^{\mathrm{SCA}_\theta} \geq \frac{4\lambda_\theta \Lambda_\theta}{(\lambda_\theta + \Lambda_\theta)^2}$. For the effective number, using $R_i^{\mathrm{SCA}_\theta} = q_i \phi_i^{\mathrm{VS}}$, $N_{\mathrm{eff}}^{\mathrm{SCA}_\theta} = \frac{(\sum_i q_i \phi_i^{\mathrm{VS}})^2}{\sum_i (q_i \phi_i^{\mathrm{VS}})^2} \geq \frac{\lambda_\theta^2 (\sum_i \phi_i^{\mathrm{VS}})^2}{\Lambda_\theta^2 \sum_i (\phi_i^{\mathrm{VS}})^2} = \left(\frac{\lambda_\theta}{\Lambda_\theta}\right)^2 N_{\mathrm{eff}}^{\phi,\mathrm{VS}}$. ◀

The entry threshold creates a security–coverage tradeoff. Increasing θ reduces the expected admitted population, $\mathbb{E}[N_o^\theta] = \sum_i \alpha_i \Pr(z_{i,o}^\star \geq \theta)$, which weakens copied-code Sybil incentives and can reduce the amount of searcher-side burn required for security. However, increasing θ can reduce $a_{i,\theta}$, because some genuine low-value contributions become ineligible for rewards. Thus the threshold changes how much of each searcher's contribution can receive a positive mechanism payment. For $\theta > 0$, the mechanism remains Shapley-based among admitted submissions, but avoids paying the entire long tail of low-certified-value submissions.

4.4 Asymmetric Entry-Filtered Benchmark

To evaluate the mechanisms under realistic searcher heterogeneity, we model execution costs by a truncated lognormal distribution. Conditional on participating in opportunity o, searcher i's normalized execution cost is

$$C_i \mid D_{i,o} = 1 \sim \mathrm{TruncLogNormal}(\mu_i, \sigma^2; [\underline{c}_i, \overline{c}_i]), \qquad z_i^\star = 1 - C_i.$$

Table 1 introduced the four asymmetric searcher populations. Searchers differ in both participation probability and cost efficiency. Lower median cost corresponds to a higher certified normalized net value. Each benchmark scenario has three searcher classes. A class entry $m(\alpha, \tilde{c}, [\underline{c}, \overline{c}])$ means m searchers with participation probability α, median cost $\tilde{c}$, and truncated-lognormal support $[\underline{c}, \overline{c}]$. The four scenarios are ordered from less concentrated to more centralized competition.

Table 2 compares the FPA with entry-filtered SCA_θ across different admission targets. For the entry-filtered SCA_θ, we report three threshold levels. The threshold θ is selected so that the expected admitted population satisfies $\mathbb{E}[N_o^\theta] \in \{8, 6, 4\}$. The cap is then chosen relative to the admitted population, using $H = \lceil \mathbb{E}[N_o^\theta] \rceil + 3$. For each θ, $\kappa_S^\star$ is the least searcher-side security parameter satisfying the Bayesian copied-code Sybil-resistance constraints over every realized value z, every copy size $q = 2, \ldots, H+1$, and every feasible copied-value profile $\mathbf{z}^c \in \mathcal{Z}_q^c(z)$. Notably, across all twelve benchmark calibrations, the binding deviation is the

Table 2 FPA versus entry-filtered SCA. Within each scenario, the SCA rows show increasingly selective admission rules; the FPA row is reported once as the winner-take-all benchmark.

Scenario	Mechanism	$\mathbb{E}[N_o^\theta]$	θ	H	$\kappa_S^\star$	$\mathcal{J}_{\text{Sh}}$	N_{eff}	Note
S1	SCA_θ	8	0.495	11	0.371	0.948	11.10	broad admission
	SCA_θ	6	0.620	9	0.293	0.706	9.67	intermediate admission
	SCA_θ	4	0.719	7	0.165	0.574	7.19	selective admission
	FPA	–	–	–	–	0.244	4.26	winner-take-all benchmark
S2	SCA_θ	8	0.580	11	0.415	0.661	8.72	broad admission
	SCA_θ	6	0.709	9	0.356	0.536	6.96	intermediate admission
	SCA_θ	4	0.790	7	0.265	0.288	5.21	selective admission
	FPA	–	–	–	–	0.251	5.00	winner-take-all benchmark
S3	SCA_θ	8	0.300	11	0.406	0.996	6.07	broad admission
	SCA_θ	6	0.598	9	0.355	0.693	5.45	intermediate admission
	SCA_θ	4	0.734	7	0.284	0.413	4.01	selective admission
	FPA	–	–	–	–	0.150	3.00	winner-take-all benchmark
S4	SCA_θ	8	0.467	11	0.388	0.970	3.84	broad admission
	SCA_θ	6	0.597	9	0.322	0.726	3.78	intermediate admission
	SCA_θ	4	0.683	7	0.201	0.500	3.23	selective admission
	FPA	–	–	–	–	0.075	1.04	winner-take-all benchmark

two-copy attack at the admission boundary, with $q = 2$, $z = \theta$, and $\mathbf{z}^c = (\theta, \theta)$. Hence, in these benchmark designs, the numerically worst deviation is a two-copy attack, although this need not hold in general.

Table 2 supports the main decentralization claim. As competition becomes more concentrated, the FPA reward allocation collapses: $N_{\text{eff}}^{\text{FPA}}$ falls from 4.26 in S1 to 1.04 in S4, and $\mathcal{J}_{\text{Sh}}^{\text{FPA}}$ falls from 0.244 to 0.075. By contrast, the entry-filtered SCA_θ consistently preserves a broader reward base. In the most centralized case, S4, the FPA is effectively controlled by one elite searcher, while SCA_θ with $\mathbb{E}[N_o^\theta] = 6$ achieves $N_{\text{eff}}^{\text{SCA}_\theta} = 3.78$ and $\mathcal{J}_{\text{Sh}}^{\text{SCA}_\theta} = 0.726$.

The other side of the result is the security–coverage tradeoff. Lower thresholds admit more searchers and improve decentralization, but require larger $\kappa_S^\star$. Higher thresholds reduce $\kappa_S^\star$, but exclude more low-value submissions from reward sharing. Thus, relative to the FPA, the entry-filtered SCA_θ offers substantially better reward decentralization, with the threshold θ controlling the cost of Bayesian Sybil resistance.

Additional numerical results on increasing elite execution advantage, together with validator-side security calibration, are reported in Appendix D of the full version [20].

5 Empirical Benchmarks

We evaluate the performance of searchers by analyzing historical MEV transactions on Ethereum[7]. In practice, a searcher entity may deploy multiple contracts to identify and capture MEV opportunities. However, establishing correlations between different contracts controlled by the same searcher entity is outside the scope of this paper. For the purposes of this section, the terms searcher and contract are treated as interchangeable.

Dataset. We use a dataset [10] developed by Luo et al., which contains 577,264 arbitrage transactions from block 16308203 to 18908892. In total, there are 855 contracts generating a total revenue of 37.79 million dollars and a total profit of 8.88 million dollars.

[7] Our implementation is available at `https://gitlab.esat.kuleuven.be/mev/searcher_game`.

Table 3 Summary of Searcher Strategies.

Strategy	#Contracts	#Tx	Revenue Share	Profit Share	Avg. Profit / Tx
C1	74	3,323	4.35%	14.02%	$374.74
C2	88	510,620	76.29%	73.57%	$12.79
C3	693	63,321	19.36%	12.40%	$17.39

Searcher strategies. Arbitrage searchers implement one or more trading strategies to capture MEV opportunities. These strategies often involve constructing cyclic transactions across decentralized exchanges, in which two or more tokens are swapped within a single atomic execution. When multiple searchers identify the same opportunity, they compete for inclusion by submitting higher bids.

We identify two commonly implemented arbitrage strategies. One prioritizes selectivity, targeting scarce opportunities leading to high profit margins. The other prioritizes volume, executing a large number of transactions with relatively low profit per transaction.

The contracts in the dataset are therefore classified into three strategy profiles: (C1) margin-oriented, which contains contracts in the top 10% by average profit per transaction; (C2) volume-oriented, which are those in the top 10% by total transaction count; and (C3) mixed-profile contracts, whose observed performance does not show a dominant margin-oriented or volume-oriented profile.

Among the classified contracts, ten satisfy both the margin-oriented and volume-oriented criteria. We assign these overlapping contracts to the volume-oriented group, since high transaction volume more directly captures repeated execution behavior. In addition, two contracts `0x03c6...67cc` and `0xeeaa...3376` are excluded from margin-oriented group after identifying volume-oriented traits in their behavior.

A summary of the classified contracts is shown in Table 3. The profit retained by a searcher varies substantially depending on the type of opportunity covered, with the average profit per transaction in the C1 group being approximately 30 times that of the C2 group.

Execution efficiency across strategies. Within each strategy group, contracts are further classified into three tiers based on their contribution to the group's total realized profit. Contracts are first ranked in descending order by their total profit within the strategy. We then compute the cumulative share of positive realized profit accounted for by this ranking. Elite contracts are defined as the smallest set of top-ranked contracts whose cumulative positive profit reaches 60% of the strategy's total positive profit. Middle contracts are the next set of ranked contracts needed to raise the cumulative positive-profit share from 60% to 90%. Tail contracts are all remaining contracts, including contracts with zero or negative realized profit. For each tier, we show the median profit-to-revenue ratio, which measures the share of generated revenue retained by searchers. A higher profit-to-revenue ratio indicates that searchers retain more value after paying the costs required to win inclusion.

The performance of each tier is summarized in Table 4. A strong concentration is observed within each strategy group, with a small fraction of top-tier contracts accounting for over 60% of total realized profit. A stark contrast emerges between the two strategies: the bottom tier of C1 retains a higher median profit-to-revenue ratio than the top tier of C2, 35.66% and 17.67% respectively. This hints that C1 faces considerably lower competition intensity.

Figure 2 illustrates the transaction distribution across profit-to-revenue ratio ranges. The profit-to-revenue ratio measures the fraction of arbitrage revenue that remains as realized profit after fees and bids, and thus provides an observable proxy for the retained margin of included transactions. Within each strategy, elite contracts generally exhibit higher

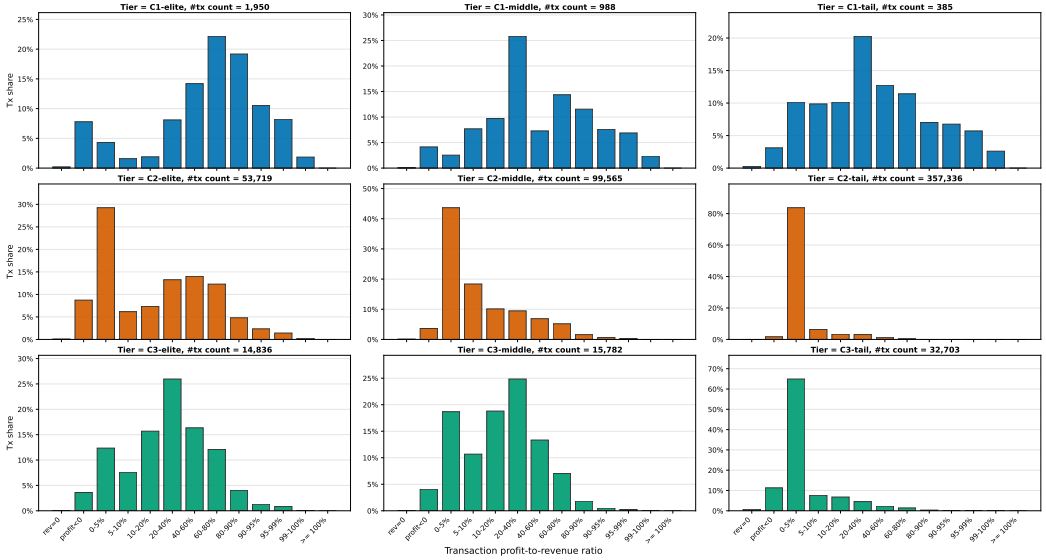


Figure 2 Transaction distribution over profit-to-revenue ratio bins, grouped by profit tier. The x-axis denotes profit-to-revenue ratio bins, while the y-axis reports the percentage share of transactions.

median profit-to-revenue ratios than middle and tail contracts, indicating that the contracts accounting for the largest share of realized profit also tend to retain a larger fraction of revenue on a per-transaction basis. Across strategies, C1 displays substantially higher retained margins than C2 and C3. This pattern is consistent with the interpretation that C1 searchers target arbitrage opportunities with lower relative execution or bidding costs, whereas C2 searchers operate in a more cost-intensive environment associated with high-frequency opportunity capture. Since the dataset contains only successful transactions and does not observe failed bids, the profit-to-revenue ratio should be interpreted as an ex-post measure of realized margin rather than a direct measure of auction-level bidding power.

Empirical calibration and limitations. Table 5 reports the truncated-lognormal cost distributions extracted from the Ethereum arbitrage data. The opportunity types C_1 and C_2 correspond to different empirical arbitrage environments: C_1 is margin-oriented, while C_2 is volume-oriented. Within each opportunity type, contracts are grouped into elite, middle, and tail tiers. A class entry $m(\alpha, \tilde{c}, [\underline{c}, \overline{c}])$ represents m contracts with relative coverage α, median normalized cost $\tilde{c}$, and truncated-lognormal support $[\underline{c}, \overline{c}]$.

Table 4 Summary of contract tiers within each strategy. (M) denotes median.

Tier	#Contracts	#Tx	Profit/Tx (M)	Profit Share	Profit/Revenue (M)
C1-elite	9	1,950	$561.76	62.08%	71.15%
C1-middle	17	988	$491.89	28.15%	40.28%
C1-tail	48	385	$274.35	9.77%	35.66%
C2-elite	11	53,719	$128.11	61.65%	17.67%
C2-middle	19	99,565	$38.32	28.86%	5.41%
C2-tail	58	357,336	$1.04	9.48%	1.26%
C3-elite	26	14,836	$45.37	62.12%	27.25%
C3-middle	51	15,782	$23.47	30.68%	18.74%
C3-tail	616	32,703	$0.02	7.20%	1.11%

A limitation is that the dataset contains only successful on-chain transactions. We therefore observe included winners, but not the full set of searchers competing for each opportunity, unsuccessful submissions, or builder- and relay-side auction messages. Since the model is defined over the full opportunity-level competition environment, the empirical calibration should be interpreted as a contract-level proxy rather than a fully identified structural estimate. Builder, relay, or private order-flow data would permit a more precise calibration of participation probabilities and value distributions.

Table 6 compares the FPA and the entry-filtered SCA_θ under these empirical settings. The raw expected active populations are $\mathbb{E}[N_o] = 14.565$ for C_1 and $\mathbb{E}[N_o] = 78.735$ for C_2. For each opportunity type, θ is selected so that $\mathbb{E}[N_o^\theta] = 14$, and the cap is set to $H = 17$. Thus, the target corresponds to near-full admission in C_1, but selective admission in C_2.

For each θ, $\kappa_S^\star$ is the smallest searcher-side parameter satisfying the Bayesian copied-code Sybil-resistance inequalities for every participating class, every realized value $z \geq \theta$, every $q = 2, \ldots, H+1$, and every feasible copied-value profile $\mathbf{z}^c \in \mathcal{Z}_q^c(z)$. The copied-value profile is selected before the rival profile is realized. In both empirical calibrations, the numerical supremum is attained by a two-copy exact-copy deviation: for C_1, by a middle-tier contract at $z \approx 0.143 > \theta$, and for C_2, by a middle-tier contract at $z = \theta$.

The empirical comparison suggests that SCA_θ is most valuable when FPA rewards are relatively concentrated. This is the case for C_1: the contribution-adjusted Jain index increases from 0.175 under the FPA to 0.983 under SCA_θ, while N_{eff} increases from 9.10 to 11.11. The increase in N_{eff} is more modest because the calibration treats the nine elite C_1 contracts as distinct searchers with identical class-level primitives. In practice, these contracts may be heterogeneous or share common ownership, implying stronger FPA reward concentration.

For C_2, the FPA is already highly decentralized at the contract level, with $\mathcal{J}_{\mathrm{Sh}}^{\mathrm{FPA}} = 0.997$ and $N_{\mathrm{eff}}^{\mathrm{FPA}} = 79.54$. The entry-filtered SCA therefore produces little additional contribution-adjusted fairness, but increases the effective reward base to $N_{\mathrm{eff}}^{\mathrm{SCA}_\theta} = 84.05$ while satisfying the reported Bayesian copied-code Sybil-resistance constraints. Overall, the empirical evidence supports the interpretation that the entry-filtered SCA is most consequential in relatively concentrated opportunity classes such as C_1. In already diffuse classes such as C_2, its economic effect is smaller, and its primary role is to preserve broad reward allocation under an explicit searcher-side Bayesian security calibration.

6 Conclusion

This paper argues that searcher competition is a distinct and important locus of MEV centralization. While the current first-price, winner-take-all auction is robust against copied-code identity splitting, its economic allocation is inherently rank-based: rewards accrue to the searcher most likely to submit the highest-value execution, rather than to searchers in proportion to their marginal contribution. Under heterogeneous access and execution quality, rank dominance can amplify small advantages into persistent reward concentration.

Table 5 Empirically extracted truncated-lognormal cost distributions.

Opportunity type	Tier	m	α	$\tilde{c}$, $[\underline{c}, \overline{c}]$
C_1	Elite	9	0.950	0.2885, $[0.01, 0.95]$
C_1	Middle	17	0.255	0.5972, $[0.05, 1.00]$
C_1	Tail	48	0.035	0.6434, $[0.01, 1.00]$
C_2	Elite	11	0.753	0.8233, $[0.01, 1.00]$
C_2	Middle	19	0.808	0.9459, $[0.10, 1.00]$
C_2	Tail	58	0.950	0.9874, $[0.20, 1.00]$

Table 6 FPA versus entry-filtered SCA for the empirical Ethereum-calibrated settings with $\mathbb{E}[N_o^\theta] = 14$. The SCA parameter $\kappa_S^\star$ is calibrated using the full Bayesian copied-value supremum.

Opportunity type	Mechanism	$\mathbb{E}[N_o^\theta]$	θ	H	$\kappa_S^\star$	$\mathcal{J}_{\mathrm{Sh}}$	N_{eff}	Note
C_1	SCA_θ	14	0.141	17	0.527	0.983	11.11	near-full admission
	FPA	–	–	–	–	0.175	9.10	winner-take-all benchmark
C_2	SCA_θ	14	0.392	17	0.457	0.999	84.05	selective admission
	FPA	–	–	–	–	0.997	79.54	winner-take-all benchmark

Motivated by this limitation, we introduced the entry-filtered Shapley-capped auction as one possible alternative. The mechanism preserves efficient execution by selecting the highest certified-value submission, but reallocates part of the retained surplus according to contribution among admitted high-quality submissions. Its cap, entry filter, and burn parameters expose the central design tradeoff: broader reward sharing improves contribution-adjusted fairness and reward decentralization, but must be disciplined by explicit Bayesian security constraints against copied-code Sybil deviations and validator–searcher coalitions. Overall, the results show that decentralization in MEV markets is not only a matter of infrastructure or order-flow access but also a mechanism-design problem requiring fairness, efficiency, and permissionless security to be addressed jointly.

References

1 Aleksei Adadurov, Sergey Barseghyan, Anton Chtepine, Antero Eloranta, Andrei Sebyakin, and Arsenii Valitov. Open vs. sealed: Auction format choice for maximal extractable value. *arXiv preprint*, 2026. arXiv:2603.16333.

2 Maryam Bahrani, Pranav Garimidi, and Tim Roughgarden. Centralization in block-building and proposer-builder separation. In Jeremy Clark and Elaine Shi, editors, *Financial Cryptography and Data Security - 28th International Conference, FC 2024, Willemstad, Curaçao, March 4-8, 2024, Revised Selected Papers, Part I*, Lecture Notes in Computer Science, pages 331–349. Springer, 2024. doi:10.1007/978-3-031-78676-1_19.

3 Pedro Braga, Georgios Chionas, Piotr Krysta, Stefanos Leonardos, Georgios Piliouras, and Carmine Ventre. Who gets the maximal extractable value? a dynamic sharing blockchain mechanism. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024)*, pages 2171–2173. International Foundation for Autonomous Agents and Multiagent Systems, 2024. doi:10.5555/3635637.3663097.

4 Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 910–927. IEEE, 2020. doi:10.1109/SP40000.2020.00040.

5 Flashbots. Searching Post-Merge, 2022. URL: https://writings.flashbots.net/searching-post-merge.

6 Pranav Garimidi, Michael Neuder, and Tim Roughgarden. Beyond winner-take-all procurement auctions. *arXiv preprint*, 2026. arXiv:2603.27779, doi:10.48550/arXiv.2603.27779.

7 Krzysztof Gogol, Johnnatan Messias, Deborah Miori, Claudio Tessone, and Benjamin Livshits. Cross-rollup MEV: Non-atomic arbitrage across L2 blockchains. *arXiv preprint*, 2024. arXiv:2406.02172.

8 Tivas Gupta, Mallesh M. Pai, and Max Resnick. The centralizing effects of private order flow on proposer-builder separation. In Joseph Bonneau and S. Matthew Weinberg, editors, *5th Conference on Advances in Financial Technologies, AFT 2023, Princeton, NJ, USA, October 23-25, 2023*, LIPIcs, pages 20:1–20:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.AFT.2023.20.

9 Lioba Heimbach, Vabuk Pahari, and Eric Schertenleib. Non-atomic arbitrage in decentralized finance. In *IEEE Symposium on Security and Privacy, SP 2024, San Francisco, CA, USA, May 19-23, 2024*, pages 3866–3884. IEEE, 2024. doi:10.1109/SP54263.2024.00256.

10 Feng Luo, Zihao Li, Wenxuan Luo, Zheyuan He, Xiapu Luo, Zuchao Ma, Shuwei Song, and Ting Chen. Light into darkness: Demystifying profit strategies throughout the MEV bot lifecycle. In *33rd Annual Network and Distributed System Security Symposium, NDSS 2026, San Diego, California, USA, February 23-27, 2026*. The Internet Society, 2026. URL: https://www.ndss-symposium.org/ndss-paper/light-into-darkness-demystifying-profit-strategies-throughout-the-mev-bot-lifecycle/.

11 Akaki Mamageishvili, Christoph Schlegel, and Benny Sudakov. Searcher competition in block building. In Rainer Böhme and Lucianna Kiffer, editors, *6th Conference on Advances in Financial Technologies, AFT 2024, Vienna, Austria, September 23-25, 2024*, LIPIcs, pages 21:1–21:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.AFT.2024.21.

12 Burak Öz, Danning Sui, Thomas Thiery, and Florian Matthes. Who wins Ethereum block building auctions and why? In Rainer Böhme and Lucianna Kiffer, editors, *6th Conference on Advances in Financial Technologies, AFT 2024, Vienna, Austria, September 23-25, 2024*, LIPIcs, pages 22:1–22:25. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.AFT.2024.22.

13 Burak Öz, Christof Ferreira Torres, Christoph Schlegel, Bruno Mazorra, Jonas Gebele, Filip Rezabek, and Florian Matthes. Cross-chain arbitrage: The next frontier of MEV in decentralized finance. *Proc. ACM Meas. Anal. Comput. Syst.*, 9(3):51:1–51:33, 2025. doi:10.1145/3771566.

14 Vabuk Pahari and Andrea Canidio. How exclusive are Ethereum transactions? Evidence from non-winning blocks. *arXiv preprint*, 2025. doi:10.48550/arXiv.2509.16052.

15 Mallesh M. Pai and Max Resnick. Structural advantages for integrated builders in mev-boost. In Jurlind Budurushi, Oksana Kulyk, Sarah Allen, Theo Diamandis, Ariah Klages-Mundt, Andrea Bracciali, Geoffrey Goodell, and Shin'ichiro Matsuo, editors, *Financial Cryptography and Data Security. FC 2024 International Workshops - Voting, DeFI, WTSC, CoDecFin, Willemstad, Curaçao, March 4-8, 2024, Revised Selected Papers*, Lecture Notes in Computer Science, pages 128–132. Springer, 2024. doi:10.1007/978-3-031-69231-4_9.

16 Minghao Pan, Bruno Mazorra, Christoph Schlegel, and Akaki Mamageishvili. On sybil-proof mechanisms. *arXiv preprint*, 2026. arXiv:2407.14485.

17 Kaihua Qin, Liyi Zhou, and Arthur Gervais. Quantifying blockchain extractable value: How dark is the forest? In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*, pages 198–214. IEEE, 2022. doi:10.1109/SP46214.2022.9833734.

18 Rasheed, Pujan Desai, Yash Chaurasia, and Sujit P. Gujar. Shapley value-based approach for redistributing revenue of matchmaking of private transactions in blockchains, 2025. arXiv:2502.15420.

19 Tim Roughgarden. Transaction fee mechanism design. *ACM SIGecom Exchanges*, 19(1):52–55, 2021. doi:10.1145/3476436.3476445.

20 Roozbeh Sarenche and Yunwen Liu. Towards decentralized searcher competition in MEV markets, 2026. doi:10.48550/arXiv.2608.05011.

21 Lloyd S. Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker, editors, *Contributions to the Theory of Games II*, volume 28 of *Annals of Mathematics Studies*, pages 307–317. Princeton University Press, 1953.

22 Christof Ferreira Torres, Albin Mamuti, Ben Weintraub, Cristina Nita-Rotaru, and Shweta Shinde. Rolling in the shadows: Analyzing the extraction of MEV across layer-2 rollups. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pages 2591–2605. ACM, 2024. doi:10.1145/3658644.3690259.

23 Gordon Tullock. Efficient rent seeking. In James M. Buchanan, Robert D. Tollison, and Gordon Tullock, editors, *Toward a Theory of the Rent-Seeking Society*, pages 97–112. Texas A&M University Press, 1980.

24 Shuzheng Wang, Yue Huang, Wenqin Zhang, Yuming Huang, Xuechao Wang, and Jing Tang. Private order flows and builder bidding dynamics: The road to monopoly in Ethereum's block building market. In Guodong Long, Michale Blumestein, Yi Chang, Liane Lewin-Eytan, Zi Helen Huang, and Elad Yom-Tov, editors, *Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025- 2 May 2025*, pages 2144–2157. ACM, 2025. doi:10.1145/3696410.3714754.

25 Wenhao Wang, Aditya Saraf, Lioba Heimbach, Kushal Babel, and Fan Zhang. Blockspace under pressure: An analysis of spam MEV on high-throughput blockchains. *arXiv preprint*, 2026. doi:10.48550/arXiv.2604.00234.

26 Ben Weintraub, Christof Ferreira Torres, Cristina Nita-Rotaru, and Radu State. A flash(bot) in the pan: measuring maximal extractable value in private pools. In Chadi Barakat, Cristel Pelsser, Theophilus A. Benson, and David R. Choffnes, editors, *Proceedings of the 22nd ACM Internet Measurement Conference, IMC 2022, Nice, France, October 25-27, 2022*, pages 458–471. ACM, 2022. doi:10.1145/3517745.3561448.

27 Fei Wu, Danning Sui, Thomas Thiery, and Mallesh M. Pai. Measuring CEX-DEX extracted value and searcher profitability: The darkest of the MEV dark forest. In Zeta Avarikioti and Nicolas Christin, editors, *7th Conference on Advances in Financial Technologies, AFT 2025, Pittsburgh, PA, USA, October 8-10, 2025*, LIPIcs, pages 26:1–26:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.AFT.2025.26.

28 Fei Wu, Thomas Thiery, Stefanos Leonardos, and Carmine Ventre. Strategic bidding wars in on-chain auctions. In *IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2024, Dublin, Ireland, May 27-31, 2024*, pages 503–511. IEEE, 2024. doi:10.1109/ICBC59979.2024.10634354.

29 Fei Wu, Thomas Thiery, Stefanos Leonardos, and Carmine Ventre. To compete or collude: Bidding incentives in Ethereum block building auctions. In *Proceedings of the 5th ACM International Conference on AI in Finance, ICAIF 2024, Brooklyn, NY, USA, November 14-17, 2024*, pages 813–821. ACM, 2024. doi:10.1145/3677052.3698629.

30 Fei Wu, Thomas Thiery, Stefanos Leonardos, and Carmine Ventre. From competition to centralization: The oligopoly in Ethereum block building auctions. In Inês Lynce, Nello Murano, Mauro Vallati, Serena Villata, Federico Chesani, Michela Milano, Andrea Omicini, and Mehdi Dastani, editors, *ECAI 2025 - 28th European Conference on Artificial Intelligence, 25-30 October 2025, Bologna, Italy - Including 14th Conference on Prestigious Applications of Intelligent Systems (PAIS 2025)*, Frontiers in Artificial Intelligence and Applications, pages 3387–3394. IOS Press, 2025. doi:10.3233/FAIA251209.

31 Sen Yang, Kartik Nayak, and Fan Zhang. Decentralization of Ethereum's builder market. In Marina Blanton, William Enck, and Cristina Nita-Rotaru, editors, *IEEE Symposium on Security and Privacy, SP 2025, San Francisco, CA, USA, May 12-15, 2025*, pages 1512–1530. IEEE, 2025. doi:10.1109/SP61157.2025.00157.

32 Ao Zhang, Yunwen Liu, Ren Zhang, Yingdi Shan, and Yongwei Wu. Order flow exclusivity and value extraction mechanisms: An analysis of Ethereum builder centralization. *arXiv preprint*, 2026. doi:10.48550/arXiv.2605.04471.

SoK: Encrypted Mempools Through the MEV Lens

Jonathan Passerat-Palmbach
Flashbots, London, UK
Imperial College London, UK

Abstract

MEV and censorship, fuelled by public mempool visibility, remain existential threats to Ethereum and have recently started to spread to its layer-two ecosystem. Encrypted mempools promise to conceal transaction content until ordering is final. While this sounds appealing, their viability rests on cryptographic, economic, and deployment trade-offs.

This paper systematises the evolution of threshold-encrypted mempools, from early schemes such as Shutter and Ferveo to the most recent research, and analyses how successive iterations have resolved bottlenecks like committee communication overhead, lack of pending transaction privacy, and position-dependent encryption. We highlight a convergence along four design axes, namely batched decryption to mitigate latency, silent setup to eliminate the complexity of distributed key generation, epochless encryption to remove position dependency, and collision-free encoding to prevent slot-collision censorship. We further survey the active Ethereum deployment debate, including EIP-8105 and the LUCID headliner submission, and map the requirements raised there onto the cryptographic corpus.

We conclude by exposing a critical limitation common to all current proposals: blind ordering and binary decryption together suppress not only the toxic part of MEV that motivated encrypted mempools, but also the same-block auction mechanisms that return value to users and sustain geographic decentralisation of the network. We argue for a programmable-privacy direction in which encrypted state can be acted on by trusted code before plaintext revelation.

2012 ACM Subject Classification Security and privacy → Cryptography; Security and privacy → Distributed systems security; Applied computing → Digital cash

Keywords and phrases Encrypted mempools, threshold encryption, MEV, Ethereum, proposer-builder separation, order-flow auctions, programmable privacy

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.22

Related Version *Full Version*: `https://eprint.iacr.org/2026/1643`

Acknowledgements The author thanks Quintus, Thomas, Mark, Sarah, and Luis, together with the authors who kindly presented their work to our reading group, for the discussions that shaped this systematisation.

1 Introduction

Maximal extractable value (MEV) is the value that a block proposer, or any party with influence over transaction ordering, can extract by reordering, censoring, or interleaving pending transactions, often at the expense of the users who submitted them [20]. The *mempool* is the network's broadcast queue of unconfirmed transactions awaiting inclusion in a block. On Ethereum, the proposer (the validator selected for each slot) can outsource block construction to a third-party *builder* under the proposer-builder separation (PBS) architecture [15], and the builder selects which transactions to include and in what order. We organise the design space for handling the incentives MEV creates in a permissionless network into three responses. Two passive responses let MEV manifest as spam, with *searchers*, the specialised actors who scan pending flow for extractable opportunities, competing for the same opportunity by submitting redundant transactions whose cost the chain absorbs in failed transactions and inflated gas usage, or let it concentrate into a latency race in


© Jonathan Passerat-Palmbach;
licensed under Creative Commons License CC-BY 4.0

which rewards accrue to the small number of operators who reach the proposer fastest, degrading the geographic distribution of the validator set [67]. The active response is to run auctions that mediate searcher competition and route the resulting value back to the user, an approach that has gained adoption through order-flow auctions [36] delivered by the Flashbots Auction [27], MEV-Blocker [19], and other providers.

These three responses map onto distinct mempool architectures, each striking a different balance between speed, decentralisation, and privacy. The public mempool is fast and decentralised but exposes pending transactions to any observer. As a result, the bulk of MEV-bearing transaction flow on Ethereum is now routed through out-of-protocol private channels concentrated in a small set of high-performance builders [51, 52]. Private mempool lanes are fast and shield content from public view, but require the user to trust the operator not to act on what it sees. Advocates of encrypted mempools argue that this trust model is brittle at scale, since every party with visibility of pre-confirmation transaction content has both the ability and the incentive to extract value once stakes are large enough [61]. They further state that the harms it enables, sandwich attacks on retail trades, real-time censorship, and opaque order-flow agreements, are central rather than residual to user experience [60, 46]. The combination of privacy and decentralisation, the third combination of the design space introduced above, is the one encrypted mempools aim to deliver by hiding transaction content cryptographically rather than by trust.

One currently deployed approach combining privacy and decentralisation is BuilderNet [28], a multi-operator block-building network in which each operator runs the same builder software inside a trusted execution environment (TEE) and users encrypt their transactions to the enclave's public key. The enclave can order, simulate, and place searcher backruns against fresh state while the plaintext remains hidden from the operator. The resulting MEV is refunded to the user. The TEE substrate moves the trust assumption from any individual builder to the hardware supply chain, which is a non-trivial shift, not one that every user accepts. Encrypted mempools, the subject of this paper, are the cryptographic alternative that removes the hardware-trust dependency.

The intuition behind an encrypted mempool is to hide the content of pending transactions until their ordering is committed, so that the information an adversary would need to mount a reordering or front-running attack is unavailable during the ordering window. The construction admits several cryptographic instantiations, and we focus on the threshold-encrypted mempool, in which a committee of validators jointly holds the decryption key and releases the plaintext once ordering is fixed. This variant has attracted the most attention because the validator set already has the right structure to serve as the committee, and the trust assumption maps cleanly onto the existing proof-of-stake trust model. We argue in this paper that, even setting aside the cryptographic gaps the literature has yet to close, threshold-encrypted mempools as currently proposed operate at the wrong design point: they suppress not only the toxic MEV they were designed to neutralise but also the auctions that today route value back to users. Instead, we believe a more desirable property is to preserve auction capabilities through the use of *programmable privacy*, in which secure and trusted code, an enclave or a cryptographic primitive, can operate on encrypted state before plaintext revelation.

The threshold-encrypted-mempool literature has grown rapidly in the last three years, organised around two parallel lines of work that resolve the principal bottlenecks of the original Shutter attempt [61]. Batched threshold encryption, introduced by Choudhuri et al. [17], makes the communication cost of decryption independent of batch size and removes the pending-transaction-privacy weakness present in per-epoch designs. Silent

threshold encryption [33] eliminates the interactive distributed key generation step and lets each committee member publish a hint from which a joint public key can be derived non-interactively. Subsequent work has combined these two lines and explored hybrid trust models in which the user retains a self-decryption option as a hedge against committee unavailability [64]. The rapid succession has left fragmented terminology, overlapping threat models, and unresolved trade-offs that the protocol-developer community is now actively weighing as it considers whether any of these constructions is ready for adoption at the consensus layer.

In this paper, we first develop a four-axis taxonomy of threshold-encrypted mempool constructions that organises the design space by setup interactivity, batching, epochless encryption, and post-quantum security, and we identify a convergent set of properties that practitioners have settled on but that no current construction jointly satisfies (Section 2). Second, we synthesise the protocol-developer perspective by surveying the Ethereum community proposals on the table and the requirements they articulate, showing how the cryptographic constructions match or fail to match those requirements (Section 3). Third, we develop an execution-quality argument that the design choice the community proposals share, namely blind ordering, imposes costs that the present proposals do not internalise even under a cryptographically perfect construction (Section 4). We conclude with four open problems spanning post-quantum security, the cryptographic-and-economic gap, network-layer metadata leakage, and decryption-aggregation cost (Section 5).

2 The cryptographic landscape

Threshold-encrypted mempools instantiate the encrypted mempool idea by entrusting a committee of validators with shared decryption capability, which fits into the existing trust assumptions of proof-of-stake consensus, since the same stake-weighted validator set can play both roles. While threshold decryption protocols have long been studied in cryptography, this section organises the literature on the specific cryptographic constructions required to meet the requirements of encrypted mempools.

Three design choices distinguish the constructions that follow, namely where the decryption committee sits, how much it must coordinate during setup, and when in the block lifecycle a ciphertext becomes openable. Each construction has been pushed against three failure modes that the deployment context generates: leakage of transactions that never make it into a block, ciphertexts that must commit to a target block height before the user knows which one, and ciphertexts that must commit to a slot index within a batch and so become vulnerable to slot-collision censorship. The argument of this section is that the literature has converged on four desirable properties, namely *silent setup*, *batched decryption*, *epochless encryption*, and *collision-free encoding*, each of which is individually achievable but that no single scheme attains together with post-quantum security; Table 1 makes the convergence explicit. Section 2.2 introduces the batched-decryption line, Section 2.3 introduces the silent-setup line, Section 2.4 covers their combination, Section 2.5 traces the path from epoch dependency to collision-free encoding, and Section 2.6 examines the two post-quantum candidates.

2.1 Problem statement and the first deployments

Shutter Network has been running on Gnosis Chain since October 2022 and is the first production threshold-encrypted mempool. Its protocol, documented in [23], builds on threshold Boneh-Franklin *Identity-Based Encryption* (IBE), with a permissioned committee

of *Keypers* holding shares of the IBE master secret. Users encrypt to an epoch identifier, and at the end of each epoch the committee releases a single epoch decryption key that opens every ciphertext bound to that epoch. The design trades a once-per-epoch coordination cost for a single $O(1)$ broadcast per epoch, regardless of how many transactions the epoch carries. Key release is on a deterministic schedule, synchronised against block finality.

The same release schedule is the source of Shutter's most-discussed limitation. Choudhuri, Garg, Piet, and Policharla [17] analyse Shutter and observe that the epoch key opens *every* ciphertext bound to its epoch, whether the transaction was included on chain or not. A user whose transaction failed to land in a block, for example because higher-fee transactions filled the block before theirs, has their cleartext revealed to every party watching the chain anyway, exposing trading intent that an attacker can replay against later submissions. They formalise the missing property as *pending transaction privacy*: a ciphertext that does not enter a block must not be decryptable by the committee. Shutter's deployment substantially predates a formal whitepaper, and the third-party analysis of [17] reconstructs the construction from production code, with a written protocol description by the original co-authors following only in [23].

A parallel early line, due to Bebel and Ojha [5], designs Ferveo for Tendermint-based proof-of-stake chains, with deployment in the Anoma stack. Ferveo runs a per-epoch distributed key generation against the active validator set and threshold-decrypts individual ciphertexts. The construction achieves pending transaction privacy by virtue of decrypting transactions one at a time after inclusion, but at the cost of $O(nB)$ communication for a batch of B transactions and a fresh DKG whenever the validator set rotates. Ferveo and Shutter therefore sit at opposite corners of the same trade-off: Shutter is communication-cheap on decryption but leaks unincluded transactions, while Ferveo preserves privacy slot by slot but pays for it linearly in batch size and in coordination per epoch.

The two failure modes of pending transaction privacy and per-epoch coordination motivate the remainder of the literature.

2.2 Batched decryption

Choudhuri et al. [17] introduce *batched threshold encryption* (BTE) as the formal alternative to both IBE-style epoch decryption and naive per-ciphertext threshold decryption. In a BTE scheme, a committee of n members chooses a subset of B ciphertexts from a larger mempool and jointly produces a partial decryption of total size $O(n)$, independent of B, that lets anyone open exactly that subset, while ciphertexts outside the chosen subset remain semantically secure. The construction in [17] casts the decryption rule as a weak form of witness encryption for a relation of shape "you can decrypt iff you know a BLS signature [11] on a KZG commitment [40] to the batch tags". The partial decryption is a single group element per committee member, and security is proved in the programmable random oracle model. The setup is a per-epoch multiparty computation that runs secret multiplications on shares to produce a batch-bound common reference string. The setup is more complex than a KZG powers-of-tau ceremony: no party ever holds the trapdoor in clear, and the shared powers are computed via Beaver-style triples. The scheme is practical only on chains willing to amortise such an MPC over a relatively long epoch.

The same group of authors removes the per-epoch MPC in their follow up paper [18]. The change consists in pushing the KZG commitment from the public statement into the witness and securing the resulting check with a shifted BLS signature [18]. This reduces the BTE setup to a single distributed key generation that holds across all future batches. The reported partial decryption is 48 bytes per committee member, with partial decryption taking roughly

3.2 seconds per committee member and reconstruction of the full plaintext batch taking a further 3.0 seconds for a 500-transaction batch in their single-threaded Rust microbenchmark, against a malicious adversary in the algebraic group model. The construction retains the BTE interface from [17], including the requirement that each ciphertext commit to a specific batch identifier; ciphertexts that fail to land in their committed batch are undecryptable thereafter. This is a milder failure mode than the Shutter epoch leak: a stuck transaction is no longer revealed, but it is also not recoverable, and the user must resubmit. This user-experience cost may block adoption in real-world deployments, as we revisit in Section 3. A weighted variant of the BTE interface, in which the threshold counts stake rather than committee heads, appears in [2] and is treated in Section 2.5.

What both BTE papers share is a design choice that became dominant in everything that followed. The decryption rule is expressed as a small set of pairing-product equations with linear verification, instantiated as threshold BLS signatures over a KZG commitment to the polynomial whose roots are the batch tags, with the message payload concealed via a witness encryption keyed on this relation. The compiler is general, and almost every subsequent pairing-based construction in this section reuses it, varying only what is in the witness and what is in the public statement. Garg et al. [32] establish the unifying perspective on this design choice by recognising these constructions as witness encryptions for "linearly verifiable" relations, composed from a small set of cryptographic gadgets such as signatures, inner products, and degree checks.

2.3 Silent setup

Even a single one-shot DKG is operationally heavy. A DKG requires synchronous interaction across the entire committee, and any committee rotation, whether intentional or membership-driven, forces a fresh proactive secret-sharing protocol [38] to migrate the shares. On Ethereum, where the validator set rotates every epoch and tens of thousands of validators come online and offline asynchronously, threshold cryptography that requires DKG-style setup does not map cleanly to the underlying network model.

Garg, Kolonelos, Policharla, and Wang [33] answer this with *silent threshold encryption* (STE). Each committee member independently generates a keypair under standard public-key infrastructure and publishes a constant-sized *hint* alongside their public key. Encryption is to the deterministic aggregate of the hints, and decryption is non-interactive in the sense that any party with the hints can verify and combine partial decryptions. The construction is pairing-based and rests on a novel *signature-based witness encryption* primitive that admits linear verification, with security proven in the generic group model. The reported encryption cost is under 7 ms in a Rust implementation. The headline limitation is that the scheme is not batched: communication is $O(n)$ per ciphertext, recovering the standard threshold encryption asymptotics. We return to the deployment cost of distributing the per-validator hints in Section 3. Independent of mempool privacy, the same silent-setup primitive has been proposed as a replacement for the relay layer in proposer-builder separation, where a small committee with no trusted dealer can enforce fair-exchange properties between builders and proposers [51].

Hall-Andersen, Simkin, and Wagner [37] revisit STE with a different set of design targets. The construction avoids both indistinguishability obfuscation, on which the earlier STE of Reyzin, Smith, and Yakoubov [56] rests, and the generic group model, on which the proof of [33] rests, and which the authors argue is too strong an idealisation to support a deployment. They introduce a new security notion called *one-shot adaptive security*, strictly stronger than static corruption but weaker than fully adaptive corruption. The construction

is, by the authors' admission, only CCA1-secure, and decryption communication remains $O(n)$ per ciphertext; its post-quantum instantiation and parameter sizes are treated in Section 2.6.

Concurrently, Waters and Wu [65] present a third silent threshold construction, also pairing-based but proved selectively secure under q-type assumptions in the plain model, that is, with no idealised heuristic such as the generic group or random oracle model to lean on. Their distributed threshold encryption scheme has ciphertexts of four source-group elements plus a short tag, and supports access structures beyond plain thresholds, namely arbitrary monotone Boolean formulae over the committee. Such a policy is any rule built from ANDs and ORs of committee members, with no negations, so that "members 1 and 2, or any of the members 5 through 9" is expressible where a plain t-of-n threshold is not [6]. It identifies weighted thresholds, important for stake-weighted committees, as an open problem specific to this plain-model line: earlier silent constructions do support weighted policies, but only through SNARK-based aggregation. Waters and Wu instead reach their plain-model proof with a SNARK-free technique, which they can unfortunately make work only for unweighted thresholds.

The three silent constructions disagree on the right security baseline. [33] accepts the generic group model in exchange for a concretely efficient pairing-based scheme. [37] pays larger ciphertexts and weaker chosen-ciphertext security for post-quantum security under standard lattice and isogeny assumptions. [65] stays in the pairing world but trades generic-group hardness for q-type assumptions in the plain model. A practical deployment must choose between these three positions explicitly, since they are not interchangeable.

2.4 Combining batched and silent

Batched decryption and silent setup are orthogonal properties, and a scheme that achieves both is the natural next step. Bormet, Choudhuri, Faust, Garg, Othman, Policharla, Qu, and Wang [13] present BEAST-MEV, the first scheme that bridges the two. The construction couples a homomorphic variant of [33] with a binary-decomposition encoding inspired by the BTE line, and the resulting Silent Batched Threshold Encryption primitive admits a one-time silent setup, batched decryption, and security in the generic group model against a fully malicious adversary. Reported figures are encryption in 16.4 ms and a 129.5-second reconstruction time for a batch of 512 transactions in Rust, the latter substantially slower than the corresponding figures in [18], which is the cost paid for the silent setup. BEAST-MEV remains pairing-based and is therefore not post-quantum.

The substance of [13], beyond the headline merger of the batched decryption and silent setup properties, is a sequence of optimisations that bring the cost of silent batched decryption into the same order of magnitude as the non-silent BTE line. The construction modifies the witness-encryption relation of [33] to admit additive homomorphism on a shared randomness scalar across ciphertexts, so that a single batched partial decryption serves all ciphertexts in a sub-batch of size ℓ, with $\ell = 8$ in the prototype. The batch is organised into $O(B/\ell)$ such sub-batches and the partial decryption is reconstructed across them. Randomness reuse across sub-batches and a multiplicative-sharing trick that avoids $\mathbb{G}_T$ elements bring the per-ciphertext partial decryption down to a single $\mathbb{G}_1$ element of 48 bytes, with 0.55 ms decryption time per ciphertext at $B = 512$. The underlying tag encoding is inherited from [14], however, and the resulting scheme inherits the slot-collision censorship attack surface analysed in the next subsection. Later constructions [3, 12], treated in Section 2.5, retain the two-mode structure of a one-time committee setup and per-batch work within a single pairing-product check.

2.5 Position dependency and slot-collision censorship

Bormet, Faust, Othman, and Qu [14] present BEAT-MEV, the first epochless BTE scheme, using key-homomorphic puncturable *PseudoRandom Functions* (PRF) composed with threshold ElGamal. The puncturable PRF acts as a per-ciphertext one-time pad whose key the committee can collectively reconstruct, and security against rogue ciphertexts follows from the puncturability of the key-homomorphic PRF. Encryption is reported in under 2 ms on standard hardware, and the scheme requires only a one-time setup that supports all future batches.

The price of removing the epoch identifier is that the user must commit to something else. In [14] each ciphertext carries an index i from a polynomially bounded slot namespace, and no two ciphertexts in the same batch can share a slot. Removing the epoch dependency uncovers a subtler failure mode. The authors classify the resulting failure modes carefully: an *honest collision* occurs when two users independently pick the same i and only one transaction survives, and an *adversarial collision* occurs when a watcher observes the mempool and submits a dummy ciphertext at the same slot as a victim, undercutting them on fee so that a rational proposer drops the victim. The adversarial case is a clean cryptographic-layer censorship vector: the attacker need not be a block builder, need not understand the transaction contents, and need not be in any particular network position.

The TrX system of Fernando, Policharla, Tonkikh, and Xiang [26] accepts a position commitment but invests in system-level integration with a high-throughput BFT consensus protocol. Each batch of ciphertexts is decrypted under a per-batch *rerandomised* KZG setup, indexed by a fresh scalar κ_{dc} per decryption context, which sidesteps the requirement that the user encrypt to a specific block height. Tags are derived from $H(\mathrm{pk})$ together with an ephemeral signature on the ciphertext, not from the block height or a polynomially bounded slot namespace. The resulting ciphertext is two $\mathbb{G}_2$ elements plus a symmetric payload, the smallest among the BTE constructions we are aware of, and is instantiated on the BN254 curve. The system-level observation is that the digest and the evaluation proofs of the BTE relation depend only on the published ciphertexts and not on the threshold shares, so both can be pipelined alongside the consensus voting rounds. The decryption shares are then released optimistically together with the final consensus vote, recovering a fast-path in which the residual pairing work after consensus completes amounts to a few operations per ciphertext. The reported end-to-end overhead is 27 ms, roughly 14% over the plaintext baseline, on a 50-node wide-area-network deployment against AptosBFT. The authors note that this construction resolves an open question, raised in earlier work on context-dependent threshold decryption, of whether such a primitive can be instantiated efficiently in a low-threshold regime. The trade-off is a common reference string that grows with the number of distinct decryption contexts per public key; on chains with two-hour epochs and ten blocks per second, this remains below one gigabyte and can be streamed into memory per epoch.

Agarwal et al. [2] extend the BEAT-MEV line to weighted committees (Weighted BTE - WBTE), an axis the rest of this section leaves implicit, keeping its one-time setup, epochless encryption, and single-group-element partial decryptions while weighing members by stake rather than equally. The scheme remains index-dependent: honest collisions surface as a small failure rate that a generalised cuckoo-hashing assignment drives below 2^{-40}, and the adversarial slot-collision vector of [14] carries over. The most recent line removes position commitment entirely. Boneh, Nema, Roy, and Tas [12] achieve epochless and collision-free BTE via partial-fraction decomposition over rational functions, extending an earlier construction of Jutla, Nema, and Roy [39]. A single group element acts as the pre-decryption

key for the entire batch; ciphertexts are two group elements plus a tag and are claimed to be CCA-secure under the construction's own definition. The construction requires an interactive setup, presented with MPC-based inversions on shares but also supporting a variant that needs only a single MPC multiplication, on par with the multiplication-only setups elsewhere in this section. The original presentation has $O(B^2)$ decryption, which the authors show can be improved to $O(B \log B)$ via FFT techniques, with a shorter-ciphertext variant added in a later revision. Policharla [55] presents an independent construction with the same epochless and collision-free properties, building directly on punctured powers of τ in pairing groups without partial fractions. The setup is interactive but multiplication-only, ciphertexts are one element each in $\mathbb{G}_1$ and $\mathbb{G}_T$, and decryption is $O(B \log B)$ via FFT. Security rests on the Decisional Bilinear q-Power Diffie-Hellman assumption [9], which is the same q-type family that underpins KZG, PLONK, and Ethereum's EIP-4844 blob commitments, rather than on a new q-type assumption specific to the construction. Concurrently, Agarwal, Das, Gilkalaye, Rindal, and Shoup [3] present BTX, also based on punctured powers of τ, achieving the same shortest ciphertext size within the collision-free line of $|\mathbb{G}_1| + |\mathbb{G}_T|$,[1] with $O(B \log B)$ decryption and concrete performance within a small factor of [55].

The three constructions [12, 55, 3] demonstrate that epochless and collision-free are jointly achievable in the pairing-based setting at $O(B \log B)$ decryption complexity. The concrete differences at $B = 512$ are small: end-to-end batch decryption of 2.80 s for [55], 3.61 s for the shorter-ciphertext variant of [12], and roughly half the time of [55] for [3] on the same hardware as reported by Policharla. A further observation in [55] separates the threshold protocol from the decryption work: the FFT-style cross-term computation depends only on the public ciphertexts and reference string, not the secret shares, so it runs in parallel with the threshold protocol, cutting post-protocol latency by roughly 12× at $B = 2048$ and leaving roughly B pairings on the critical path.

When it comes to ingenious tricks that make live deployments viable, it is worth considering the shared ideas and lineage between some of these BTE constructions.

The remaining design axis is post-quantum security.

2.6 Post-quantum paths

The constructions described above are all pairing-based, with security resting on discrete-logarithm-flavoured assumptions in bilinear groups. Two independent attempts have been made to step outside this regime.

Boneh, Laufer, and Tas [10] take the first of two post-quantum routes by tackling the batched-decryption side directly. The paper presents three epochless batch-decryption constructions: a key-policy ABE warm-up, a headline lattice scheme proved secure under the Learning with Errors assumption and therefore plausibly post-quantum, and a trilinear-map construction whose only known instantiation requires indistinguishability obfuscation [26].

Two observations from [10] matter for deployment. First, the lattice construction inherits a *succinctness vulnerability* once combined with the sub-batch techniques required to lift the encryptor-coordination requirement: an adversary that submits a full batch of ciphertexts under the same identity tag forces the pre-decryption key to grow linearly in the batch size, and the natural mitigation of capping the pre-decryption at the first $O(\log \ell)$ ciphertexts per tag opens a censorship vector in which $O(\log \ell)$ adversarial ciphertexts on a victim's tag

[1] The BTE papers report ciphertext size differently: [12] counts the $\mathbb{G}_T$ mask that TrX hashes away, and TrX excludes its ephemeral verification key and signature. Under a uniform convention the TrX ciphertext remains the smallest overall, so "shortest" here is within the collision-free line.

can exclude the victim from the decryptable set. Second, thresholdising the construction by secret-sharing a random matrix's trapdoor is possible but needs two rounds of interaction during pre-decryption, and the authors flag non-interactive lattice thresholding as an explicit open problem. The headline trade-off against the pairing line is parameter size: the pairing-based BTE of [55] has a ciphertext of $|\mathbb{G}_1| + |\mathbb{G}_T|$ plus a $2|\mathbb{F}|$ NIZK proof, in the hundreds of bytes on BLS12-381, against the kilobyte-scale parameters of [10].

Hall-Andersen et al. [37] study the orthogonal axis: where [10] targets batched decryption and accepts lattice-scale parameters, [37] targets silent setup and engineers a modular blueprint with two instantiations: a classical one on elliptic curves, and a post-quantum one combining the Kyber lattice KEM with an isogeny-based key exchange. The reported ciphertext sizes are 1.072 MB for the classical elliptic-curve instantiation and 1.431 MB for the post-quantum one, both at $n/3$ corruption and a 1 MB plaintext. The paper section on encrypted mempool deployment makes the architecture explicit: the user picks an ephemeral symmetric key K, encrypts the transaction body with AES under K, and uses the silent threshold scheme to encrypt only K, not the body. The small key-encapsulation ciphertext goes on chain via the mempool, and the larger symmetric ciphertext can either go on chain or sit on a separate availability layer. The technical contribution that makes this practical is the construction of small recipient public keys, replacing the linear-sized hints that [33] required.

The two post-quantum constructions are not interchangeable. [10] provides post-quantum batched decryption at kilobyte-scale parameters, and [37] provides post-quantum silent setup without batching. A construction combining silent setup, batched decryption, collision-free encoding, and post-quantum security under standard assumptions does not exist in the current literature to the best of our knowledge.

2.7 A four-axis taxonomy

The evolution of the field is naturally summarised along four binary axes introduced in this section, namely *setup* (per-epoch MPC, one-time DKG, or silent PKI-based), *decryption* (batched in the BTE sense versus standard per-ciphertext), *epochless* (whether the encryptor must commit to a target block), and *collision-free* (whether the encryptor must commit to a slot index within a batch),[2] together with a fifth column for post-quantum security under standard cryptographic assumptions. Table 1 places every scheme along these axes, alongside the security notion and idealised model in which the construction is proved secure, the committee-communication complexity of decrypting a batch, and the computational cost of that decryption. GGM, AGM, ROM, and "standard" in the security column denote the generic group, algebraic group, random oracle, and plain models respectively.

Three observations follow directly from Table 1. First, within the pairing-based corpus, silent setup, batched decryption, and the collision-free property each appear in multiple schemes, but no construction achieves all three together: [13] is silent and batched but not collision-free, [3, 55, 12] are batched and collision-free under a one-time DKG or MPC setup, and the silent constructions [33, 37, 65] do not batch. Combining the silent-batched line with the collision-free encoding of [3, 55, 12] appears to be the natural near-term direction. Second, post-quantum security is essentially orthogonal to the other axes as currently constructed: [10] achieves it in the BTE line under a one-time DKG, and [37] achieves it in the silent line

[2] Following [3]: no user-chosen index in the ciphertext, ruling out honest user collisions and the adversarial slot-collision censorship of [14].

Table 1 Taxonomy of threshold-encrypted mempool constructions, grouped by the lineages traced in this section. The four binary axes (setup, decryption, epochless, coll.-free) characterise the design choices analysed in this section, with post-quantum security as a fifth column. **Bold** marks a cell meeting the corresponding convergent property (silent setup, batched decryption, epochless encryption, collision-free encoding) or post-quantum security: no row is bold across all five columns, which is the convergence gap discussed in Section 2.7. The two closest deployment candidates under the Wagner-Ma criteria (Section 3) are set in **bold type** and marked †. The "Security" column reports the security notion together with the idealised model in which the construction is proved secure, and B-IND-CCA denotes the construction-specific batched chosen-ciphertext notion used by each BTE paper. The "Committee comm." column reports the total committee bandwidth to decrypt a batch of B ciphertexts, and the "Decryption cost" column the total computational cost of that decryption over the same unit, counting group operations and pairings, with FFT-style optimisations applied where the authors report them and $\tilde{O}$ suppressing logarithmic factors. "n/a" under "Coll.-free" marks schemes without a BTE-style batch interface, where index collision is not a meaningful failure mode. "Yes (lattice)" under "Post-quantum" indicates a concrete post-quantum instantiation, and "Yes (mod.)" indicates a modular blueprint instantiable post-quantum under standard assumptions. BLT's headline lattice construction proves selective IND-CPA in the random oracle model under LWE, weaker than the B-IND-CCA notion of the pairing line, with CCA2 reachable via the standard transforms the authors cite. The paper reports no benchmark or tight batch-size asymptotic, so the "Decryption cost" entry is the polynomial-in-B bound its lattice-arithmetic algorithm admits. WBTE's headline property, stake-weighted thresholds, is orthogonal to the four axes and not shown as a column.

Scheme	Setup	Decryption	Epochless	Coll.-free	PQ	Security	Committee comm.	Decryption cost
Epoch-bound pioneers (Section 2.1)								
Shutter [23]	1-time DKG	IBE-batched	No	n/a	No	IND-CPA, ROM	$O(1)$ per epoch	$O(n+B)$
Ferveo [5]	per-epoch DKG	Standard	No	n/a	No	IND-CCA, ROM	$O(nB)$ per batch	$O(nB)$
Batched decryption (Sections 2.2 and 2.5)								
CGPP24 [17]	per-epoch MPC	**Batched**	No	No	No	B-IND-CCA, ROM	$O(n)$ per batch	$O(B \log B)$
CGPW24 [18]	1-time DKG	**Batched**	No	No	No	B-IND-CCA, AGM+ROM	$O(n)$ per batch	$O(B \log B)$
BEAT-MEV [14]	1-time DKG	**Batched**	**Yes**	No	No	B-IND-CCA, AGM+ROM	$O(n)$ per batch	$O(B^2)$
TrX [26]	1-time DKG	**Batched**	**Yes**	**Yes**	No	B-IND-CCA, GGM	$O(n)$ per batch	$\tilde{O}(n+B)$
BLT [10]	1-time DKG	**Batched**	**Yes**	**Yes**	**Yes (lattice)**	sel. IND-CPA, ROM	$O(n)$ per batch	poly(B) lattice
WBTE [2]	1-time MPC	**Batched**	**Yes**	No	No	B-IND-CCA, AGM+ROM	$O(n)$ per batch	$O(B \log B)$
Silent setup (Section 2.3)								
GKPW STE [33]	**silent**	Standard	**Yes**	n/a	No	IND-CPA, GGM	$O(n)$ per ciphertext	$O(n)$
HSW STE† [37]	**silent**	Standard	**Yes**	n/a	**Yes (mod.)**	IND-CCA1, standard	$O(n)$ per ciphertext	$O(n)$
Waters-Wu [65]	**silent**	Standard	**Yes**	n/a	No	IND-CPA, plain	$O(n)$ per ciphertext	$O(n)$
Silent and batched (Section 2.4)								
BEAST-MEV† [13]	**silent**	**Batched**	**Yes**	No	No	B-IND-CCA, GGM	$O(n)$ per batch	$O(B^2)$
Epochless and collision-free (Section 2.5)								
BNRT [12]	1-time MPC	**Batched**	**Yes**	**Yes**	No	B-IND-CCA, AGM+ROM	$O(n)$ per batch	$O(B \log B)$
Policharla 26 [55]	1-time MPC	**Batched**	**Yes**	**Yes**	No	B-IND-CCA, AGM+ROM	$O(n)$ per batch	$O(B \log B)$
BTX [3]	1-time DKG	**Batched**	**Yes**	**Yes**	No	B-IND-CCA, AGM+ROM	$O(n)$ per batch	$O(B \log B)$

for standard threshold encryption only, so the four-property convergence with post-quantum security under standard assumptions remains open. Third, every construction in Table 1 that achieves both byte-scale ciphertexts and batch-independent decryption communication is pairing-based: the lattice scheme of [10] reaches succinct pre-decryption keys at kilobyte-scale parameters, the post-quantum silent line of [37] sits at megabyte-scale, and the smallest pairing-based ciphertexts of [55, 3] at $|\mathbb{G}_1| + |\mathbb{G}_T|$ are roughly 700 bytes on BLS12-381. Pairings therefore remain a critical primitive for the combination of byte-scale ciphertexts and batch-independent decryption communication that real-world deployments require, and a post-quantum replacement would have to give up one of these properties or develop a new technique.

3 The blockchain protocol developer's perspective

The cryptographic literature surveyed above is currently being read against an active deployment push on Ethereum. In December 2025, Luhn [45] proposed *EIP-8105: Universal Enshrined Encrypted Mempool*, a scheme-agnostic encrypted mempool architecture that admits arbitrary decryption providers including threshold encryption, MPC committees, trusted execution environments, delay encryption, and FHE schemes. In February 2026, Elowsson, Ma, and Florentine submitted [25] a minimum viable EIP of LUCID as a candidate headliner for the next Ethereum protocol fork. We focus on Ethereum throughout this paper because the deployment debate there is the most concrete: the Shutterized Gnosis Chain anchors a production baseline, while EIP-8105 and the LUCID headliner are the most advanced in-protocol proposals, and the discussion on the Ethereum Magicians forum makes the requirements that the cryptographic corpus has not yet jointly met explicit. We organise the remainder of this section around the community proposals themselves, the requirements they place on the cryptographic schemes surveyed earlier, and the questions that the cryptographic literature does not yet address.

3.1 Community proposals

Wagner and Ma [64] carry out a parallel survey from the protocol-developer side and arrive at an explicit list of requirements for a threshold-encrypted mempool at the consensus layer, namely silent setup with small public keys, non-interactive decryption, ciphertexts small enough to fit in blob payloads, and CCA2 security. Their requirements track closely with the design axes our literature review organises in Table 1, with silent setup matching the setup-ceremony axis, blob-compatible ciphertext size implied by the decryption type, and CCA-style security a finer reading of the security axis. The sharper consequence Wagner and Ma draw is that no current scheme meets all four requirements together, with [37] the closest match but only CCA1-secure. No threshold-encryption scheme in the corpus can therefore be enshrined into the consensus layer as it stands, and any deployable design must instead accommodate flexibility over which scheme is used. The authors also propose a concrete design, a *hybrid encrypted mempool*, in which the user encrypts a transaction towards both a threshold committee and a self-held key, with either party able to decrypt. Liveness drives this choice: a threshold committee can go offline, and a design that depends on the committee alone risks stalling the mempool. The security of the construction reduces to the user being unable to predict committee availability, which the design enforces through frequent committee rotation and upfront gas charging that disincentivises selective non-revelation.

Elowsson [24] takes the implication of cryptographic non-enshrinement to its logical conclusion in LUCID, an encrypted mempool design that is explicitly *decryption-agnostic*. The decryptor for any given transaction may be a threshold committee, a single trusted

party, or the user themselves, with the choice surfaced as a protocol parameter rather than a built-in. Two design properties of LUCID matter for the discussion below. First, execution of decrypted transactions is delayed to the next block, so the protocol effectively acts on one-slot-stale information. Second, transaction ordering inside a block is committed by the builder while the encrypted bodies are still sealed, a property we will call *blind ordering*. The stated goal of the design is to suppress the toxic part of MEV, namely the front-running and sandwich attacks that exploit visible mempool state, while leaving non-toxic activity such as backrunning arbitrage open. Thiery [62] subsequently proposes *encrypted frame transactions*, a same-slot variant of LUCID that uses order-execute separation to bring decrypted execution into the slot in which the encrypted transaction was committed. LUCID's authors did however show concern about the same slot design on the grounds that it moves key synchronisation onto the critical path, which compresses the propagation window for the rest of the block and introduces builder timing games [59].

3.2 Applicability to Ethereum

Reading Table 1 through the Wagner-Ma filter, only two constructions sit one requirement short of meeting all four together. HSW STE [37] has silent setup, blob-fittable ciphertexts via hybrid encryption, and a modular post-quantum instantiation, but only attains CCA1 rather than CCA2. BEAST-MEV [13] is silent and batched with small ciphertexts and B-IND-CCA security, but is pairing-based and therefore not post-quantum, with security proven in the generic group model. All other constructions in the corpus fail at least two of the four requirements, predominantly on the silent-setup axis. By the Wagner-Ma criteria, these two schemes are the closest credible deployment candidates today.

The same requirements map to concrete numerical bounds when applied to Ethereum's deployment context. Ciphertext size matters because the data must travel through either the execution payload or a data-availability blob, and the twelve-second Ethereum slot is further sub-divided under enshrined proposer-builder separation [21] into windows for proposer publication, attestation, payload reveal, and timeliness-committee voting, leaving only a fraction of the slot for cryptographic work. The HSW STE figures of 1.072 MB (classical elliptic-curve) and 1.431 MB (post-quantum) at one-megabyte plaintexts [37], per ciphertext in a non-batched scheme, sit an order of magnitude beyond the 128 KiB of a single EIP-4844 blob, so a hybrid design carrying them would need multi-blob payloads and proportionally longer propagation windows. Batched constructions such as BEAST-MEV [13] amortise differently: a per-ciphertext partial decryption of 48 bytes sits well inside a blob. The 3.2 seconds per partial decryption that the one-time-setup BTE of [18] reports at a 500-transaction batch parallelises across committee members and so carries a wall-clock cost bounded by single-member compute, but reconstruction is sequential after partial decryptions arrive and itself runs before the plaintext is available: 3.0 seconds for [18], and 129.5 seconds for the silent-setup BEAST-MEV as reported in Section 2.4. For an empirical reference scale, Mazorra, Öz, Schlegel, and Wu [47] report 6.9 million CEX-DEX searcher transactions across 1.9 million blocks containing relevant DEX trades between January 2024 and March 2025, an average of close to four CEX-DEX searcher trades in every block that carries a relevant DEX trade. The megabyte-scale ciphertexts and seconds-long decryption budgets reported in the literature must be evaluated against this scale: a handful of latency-sensitive, MEV-bearing trades in most blocks.

Storage of public-key material and the structured reference string is a second deployment constraint that receives limited attention in the cryptographic literature. The silent-setup construction of [33] has public keys that grow linearly in the committee size, an artefact

that Wagner and Ma [64] explicitly flag as disqualifying. For silent-setup schemes more broadly, the hints each validator must publish for public-key aggregation pose a coupled data-availability and integrity problem: Noyes and Policharla [51] estimate roughly three kilobytes of hints per validator at committee size 128, on the order of three gigabytes across Ethereum's roughly one million validators, all of which participating parties must retrieve and verify. The pairing-based BTE line [18, 13, 3, 55, 12] instead relies on a structured reference string that scales with the maximum batch size, produced by a one-time ceremony and persisted by every encrypting or decrypting node. The hybrid design of [64] addresses the free-option problem by routing the threshold ciphertext through a blob and conditioning the rest of the slot on attester sampling, binding the user to a ciphertext at submission time and keeping the cryptographic state off chain. This is the free option Mazorra et al. [47] identify for enshrined proposer-builder separation, where a party that can reveal or withhold a commitment after observing the market exercises an option whose value grows with volatility and the reveal window. Here the user holding the self-decryption key is the option holder.

Committee rotation is the security primitive on which the hybrid mempool design rests, and only the silent-setup line of constructions supports it without an interactive ceremony. The validator set on Ethereum rotates at every epoch boundary, and any scheme whose setup must be refreshed on rotation accumulates a ceremony cost that scales with the rotation frequency. Under proposer-builder separation [21], where a small set of well-resourced builders builds blocks and validators only attest, the cryptographic burden of partial-decryption verification, blob routing, and reconstruction can be concentrated on builders, who bear it within the payload-reveal window, without harming validator decentralisation.

Wallet and client integration is a third constraint, central to the deployment debate but absent from the cryptographic literature. Every proposal in this section introduces new transaction types, new commitment lifecycles, new key-release timing rules, and new failure modes that wallets and clients must surface to users as concrete UX consequences: longer time to inclusion, less predictable execution quality, complications for MEV refund flows, and a pay-for-envelope fee model in which the user is charged for the declared gas limit regardless of realised consumption, an arrangement that overcharges any transaction with a gas-limit buffer. These end-user costs, rather than the cryptographic ones, are what the Ethereum Magicians discussion of [25] has surfaced as the principal blockers to wallet adoption.

3.3 Shared blind spots

The community proposals agree on two requirements that follow from the realisation that no single cryptographic scheme suffices. The first is *liveness without coordination*, which the hybrid design of [64] achieves through user fallback and which the decryption-agnostic structures of [24, 45] achieve through trustless self-decryption. The second is *encryption agnosticism*, the explicit refusal to hard-code a single cryptographic scheme into the consensus layer so that the deployed scheme can evolve as the literature advances. EIP-8105 [45] formalises this by introducing a key-provider registry as a system contract, allowing any party to register a decryption provider that the protocol then treats uniformly.

A more contested question than what to enshrine is whether to enshrine the primitive at the protocol layer at all. The argument for in-protocol deployment, articulated by the Shutter team [60, 61] and by Ma [46], rests on three premises: (1) that out-of-protocol private mempools and trusted builder relationships have produced a centralisation drift that competition alone cannot unwind, (2) that visibility of pre-confirmation transaction content is the root cause of the malicious MEV that drives users towards those pipelines,

and (3) that only by-default protection at the consensus layer can break the dependency. The counter-position raised by wallet and protocol developers in the Magicians discussion of [25] is that the cryptographic constructions remain experimental, that the implementation burden on client teams is substantial, and that user demand at adoption time is uncertain, all of which favour continued off-protocol experimentation. The cryptographic literature surveyed in this paper is agnostic on this question, since every construction we examined is deployable either in-protocol or as an out-of-protocol service operated by a single committee, and the choice is thus driven by economics and governance rather than by cryptography.

The literature review assembled in this paper addresses the enshrinement question directly. By the Wagner-Ma criteria, no construction in the corpus jointly meets the four requirements: HSW STE sits one CCA gap short, BEAST-MEV one post-quantum gap short, and every other construction we examined fails on at least the silent-setup axis. The criteria were derived from a core protocol developer perspective and intended to capture what a deployable scheme should look like. By the community's own deployment criteria, enshrinement remains premature on cryptographic grounds today. Section 4 reinforces this preliminary conclusion by investigating a third *requirement*: the execution-quality cost that blind ordering imposes on MEV-bearing transactions.

The cryptographic literature treats blind ordering as a privacy feature rather than a fairness concern, since the builder learns nothing about the encrypted bodies while committing the order. In practice the builder retains broad ordering discretion that can be used to position encrypted transactions advantageously once the keys are released, and the community proposals develop different but uniformly partial mitigations, including top-of-block fee auctions in [24] and view-merge constraints on reveal timing in [62]. The corpus surveyed in Section 2 is largely silent on the question, except where the collision-free property removes the user-chosen slot index from the ciphertext. A related blind spot is *binary decryption*, namely that all schemes in the corpus treat the encrypted transaction as an all-or-nothing object that is either entirely sealed or entirely revealed. The frame-transaction structure underlying [62] hints at a richer design space in which some execution parameters can be hidden while others remain public, but no current cryptographic construction supports this directly.

The cost of these design choices is unequally distributed across transaction types. For simple transfers and similar low-contention payloads, blind ordering imposes no execution-quality cost, since their outcome does not depend on fresh and accurate state information about asset prices. The only consequence is an inclusion-latency overhead under the next-slot model, which is a UX cost rather than an economic one. For DeFi transactions and other interactions that touch contentious state, by contrast, outcomes depend on other transactions in the same block and on price movements on centralised exchanges that occur on a millisecond timescale. Here, latency translates directly into worse execution rather than into UX friction alone. Meanwhile, the leading community proposals have embraced blind ordering as a feature to combat toxic MEV. They differ only in how long they delay execution after commitment, ranging from the one-block delay of [24] to the same-slot reveal of [62]. We will now explore the implication this design choice has in forcing execution to take place over stale information.

4 Encrypted Mempools through the execution-quality lens

The cryptographic landscape in Section 2 and the protocol-developer perspective in Section 3 together establish that no current threshold-encrypted mempool construction jointly meets the four convergent properties the community has articulated. This section reinforces that

finding from a third direction, by examining what the design choices the community proposals share, namely blind ordering and binary decryption, cost users at the execution layer. *Even a cryptographically perfect threshold-encrypted mempool would impose costs the present proposals do not internalise.*

We organise the remainder of this section around two claims. First, blind ordering and binary decryption together do more than neutralise the front-running and sandwich strategies that motivated MEV mitigation in the first place: they eliminate the same-block positioning auctions that today route MEV value back to users, and degrade the backrun auctions that survive into next-slot execution over stale state. Order-flow auctions, surveyed by Gosselin and Chiplunkar [36], sit on top of the Flashbots Auction architecture [27] and are also delivered by MEV-Blocker and CoW Swap [19] for swap-specific flow. Yang et al. [66] show in an empirical study who in the supply chain captures their value, and Section 4.1 treats the concrete consequences: spam, validator centralisation, and DeFi execution quality. Second, the right design target is not full encryption from broadcast to ordering, but *programmable privacy* in the sense of Benarroch et al. [7], in which an enclave or a secure program can act on encrypted state before plaintext is revealed (Section 4.2).

4.1 The cost of blind ordering

The MEV literature distinguishes between toxic MEV, which extracts value from users without their consent through front-running and sandwich attacks [20], and non-toxic MEV, in which searchers compete to deliver execution improvements such as backrun arbitrage refunds. The two share an informational substrate: transaction content is visible during the ordering window. Blind ordering removes that visibility uniformly. Toxic and non-toxic strategies are suppressed together, since the schemes surveyed in Section 2 treat the encrypted payload as an atomic object that is either entirely sealed or entirely revealed, a property we will call *binary decryption*. The property attaches to the interface the current threshold-mempool schemes expose, not to threshold encryption as a primitive. However, the programmable directions surveyed in Section 4.2 show the primitive admits richer modes. There is no intermediate state in which an auction can run over hidden but operationally usable payload: the same-block family of mechanisms that today route MEV into user rebates is collapsed, and what survives must execute against state at least one slot stale.

4.1.1 Spam, validator centralisation and execution quality

The first concrete cost is *spam* incurred by blind ordering, and it appears in two related dynamics that are usually conflated. On one side, spam appears on-chain as reverting transactions that occupy block space, since builders cannot simulate encrypted payloads at ingress and so cannot filter reverts. On the other side, spam appears at the interface between builders and order-flow auction frontends as a flood of speculative bundles submitted by searchers who, in the absence of plaintext, can only guess at the user's intent.

On chains without an auction to channel searcher competition, that competition manifests as spam, a contemporary instance of the priority gas auction dynamic originally described by Daian et al. [20]: Miller [49] documents OP-Stack rollups where spam bots consume more than 50% of gas while paying less than 10% of fees, and Solana where MEV bots account for 40% of blockspace. Gogol et al. [35] report that more than 80% of reverted transactions on major Ethereum rollups are swaps, half targeting USDC/WETH pools on Uniswap v3 and v4. The equilibrium analysis of Mazorra et al. [48] brings a matching theoretical statement: under first-come-first-served sequencing with delayed observability of the target, searchers

compete by sending redundant transactions, and the number of transactions sent is the natural measure of cost. We revisit this connection between auctions and spam in Section 4.2. Removing expressiveness from searchers does not reduce their activity but channels it into spam.

Selective-disclosure systems sit between full plaintext and full encryption: Passerat-Palmbach and Wadhwa [54] formalise the trade-off on MEV-Share, analysing both a backrunning contract that reconstructs the trade on-chain at a gas premium and a brute-force search over the missing fields that floods the MEV-share node. Under blind ordering, equivalent to the hash-only tier that Flashbots Protect calls *full privacy mode*, in which only the transaction hash is shared [30], neither searcher channel is suppressed: the MEV-Share node has no plaintext to reveal selectively, simulation pre-inclusion is unavailable, and both spam channels are amplified. Garimidi et al. [34] reach the same conclusion from the cryptographic angle, observing that if fee data is itself encrypted, malformed ciphertexts cannot be filtered without an auxiliary zero-knowledge proof of well-formedness whose overhead is substantial.

Figure 1, drawn from a public Dune query on Flashbots Protect, is consistent with this trade-off. Within each of the two Flashbots Protect RPC endpoints (*default*, which routes to BuilderNet, a TEE-attested distributed builder network with approximately 25% block share, and *fast*, which routes to all registered builders), moving from a less informative hint tier to a more informative one cuts the p95 inclusion latency by approximately half: on *default* the hash-only to default-logs step tightens p95 from 55.4 to 27.2 seconds, and on *fast* the default config (hash + default_logs) to full hints (calldata + logs + contract_address + function_selector) step tightens p95 from 39.0 to 18.6 seconds. The median remains at one Ethereum slot (12 seconds) across all tiers, so the gain is entirely in the tail. The evidence is suggestive rather than causal: hint tiers are user-selected, so the latency differences may partly reflect differences in transaction type, fee, urgency, or MEV attractiveness across the users who choose each tier, the percentiles are conditional on inclusion, and whether richer hints also change the probability of landing at all cannot be answered from the public data. Across MEV-Share's lifetime, the same dataset shows roughly 85% of landed flow in the two tiers in which searchers operate effectively blind, either with the transaction hash alone or with the hash plus the fact that a swap occurred in a particular pool. An encrypted-mempool design committed to blind ordering forgoes that headroom entirely, and effectively behaves as the hash-only tier.

The second cost is *validator centralisation*. Today, MEV revenue reaches the validator set through MEV-Boost, which Yang et al. [66] document at the limit of a competitive market: two builders produced more than 85% of blocks at the time of their study, three builders today,[3] and the proposer losses they characterise accrue precisely because builders with privileged order flow need not bid the full extractable value to win. MEV mitigations slated for adoption rely on this builder market as an "MEV oracle" to price the lost revenue, and centralisation distorts that price. Under blind ordering, the entire mechanism that translates the builder market's pricing into validator rewards is suppressed. The competitive surface that remains for validators is ingress speed, which favours operators in a few low-latency locations and removes the principal subsidy that has kept solo and small-operator staking viable. The intervention designed against builder centralisation thus accelerates centralisation of the validator set instead, the very outcome proposer-builder separation was meant to avoid [15]. Rondelet and Kilbourn [57] raise the same structural point at the level of mempool privacy in general: interventions designed against a single MEV symptom reorganise the surrounding market in ways the cryptographic analysis does not anticipate.

[3] Per the public MEV-Boost dashboard at `https://mevboost.pics`, May 2026.

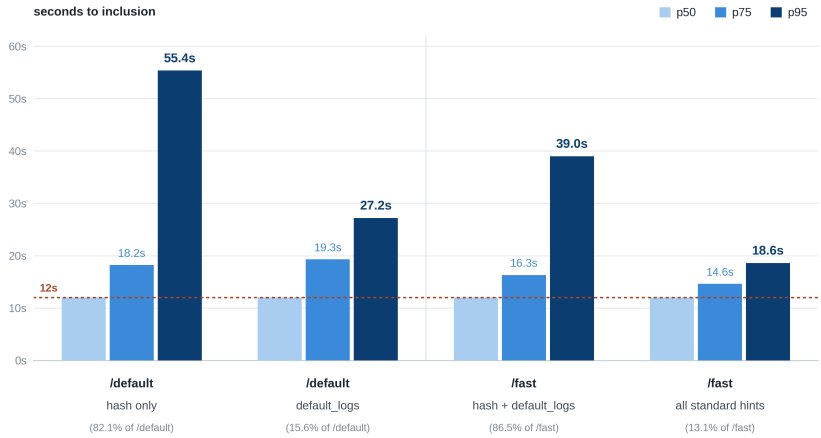


Figure 1 Within-endpoint inclusion latency by hint tier on Flashbots Protect, April 2023 to May 2026. Bars give p50, p75, and p95 of the time from submission to inclusion (seconds) for each (endpoint, hint tier) cell with at least 100k landed transactions, with bracketed values giving the tier's share of landed flow within that endpoint and the dashed line marking one Ethereum slot. The /default endpoint routes to BuilderNet (TEE-attested, ≈25% block share), and /fast routes to all registered builders (≈99% block share). Within each endpoint, the richer hint tier roughly halves p95 while p50 stays on the one-slot reference. Percentiles are conditional on inclusion. Tier shares within an endpoint do not sum to 100% because cells below the 100k-transaction threshold are omitted. Source: Dune query 3397259 (volume-weighted across monthly buckets).

The third cost is *execution quality* for DeFi transactions, and it has been quantified in previous studies. Kim and Akdeniz [44] measure reordering slippage on Uniswap v2 and v3 over September and October 2023 and report $9 million in user losses from intra-block position alone, with 72% of swaps and 92% of swaps over $250,000 negatively affected. Their measurement places most of the slippage beyond the first 10% of block positions, with the sharpest degradation between the 10% and 25% position marks, and they document up to a six-fold difference in execution price between top and bottom of block for the same trade. Execution outcomes for transactions that touch contentious state depend on fresh information about prices on centralised exchanges, which move on millisecond timescales, and on the position of the transaction within the block. Under blind ordering, the builder cannot place a transaction at the position that minimises slippage because the content is not yet visible. Under next-slot execution, an entire block has been finalised by the time the plaintext can be acted upon, and the execution decision is made against price information already at least one slot stale. Centralised execution venues keep compressing, not extending, the gap between price observation and trade settlement, Ethereum itself proposes to halve its slot time [1], and the leading community proposals for encrypted mempools move in the opposite direction.

4.1.2 Concrete impact on execution quality

Let's consider the example of a swap large enough for its execution price to depend on where it lands in the block, the regime Kim and Akdeniz [44] quantify above. Under the encrypted-mempool design of Elowsson in LUCID [24], the swap is committed in slot N and executed in slot $N+1$, so any backrun that would refund arbitrage value to the user runs at

slot $N + 1$ at the earliest, against state already twelve seconds stale to off-chain markets. The same-slot variant of Thiery [62] compresses the gap but still commits the routing path, the pools the swap traverses to minimise price impact, before the encrypted body is revealed. Under a TEE-mediated private mempool such as BuilderNet [28], the same swap is solved against fresh state inside the enclave, the backrun is placed in the same block, and the arbitrage value returns to the user as a rebate. The externality of the delayed price update extends beyond the user: during the twelve-second window in which the Uniswap pool price diverges from off-chain markets, other protocols that read the Uniswap pool as a price oracle execute against the stale price, and lending protocols that depend on those oracle reads may miss liquidations that the fresh price would have triggered. Recently, proprietary-AMM designs have gained traction on other chains, and to be viable on Ethereum L1, they would depend on the same builder access to fresh state to land repricing updates, which blind ordering rules out [50].

The analysis above leads to three conclusions the surveyed encrypted-mempool proposals do not resolve. (1) Blind ordering does not in fact eliminate toxic MEV: sophisticated attackers route around the protection by submitting attack legs across blocks, and wide sandwich bots such as the one identified in [22] succeed precisely because the attack does not need to bundle the frontrun and backrun in the same block, so an encrypted mempool that hides the payload but leaves the underlying execution model untouched would not prevent this class. (2) The proposals impose substantial engineering cost on the protocol and surrounding stack, since builders must be incentivised to include encrypted transactions while out-of-protocol private-mempool revenue remains available, and low end-user adoption would leave wallet and client ecosystems carrying technical debt while functionally similar out-of-protocol solutions deliver the same pre-trade privacy properties with better execution. (3) Decryption itself does not create new state transitions, but the execution of decrypted content does, and the timing gap between key release and execution opens MEV vectors that the proposals acknowledge but do not fully resolve. Taken together, these point to a design space in which auctions, not encryption alone, do the heavy lifting on MEV mitigation, and we turn to that design space in the next subsection.

4.2 The case for programmable privacy

The structural observation that follows from the execution-quality discussion is that running an auction *after* decryption is too late. By the time the plaintext is revealed, the ordering is fixed and the only remaining degree of freedom is the position of subsequent unencrypted transactions in the next block, against stale state. The auctions that today route MEV into user rebates require an intermediate computational mode in which an algorithm can route, simulate, and price against the encrypted payload while keeping the plaintext hidden from its operator. The cryptographic constructions in Section 2 do not provide that mode. They are designed to protect the plaintext until ordering is fixed and then release it, and the design is binary by intention.

The intent-based framing now common in the OFA literature sharpens this observation. The user expresses an intent, namely the outcome they want, rather than the transaction that realises it, and a solver competes for the right to translate that intent into execution at the best price. The chain-abstraction taxonomy of Chiplunkar and Gosselin [16] decomposes the value at stake into the part that depends only on ordering and the part that depends on observing the intent before settlement. An encrypted mempool can capture only the former, since whoever first sees the revealed transaction can act on it at the next slot; a private mempool captures both, precisely because its operator observes the intent in the clear, which

is the trust cost that motivated encryption in the first place. Programmable privacy is the design target that responds to both components together without reintroducing that trust. We use the term *programmable privacy*, in the sense of Benarroch et al. [7], for the alternative design target in which the encrypted state can be acted on by trusted code before plaintext revelation. Two technological substrates support programmable privacy today, with very different trust models.

The first is hardware-based via Trusted Execution Environments (TEEs). The Rollup-Boost system described by the Flashbots team [29] and the BuilderNet deployment in [28] both operate the block builder inside a TEE, with users encrypting their transactions to the enclave's public key. The enclave can simulate transactions against current state, filter reverts, place backruns in the same block, and refund the resulting MEV to the user, all without exposing plaintext to the operator of the hardware. The bottom-of-block searcher architecture of Bakhtin et al. [4] sits in the same family and is best understood as a hardware-backed analogue of selective-disclosure systems such as MEV-Share, in which the searcher receives the equivalent of full-reveal hints over the state changes induced by a partially built block. Here, the hardware boundary restricts the strategy the searcher can run on this state to the bottom-of-block backrun, ruling out frontrunning by construction. The trust assumption across these systems rests on the hardware supply chain rather than on a threshold of honest committee members, a substantively different security model that must be evaluated on its own terms. This shifts trust rather than enshrining new trusted parties: Ethereum already relies on a small set of builders, a concession accepted to keep the validator set decentralised, and the TEE model moves that reliance onto a comparatively disinterested ecosystem, the CPU manufacturers and their supply chain, who have no direct stake in transaction ordering. It carries a real residual cost worth clearly identifying for a fair comparison, since for Intel SGX and TDX a single vendor designs, fabricates, and attests the silicon, the security-critical microcode cannot be publicly audited, and the design tooling and advanced fabrication nodes rest with a handful of firms [43, 42]. Even a breach stays comparatively contained, however: it returns trust to the operator running the enclave, the same reliance on a known builder that private mempools already require.

The second substrate is software-based, namely programmable cryptography in the sense of fully homomorphic encryption and related primitives. Boneh, in a recent talk on the cryptographic frontier for encrypted mempools [8], sketches a long-horizon programme in which selective computation on encrypted transactions replaces the binary release of plaintext. The framework of Garg et al. [32] provides one unifying construction in the threshold setting, in the form of witness encryption for linearly verifiable relations. An otherwise intractable application such as an FHE-based backrun, demonstrated as a feasibility prototype by Passerat-Palmbach [53], would become viable as FHE hardware acceleration matures, with visible efforts in that direction [63, 68]. The performance penalty rules out deployment at Ethereum block-time scale today, but the constructions have improved substantially and the open problems are engineering blockers rather than fundamental.

The implication for the present deployment debate is that enshrining a binary-decryption threshold scheme today fixes the wrong design point. Even setting aside the cryptographic gap identified by Wagner and Ma [64], the design choice closes off the programmable-privacy direction without making the trade-off explicit. A more cautious position is to keep experiments running off-protocol, both on the TEE substrate where programmable privacy is already deployable, and on the cryptographic substrate where the building blocks are maturing, before fixing the interface at the consensus layer. The overall direction towards encrypted mempools is commendable, but it requires new designs that fully embrace the economic incentives that have proved pivotal in driving adoption in the space.

5 Open challenges

The literature review and the discussion of execution quality both point to a set of open problems that the cryptographic and protocol-development communities will need to address jointly. We close the paper by identifying four: post-quantum convergence, the crypto-economic gap, network-layer metadata leakage, and decryption-aggregation cost.

The first is *post-quantum security under the convergent design properties*. Table 1 contains one construction with a concrete post-quantum instantiation, the lattice-based scheme of Boneh et al. [10] in the batched-decryption line under a one-time DKG, and one modular blueprint instantiable post-quantum, the silent construction of Hall-Andersen et al. [37], which decrypts one ciphertext at a time rather than batching a block and reports megabyte-scale ciphertexts at one-megabyte plaintexts. The combination of silent setup, batched decryption, collision-free encoding, and post-quantum security under standard assumptions does not exist in the corpus to the best of our knowledge. The challenge is structural rather than incremental. The pairing-based BTE line owes its byte-scale ciphertexts and batch-independent decryption communication to algebraic properties of bilinear groups, and a post-quantum replacement would have to either reproduce those properties under lattice or isogeny assumptions or abandon one of the convergent properties. Designing a non-interactive thresholding protocol for post-quantum batched decryption remains an explicit open problem flagged by the authors of [10].

The second open problem is the gap between cryptographic and economic analysis this paper argues for. Rondelet and Kilbourn [57] argue that the impact of private mempools on markets and incentives renders an analysis omitting the economic lens incomplete. The discussion above instantiates that critique against the proposals on the table, but the broader question of how to design encrypted-mempool primitives that internalise execution quality as a first-class objective is open. None of the constructions in Section 2 models the economic value of the plaintext as part of the security definition, and the security games are written purely in terms of what stays encrypted. A design blueprint that takes the economic lens as a constraint rather than as a downstream evaluation is yet to be articulated.

The third open problem is *metadata leakage at the network layer*. Cryptographic mempools hide the transaction payload but not the network-layer metadata that wraps it, including the IP address of the broadcaster, the timing of the broadcast, the size of the ciphertext, and the position of the ciphertext within a batch. Kavousi et al. [41] formalise the problem and propose BlindPerm, a protocol-layer mechanism that piggybacks on an encrypted mempool to shuffle transactions before execution without an external randomness beacon, at a small constant-factor overhead. BlindPerm addresses the in-protocol permutation question, which is orthogonal to the broader network-layer problem of hiding the broadcaster's identity. The anonymous-broadcast direction targets the latter: the ZipNet construction of Rosenberg et al. [58] uses trusted execution environments for liveness and classical cryptography for anonymity, and the Panetière design of [31] pursues the same TEE-assisted anonymous-broadcast direction with guaranteed output delivery, remaining live with up to half of the servers arbitrarily corrupt. Integrating these network-layer constructions with the threshold-encrypted mempool corpus is an open architectural problem.

The fourth open problem is the *decryption-aggregation cost in epochless schemes*. Epoch-bound schemes make the user commit each ciphertext to a target block, but the user cannot reliably predict the landing slot, so a meaningful fraction miss their target and become undecryptable, and the slot-collision attack of [14] recreates the censorship vector encryption was meant to remove. Epochless schemes shift the work from the user to the committee,

which aggregates partial decryptions across a batch on demand. The key-homomorphic puncturable PRF construction of Bormet et al. [14] requires $O(B^2)$ pairings to aggregate at batch size B. The FFT-in-the-exponent decryption of Agarwal et al. [2] first brought the cost to $O(B \log B)$, in the weighted setting, and the partial-fraction technique of Boneh et al. [12] together with the punctured-powers constructions of [55, 3] reach the same quasi-linear cost in the collision-free setting. The most recent numbers, from BTX [3], put end-to-end decryption at $B = 512$ at roughly 600 ms on a single core against 1,200 ms for the partial-fraction baseline of [12], both on BTX's AVX-512 C++ codebase; Policharla's earlier evaluation (Section 2.5) is about twice as large with the same ordering. Viable batch sizes today therefore sit in the few hundreds. Finding an epochless BTE scheme with $O(B)$ decryption-aggregation cost is an explicit open problem, necessary to lift the batch-size ceiling toward L2 throughput targets of thousands of transactions per slot. The post-quantum setting compounds the challenge, since the partial-fraction and FFT techniques relied on by current solutions exploit pairing structure that does not transfer to lattice constructions.

6 Conclusion

The convergent properties our taxonomy identifies, silent setup, batched decryption, and epochless collision-free encoding, describe a design point optimised for a binary-decryption interface that this paper argues is the wrong target. The cryptographic gaps and the economic costs compound rather than conflict: closing the four-property gap leaves blind ordering in place, and a TEE-based programmable-privacy alternative captures the auctions only by shifting trust from the cryptographic committee to the hardware supply chain. The four open problems we identify sit ahead of any deployable design, so the prudent path is off-protocol experimentation: on the TEE substrate where programmable privacy is deployable today, and on the cryptographic substrate where the building blocks are maturing. Premature enshrinement would cost more than a privacy primitive that misses its target and fresh technical debt for wallet and client teams. Above all, it would disrupt the same-block auctions that today route value back to users.

References

1 Ben Adams, Dankrad Feist, Maria Inês Silva, and Paul Harris. EIP-7782: Reduce block latency. Ethereum Improvement Proposals, 2024. Case post: `https://ethereum-magicians.org/t/eip-7782-the-case-for-2x-shorter-slot-times-in-glamsterdam/24616`. URL: `https://eips.ethereum.org/EIPS/eip-7782`.

2 Amit Agarwal, Kushal Babel, Sourav Das, Babak Poorebrahim Gilkalaye, Arup Mondal, Benny Pinkas, Peter Rindal, and Aayush Yadav. Weighted batched threshold encryption with applications to mempool privacy. Cryptology ePrint Archive, Paper 2025/2115, November 2025. To appear, IEEE S&P 2026. URL: `https://eprint.iacr.org/2025/2115`.

3 Amit Agarwal, Sourav Das, Babak Poorebrahim Gilkalaye, Peter Rindal, and Victor Shoup. BTX: Simple and efficient batch threshold encryption. Cryptology ePrint Archive, Paper 2026/754, 2026. URL: `https://eprint.iacr.org/2026/754`.

4 Artyom Bakhtin, Frieder Erdmann, Angela Lu, Moe Mahhouk, and Mateusz Morusiewicz. Searching in TDX. Flashbots Collective forum, September 2024. Describes the "bob" (bottom-of-block) construction. URL: `https://collective.flashbots.net/t/searching-in-tdx/3902`.

5 Joseph Bebel and Dev Ojha. Ferveo: Threshold decryption for mempool privacy in BFT networks. Cryptology ePrint Archive, Paper 2022/898, 2022. URL: `https://eprint.iacr.org/2022/898`.

6 Amos Beimel. Secret-sharing schemes: A survey. In *Coding and Cryptology (IWCC 2011)*, volume 6639 of *Lecture Notes in Computer Science*, pages 11–46. Springer, 2011. doi:10.1007/978-3-642-20901-7_2.

7 Daniel Benarroch, Bryan Gillespie, Ying Tong Lai, and Andrew Miller. SoK: Programmable privacy in distributed systems. Cryptology ePrint Archive, Paper 2024/982, 2024. URL: https://eprint.iacr.org/2024/982.

8 Dan Boneh. Cryptography for encrypted mempools. a16z crypto research talks, YouTube, November 2025. URL: https://www.youtube.com/watch?v=0NNDX63lySQ.

9 Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer, 2005. doi:10.1007/11535218_16.

10 Dan Boneh, Evan Laufer, and Ertem Nusret Tas. Batch decryption without epochs and its application to encrypted mempools. Cryptology ePrint Archive, Paper 2025/1254, 2025. URL: https://eprint.iacr.org/2025/1254.

11 Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, 2004. Preliminary version in ASIACRYPT 2001, LNCS 2248, pp. 514–532. doi:10.1007/s00145-004-0314-9.

12 Dan Boneh, Rohit Nema, Arnab Roy, and Ertem Nusret Tas. Efficient batch threshold encryption using partial fraction techniques. Cryptology ePrint Archive, Paper 2026/674, 2026. URL: https://eprint.iacr.org/2026/674.

13 Jan Bormet, Arka Rai Choudhuri, Sebastian Faust, Sanjam Garg, Hussien Othman, Guru-Vamsi Policharla, Ziyan Qu, and Mingyuan Wang. BEAST-MEV: Batched threshold encryption with silent setup for MEV prevention. Cryptology ePrint Archive, Paper 2025/1419, 2025. URL: https://eprint.iacr.org/2025/1419.

14 Jan Bormet, Sebastian Faust, Hussien Othman, and Ziyan Qu. BEAT-MEV: Epochless approach to batched threshold encryption for MEV prevention, August 2025. URL: https://www.usenix.org/conference/usenixsecurity25/presentation/bormet.

15 Vitalik Buterin. Proposer/block builder separation-friendly fee market designs. ethresear.ch, June 2021. URL: https://ethresear.ch/t/proposer-block-builder-separation-friendly-fee-market-designs/9725.

16 Ankit Chiplunkar and Stephane Gosselin. Introducing the CAKE framework. Frontier Research, February 2024. URL: https://frontier.tech/the-cake-framework.

17 Arka Rai Choudhuri, Sanjam Garg, Julien Piet, and Guru-Vamsi Policharla. Mempool privacy via batched threshold encryption: Attacks and defenses. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 3513–3529, Philadelphia, PA, 2024. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/choudhuri.

18 Arka Rai Choudhuri, Sanjam Garg, Guru-Vamsi Policharla, and Mingyuan Wang. Practical mempool privacy via one-time setup batched threshold encryption. Cryptology ePrint Archive, Paper 2024/1516, 2024. To appear, USENIX Security 2025. URL: https://eprint.iacr.org/2024/1516.

19 CoW DAO. MEV Blocker. cow.fi. Accessed 2026. URL: https://cow.fi/mev-blocker.

20 Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash Boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 910–927, 2020. doi:10.1109/SP40000.2020.00040.

21 Francesco D'Amato, Nico Flaig, Barnabé Monnot, Michael Neuder, Potuz, Justin Traglia, and Terence Tsao. EIP-7732: Enshrined proposer-builder separation. Ethereum Improvement Proposals (draft), June 2024. Created 28 June 2024; status: draft. Accessed 2026-05-22. URL: https://eips.ethereum.org/EIPS/eip-7732.

22 Dataalways. Twitter thread on the 0x841 wide-sandwich bot. Twitter/X, 2024. Pseudonymous author; archival via web.archive.org recommended before final submission. URL: https://x.com/dataalways/status/2047693168791928906.

23 Stefan Dziembowski, Sebastian Faust, and Jannik Luhn. Shutter network: Private transactions from threshold cryptography. Cryptology ePrint Archive, Paper 2024/1981, 2024. URL: https://eprint.iacr.org/2024/1981.

24 Anders Elowsson. LUCID: Encrypted mempool with distributed payload propagation. Ethereum Research forum (ethresear.ch), February 2026. Posted 5 February 2026. Accessed 2026-05-22. URL: https://ethresear.ch/t/lucid-encrypted-mempool-with-distributed-payload-propagation/24042.

25 Anders Elowsson, Julian Ma, and Justin Florentine. Hegota headliner: LUCID encrypted mempool. Ethereum Magicians forum (ethereum-magicians.org), February 2026. Posted 5 February 2026. Accessed 2026-05-22. URL: https://ethereum-magicians.org/t/hegota-headliner-lucid-encrypted-mempool/27658.

26 Rex Fernando, Guru-Vamsi Policharla, Andrei Tonkikh, and Zhuolun Xiang. TrX: Encrypted mempools in high performance BFT protocols. Cryptology ePrint Archive, Paper 2025/2032, 2025. URL: https://eprint.iacr.org/2025/2032.

27 Flashbots. Flashbots Auction overview. Online documentation. Accessed 2026. URL: https://docs.flashbots.net/flashbots-auction/overview.

28 Flashbots. Introducing BuilderNet. Flashbots / BuilderNet, November 2024. Decentralised, TEE-based collaborative block-building network for Ethereum L1. Migration write-up at https://writings.flashbots.net/migrating-to-buildernet. URL: https://buildernet.org/blog/introducing-buildernet.

29 Flashbots. Introducing Rollup-Boost: Launching on Unichain. Flashbots Writings, October 2024. URL: https://writings.flashbots.net/introducing-rollup-boost.

30 Flashbots. Flashbots protect settings guide. Flashbots documentation, 2025. Accessed 2026-08-08. URL: https://docs.flashbots.net/flashbots-protect/settings-guide.

31 Nils Fleischhacker, Mateusz Morusiewicz, and Mark Simkin. Panetière: Efficient anonymous broadcast. Draft, Flashbots, July 2026. Formerly known as Flashnet. Accessed 2026-08-07. URL: https://net.flashbots.net/panetiere-draft.pdf.

32 Sanjam Garg, Mohammad Hajiabadi, Dimitris Kolonelos, Abhiram Kothapalli, and Guru-Vamsi Policharla. A framework for witness encryption from linearly verifiable SNARKs and applications. Cryptology ePrint Archive, Paper 2025/1364, 2025. CRYPTO 2025. URL: https://eprint.iacr.org/2025/1364.

33 Sanjam Garg, Dimitris Kolonelos, Guru-Vamsi Policharla, and Mingyuan Wang. Threshold encryption with silent setup. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part VII*, volume 14926 of *Lecture Notes in Computer Science*, pages 352–386, Cham, 2024. Springer. Also available as Cryptology ePrint Archive Paper 2024/263, https://eprint.iacr.org/2024/263. doi:10.1007/978-3-031-68394-7_12.

34 Pranav Garimidi, Joseph Bonneau, and Lioba Heimbach. On the limits of encrypted mempools. a16z crypto, 2025. URL: https://a16zcrypto.com/posts/article/limits-encrypted-mempools/.

35 Krzysztof M. Gogol, Manvir Schneider, and Claudio J. Tessone. When priority fails: Revert-Based MEV on fast-finality rollups, 2025. arXiv:2506.01462.

36 Stephane Gosselin and Ankit Chiplunkar. The orderflow auction design space. Frontier Research, March 2023. URL: https://frontier.tech/the-orderflow-auction-design-space.

37 Mathias Hall-Andersen, Mark Simkin, and Benedikt Wagner. Silent threshold encryption with one-shot adaptive security. Cryptology ePrint Archive, Paper 2025/1384, 2025. URL: https://eprint.iacr.org/2025/1384.

38 Amir Herzberg, Stanisław Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In Don Coppersmith, editor, *Advances in Cryptology – CRYPTO 1995*, volume 963 of *Lecture Notes in Computer Science*, pages 339–352. Springer, 1995. Cite key includes "jakobsson"; PSS 1995 author list is HJKY only. If you intended the related Herzberg-Jakobsson-Jarecki-Krawczyk-Yung paper "Proactive Public Key and Signature Systems" (ACM CCS 1997), swap this entry. doi:10.1007/3-540-44750-4_27.

39 Charanjit S. Jutla, Rohit Nema, and Arnab Roy. Partial fraction techniques for cryptography. In *Advances in Cryptology – EUROCRYPT 2026*, Lecture Notes in Computer Science, pages 423–453. Springer, 2026. Cryptology ePrint Archive, Paper 2025/2081. doi:10.1007/978-3-032-25330-9_15.

40 Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 177–194, Berlin, Heidelberg, 2010. Springer. doi:10.1007/978-3-642-17373-8_11.

41 Alireza Kavousi, Duc V. Le, Philipp Jovanovic, and George Danezis. BlindPerm: Efficient MEV mitigation with an encrypted mempool and permutation. In *29th International Conference on Principles of Distributed Systems (OPODIS 2025)*, volume 361 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 36:1–36:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.OPODIS.2025.36.

42 Quintus Kilbourn, Sylvain Bellemare, Andrew Huang, and Michael Gao. ZTEE: Trustless supply chains. Flashbots Writings, November 2024. Accessed 2026-08-08. URL: https://writings.flashbots.net/ZTEE2-Supply-Chains.

43 Quintus Kilbourn, Sylvain Bellemare, Jonathan Passerat-Palmbach, and Andrew Miller. Zero trust execution environments. Flashbots Writings, October 2024. Accessed 2026-08-08. URL: https://writings.flashbots.net/ZTEE.

44 Evan Kim and Murat Akdeniz. Trading in the dark: Measuring reordering slippage in MEV. Primev, in collaboration with Frontier Research, January 2024. URL: https://frontier.tech/measuring-reordering-slippage-in-mev.

45 Jannik Luhn. EIP-8105: Universal enshrined encrypted mempool. Ethereum Improvement Proposals (draft), December 2025. Created 17 December 2025; status: draft. Accessed 2026-05-22. URL: https://eips.ethereum.org/EIPS/eip-8105.

46 Julian Ma. Why Ethereum needs an encrypted mempool. Julian's Research (Substack), March 2026. Posted 12 March 2026. Accessed 2026-05-22. URL: https://julianma.substack.com/p/why-ethereum-needs-an-encrypted-mempool.

47 Bruno Mazorra, Burak Öz, Christoph Schlegel, and Fei Wu. The free option problem of ePBS, 2025. Submitted 29 September 2025; last revised 8 January 2026 (v3). doi:10.48550/arXiv.2509.24849.

48 Bruno Mazorra, Christoph Schlegel, and Akaki Mamageishvili. Timing games: Probabilistic backrunning and spam, 2026. doi:10.48550/arXiv.2602.22032.

49 Robert Miller. MEV and the limits of scaling. Flashbots Writings, June 2025. URL: https://writings.flashbots.net/mev-and-the-limits-of-scaling.

50 Mike Neuder and Maryam Bahrani. Proprietary AMMs and Ethereum. Ethereum Research forum (ethresear.ch), July 2026. Accessed 2026-08-08. URL: https://ethresear.ch/t/proprietary-amms-and-ethereum/25543.

51 Charlie Noyes and Guru Vamsi Policharla. How to remove the relay. Paradigm Research, October 2024. Posted 8 October 2024. Accessed 2026-05-22. URL: https://www.paradigm.xyz/2024/10/removing-the-relays.

52 Vabuk Pahari and Andrea Canidio. How exclusive are Ethereum transactions? Evidence from non-winning blocks, 2025. Submitted 19 September 2025. doi:10.48550/arXiv.2509.16052.

53 Jonathan Passerat-Palmbach. Leveraging homomorphic encryption for maximally extractable value (MEV) mitigation: Enabling blind arbitrage on decentralised exchanges. In *Financial Cryptography and Data Security 2025*, 2025. Paper 238 in preproceedings. URL: https://fc25.ifca.ai/preproceedings/238.pdf.

54 Jonathan Passerat-Palmbach and Sarisht Wadhwa. Differentially private aggregate hints in MEV-Share, 2025. doi:10.48550/arXiv.2508.14284.

55 Guru-Vamsi Policharla. A simple batched threshold encryption scheme. Cryptology ePrint Archive, Paper 2026/760, 2026. URL: https://eprint.iacr.org/2026/760.

56 Leonid Reyzin, Adam D. Smith, and Sophia Yakoubov. Turning HATE into LOVE: Compact homomorphic ad hoc threshold encryption for scalable MPC. In Shlomi Dolev, Oded Margalit, Benny Pinkas, and Alexander Schwarzmann, editors, *Cyber Security Cryptography and Machine Learning (CSCML 2021)*, volume 12716 of *Lecture Notes in Computer Science*, pages 361–378, Cham, 2021. Springer. Also available as Cryptology ePrint Archive Paper 2018/997, `https://eprint.iacr.org/2018/997`. `doi:10.1007/978-3-030-78086-9_27`.

57 Antoine Rondelet and Quintus Kilbourn. Mempool privacy: An economic perspective, 2023. `arXiv:2307.10878`.

58 Michael Rosenberg, Maurice Shih, Zhenyu Zhao, Rui Wang, Ian Miers, and Fan Zhang. ZIPNet: Low-bandwidth anonymous broadcast from (dis)Trusted Execution Environments. Cryptology ePrint Archive, Paper 2024/1227, 2024. Published version: Proceedings on Privacy Enhancing Technologies, 2025(2):211–225, `https://doi.org/10.56553/popets-2025-0058`. URL: `https://eprint.iacr.org/2024/1227`.

59 Caspar Schwarz-Schilling, Fahad Saleh, Thomas Thiery, Jennifer Pan, Nihar Shah, and Barnabé Monnot. Time is money: Strategic timing games in proof-of-stake protocols. In *5th Conference on Advances in Financial Technologies (AFT 2023)*, volume 282 of *LIPIcs*, pages 30:1–30:17, 2023. `doi:10.4230/LIPIcs.AFT.2023.30`.

60 Shutter Network. Introducing EIP-8105 universal enshrined encrypted mempool. Shutter Blog, December 2025. Posted 17 December 2025. Accessed 2026-05-22. URL: `https://blog.shutter.network/introducing-the-universal-enshrined-encrypted-mempool-eip/`.

61 Shutter Network and Tatu. Why encryption is the only real protection against MEV. Shutter Blog, May 2025. Posted 15 May 2025. Accessed 2026-05-22. URL: `https://blog.shutter.network/why-encryption-is-the-only-real-protection-against-mev/`.

62 Thomas Thiery. Encrypted frame transactions. Ethereum Research forum (ethresear.ch), March 2026. Posted 18 March 2026. Accessed 2026-05-22. URL: `https://ethresear.ch/t/encrypted-frame-transactions/24440`.

63 Michiel Van Beirendonck, Jan-Pieter D'Anvers, Furkan Turan, and Ingrid Verbauwhede. FPT: A fixed-point accelerator for torus fully homomorphic encryption. *arXiv preprint*, 2022. Also published at CCS 2023. `doi:10.48550/arXiv.2211.13696`.

64 Benedikt Wagner and Julian Ma. Hybrid encrypted mempools. Ethereum Research forum (ethresear.ch), October 2025. Posted 27 October 2025. Accessed 2026-05-22. URL: `https://ethresear.ch/t/hybrid-encrypted-mempools/23360`.

65 Brent Waters and David J. Wu. Silent threshold cryptography from pairings: Expressive policies in the plain model. In *Advances in Cryptology – EUROCRYPT 2026, Part V*, volume 16545 of *Lecture Notes in Computer Science*, pages 517–547. Springer, 2026. Cryptology ePrint Archive, Paper 2025/1547. `doi:10.1007/978-3-032-25330-9_18`.

66 Sen Yang, Kartik Nayak, and Fan Zhang. Decentralization of Ethereum's builder market, 2024. `doi:10.48550/arXiv.2405.01329`.

67 Sen Yang, Burak Öz, Fei Wu, and Fan Zhang. Geographical centralization resilience in Ethereum's block-building paradigms. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 10(2):39:1–39:42, 2026. Presented at ACM SIGMETRICS 2026. Preprint: arXiv:2509.21475. `doi:10.1145/3805637`.

68 Zama. A systemVerilog implementation of the Homomorphic Processing Unit (HPU) targeting AMD Alveo V80 FPGA board, 2025. URL: `https://github.com/zama-ai/hpu_fpga`.

RANDAO Manipulation in the Presence of MEV

Kaya Alpturer
Princeton University, NJ, USA

Nicholas Hope
Talarion, San Francisco, CA, USA

S. Matthew Weinberg
Princeton University, NJ, USA

Abstract

Ethereum's randomness beacon (RANDAO) is well-known to be manipulable, and prior work [2] computes the precise fraction of blocks a strategic proposer can propose. The fraction of blocks proposed, however, is only a proxy for participants' rewards. We propose a generalized reward model capturing many canonical forms of rewards: consensus reward rollover, multi-block MEV, CEX-DEX arbitrage, oracle manipulation, and others. We provide a methodology that computes an ε-optimal strategy for any reward scheme in our model (and in particular, any combination of the above rewards). Finally, we apply our methodology to several canonical examples, and establish the sensitivity of RANDAO manipulation to the underlying rewards. We find that if rewards partially roll over, or scale super-linearly with consecutive blocks, the incentive to manipulate RANDAO is amplified. Lastly, we investigate tail-slot slashing, which can be modeled as a reward function, and show that honest equilibria can be recovered.

2012 ACM Subject Classification Theory of computation → Algorithmic game theory and mechanism design; Information systems → Digital cash; Security and privacy → Distributed systems security

Keywords and phrases Proof of Stake, Consensus, Blockchain, Ethereum, Randomness manipulation

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.23

Supplementary Material *Software*: `https://github.com/kalpturer/randao-mev`
archived at `swh:1:dir:5fc25f28e664c64535e1f20e516a4f7ca0e47e76`

Funding This work was supported by an Ethereum Foundation grant.

Acknowledgements During Prof. Weinberg's development of this paper, he participated as an expert witness on behalf of the State of Texas in ongoing litigation against Google (the "Google Litigation").

1 Introduction

A major concern for blockchain protocols is the potential for economically rational participants to deviate from the protocol specification to maximize their rewards. In many protocols, randomness is used to ensure the fair distribution of block proposing rights. Thus, if the randomness in a protocol is not robust, deviations that bias it will be incentivized. In Bitcoin, protocol deviations such as selfish mining have relied mainly on forking and hiding blocks, as biasing the randomness requires breaking the robust SHA256 hash function. In Proof-of-Stake protocols, however, the randomness is often provided by a distributed randomness beacon built into the protocol itself, creating a novel attack vector. In particular, Ethereum's implementation of randomness, called RANDAO, has been well known to be vulnerable to block withholding attacks [2]. Given RANDAO's role in determining the winners of the block proposal lottery, there is great financial incentive to manipulate it.


© Kaya Alpturer, Nicholas Hope, and S. Matthew Weinberg;
licensed under Creative Commons License CC-BY 4.0



With Ethereum's stake pool at roughly $80 billion, the concern for RANDAO manipulation has resulted in several lines of work, analyzing the potential impacts of this type of deviation [2, 20, 27, 28]. These works establish that an adversary with an α fraction of the total stake is able to propose strictly more than an α fraction of the total blocks, and therefore earn greater than an α fraction of the total consensus rewards.

However, rewards in Ethereum have increasingly been dominated by execution layer rewards that stem from decentralized finance activity as opposed to constant rewards per block proposed. We have seen, for example, that sophisticated actors are able to extract rewards from arbitrage, front-running, and transaction fees [11, 21]. Therefore, there is a need to model rewards more generally to include higher-order reward properties beyond a single block.

The importance of varying rewards is well-established in the proof-of-work setting. For Bitcoin, [10] shows that variance in transaction fees fundamentally alters the equilibrium structure of selfish mining, with subsequent work generalizing the analyses to richer reward models [6, 7]. For RANDAO manipulation, however, the analogous question of how MEV and similar non-uniform reward structures affect the incentive to manipulate it has remained open. While prior work laid the foundation for the MDP-based analysis and formalization of the larger problem, these analyses mainly measure the fraction of blocks proposed.

Our primary contribution is a sampling-based methodology to compute ε-optimal policies for RANDAO manipulation for significantly more general reward models. Our approach relies on a novel truncation of the RANDAO game that can provably provide bounds on the optimal solution. Prior work [2] relies critically on uniform rewards to collapse the state space, and our truncation and sampling-based approach requires a new argument for bounding the optimal manipulation.

As a warmup exercise, we show that our methodology reproduces the results of [2] in the case of uniform rewards (single unit of reward per proposed block). Our approximate solver is able to more efficiently compute approximate results, and larger runs with sufficient samples are capable of achieving progressively more accurate results. We then investigate several reward models relevant to transaction fees, arbitrage, oracle manipulation, and consensus rewards. In the process, we characterize specific types of MEV and their interactions with RANDAO manipulation, including *per-block rewards (with partial rollover)*, and two models of *multi-block MEV*.

1.1 Overview of Methodology

Below, we overview several instances of execution layer rewards, and how they fit into our methodology. Recall that RANDAO proceeds as follows. Every time a proposer proposes a block during their slot, RANDAO changes, becoming a new value that is: (a) deterministic, to the proposer, (b) appears uniformly random, to anyone without the proposer's private key. At the end of every 32-block epoch, the next 32 proposers are selected according to the current RANDAO value.[1] Therefore, when proposers are selected for the last k blocks of an epoch, they can selectively propose/skip slots to select among 2^k independent random options for the next epoch's 32 proposers. The cost of doing so is whatever rewards are sacrificed from skipped blocks. The benefit of doing so is the potential for more rewards (and perhaps more opportunities for future manipulation) in the next epoch. We call the largest such k controlled by an adversary the *tail*.

[1] In the actual Ethereum implementation, the RANDAO from epoch i determines the proposers for epoch $i + 2$. We ignore interleaving and discuss its effect in the model assumptions.

When discussing sample rewards below, we use Tail to refer to the pattern of proposed/skipped slots by the adversary in the tail of an epoch and Next for the pattern of proposed slots by the adversary in the non-tail. Assuming that other proposers honestly follow the intended protocol (and propose in every slot), a strategic proposer who does not control the last slot of an epoch optimizes their expected rewards by proposing in every non-tail slot. This is simply because: (a) such a proposer has no control over the next epoch's RANDAO, (b) a proposer seeking to optimize rewards in this epoch alone can always replace a skipped block with an empty block without affecting the execution rewards and still collect block rewards.

▶ **Example 1** (Reward Rollover)**.** Many execution layer rewards, such as those from on-chain arbitrage and sandwich attacks, roll over if not claimed in a block. For instance, an MEV opportunity in the mempool simply becomes available to the next proposer after a missed block.[2] This also occurs with consensus layer rewards. The block reward increases after a miss due to the increase in the number of attestations that can be included (see Figure 6). Intuitively, this rollover helps RANDAO manipulation by reducing the cost of missing a block if the adversary knows that they will get to propose after a miss. Because this rollover can be represented as a function on Tail and Next, we investigate its effect in our per-slot reward model.

▶ **Example 2** (Arbitrage Value)**.** For an adversary controlling k consecutive blocks, a single arbitrage at the end is more profitable than multiple arbitrage transactions due to trading fees [19]. Fees introduce a profitability threshold, giving the arbitrage an option-like payoff.[3] Controlling a longer sequence of k blocks increases the probability that the price will drift beyond this threshold. Thus, a validator intending to perform CEX-DEX arbitrage values a run of k consecutive blocks more than k independent blocks, motivating a superlinear model for MEV extraction. In particular, observe again that the reward earned by the proposer can be written as a function on Tail and Next.

▶ **Example 3** (Oracle Manipulation)**.** Manipulation of time-weighted average price (TWAP) oracles in order to force illegitimate liquidations is a well-understood form of multi-block MEV [18]. While oracle manipulation of this form is already a concern without an adversary controlling multiple blocks in a row[4], RANDAO manipulation exacerbates the threat by allowing an adversary complete control over transaction inclusion, reducing the attack cost. For a given adversary, TWAP attacks represent a model in which a large, fixed value (the liquidation profit) is available if the validator can secure enough sequential blocks to initiate the liquidation. As such rewards can be written as a function on Tail and Next, we explore this further in the threshold multi-block MEV model.

---

[2] See Figure 7 for the distribution of MEV bids on the MEV-Boost auction showing the distribution of these rewards.

[3] For instance, consider a scenario in which DEX and CEX prices begin perfectly equal, with a de minimis fee parameter on the CEX but a 30 basis point fee on the DEX. In such a scenario, a price discrepancy of more than 30 basis points must arise via price movement on the CEX before a validator may execute an arbitrage transaction profitably. Reserving sole arbitrage rights via multi-slot control thus exhibits the same nonlinear payoff function as a put or call option: the opportunity is worthless until the option becomes in the money (that is, the price discrepancy exceeds cumulative fee parameters of the target CEX-DEX pair).

[4] Prior work [18] primarily considers block proposers that include the attacker's transactions, but also transactions from arbitrageurs that de-manipulate asset prices. Such a model remains appropriate in a setting where the sequence of proposers cannot be guaranteed in advance, such as Proof-of-Work in contrast to Ethereum.

Our Methodology. All of the above examples can be captured by specifying two functions $f(\cdot)$ and $g(\cdot)$. $f(\cdot)$ takes as input a sequence of proposals by the adversary in the non-tail part of the epoch (which we refer to as NEXT), and outputs "under the specified reward model, what is the expected reward for the adversary if the adversary proposes all of their blocks and honest validators propose the other blocks?" $g(\cdot)$ takes as input a fully specified TAIL sequence and outputs "under the specified reward model, what is the expected reward for the adversary if the adversary proposes the specified blocks and misses the others in TAIL?" Note the distinction between the two: in NEXT the non-proposed blocks are proposed by an honest party and in TAIL non-proposed blocks are missed. As we later explore, this may make a difference if honestly proposed blocks affect rewards of other blocks. For *any* reward model that can be fully specified by some $f(\cdot)$ and $g(\cdot)$ (including all three examples above, and any combination of them), our methodology computes an ε-optimal strategy for a strategic proposer. In the formal model of Section 2, f corresponds to $\mathsf{next}(\cdot)$ and g corresponds to $\mathsf{propose}(\cdot)$.

Highlights. In addition to developing our methodology, we also apply it to explore canonical settings and draw conclusions about the role of execution rewards in RANDAO incentives. First, we reproduce the results of [2] in the case of uniform rewards (constant per block), and show that our approximate solver can quickly reproduce these results at coarse ε. We then demonstrate the methodology on three execution-layer reward models: per-slot rollover, a smooth multi-block model, and a threshold multi-block model. Consensus-reward rollover is directly visible in the data (Figure 6), but MEV rollover is harder to verify empirically: slot misses are rare on Ethereum, so the available observational data on what happens to MEV opportunities across a missed slot is scarce. Systematic multi-block MEV is also weak in current Ethereum [26], so the smooth and threshold models are stylized. The modelled MEV regimes may matter in the future under block-building redesigns. In the regimes we model, partial reward rollover and multi-block MEV both amplify the incentive to manipulate RANDAO.

Lastly, we focus on two practically relevant investigations using the same framework. We show how increasing the epoch length decreases the manipulation gains across reward models[5]. We then encode, as a reward function, a tail-slashing curve that penalizes misses toward the tail more heavily and amplifies penalties for groups of consecutive misses, and show that honest behavior can be made ε-optimal under appropriate slashing magnitudes.

1.2 Related Work

RANDAO Manipulation. Since its inception, RANDAO has been known to be manipulable to a certain degree by withholding tail slots [13, 28]. Early probabilistic modeling based work [3] shows that this manipulation is small but non-negligible. [2] develops a Markov Decision Process (MDP) formulation of RANDAO manipulation to compute optimal strategies. Later work extends this analysis to consider forking [20, 27]. Aside from RANDAO, prior work also considers manipulation of other proof-of-stake protocols [9, 15, 16]. However, all prior work considers strategic proposers who seek only to optimize their fraction of proposed slots, and do not consider variable execution rewards.

[5] [2] observed this for uniform rewards. Our results extend it to non-uniform reward structures, strengthening the case for longer epochs as a mitigation.

Selfish Mining. The analogous manipulation in Bitcoin is selfish mining, first analyzed by [14]. Subsequent work improves the analysis and considers optimal strategies [7, 23, 30] via an MDP methodology. Several works consider the effect of transaction fees on selfish mining. [10] shows that transaction fees radically change the analysis, and selfish mining variants can be profitable for arbitrarily small miners [7]. In a similar spirit to our work, [6] develops a generalized model of selfish mining to accommodate execution layer rewards. In comparison to these works, ours is the first to consider execution rewards such as MEV extracted through multiple consecutive blocks. However, our model does not subsume these – a strategic proposer in [6, 10] can adapt their strategy on the realized draw of transaction fees, whereas a proposer in our model does not. Moreover, our methodology finds the optimal strategy, rather than analyzing specific strategies that outperform honesty.

Computing Optimal Manipulations. [23] are the first to use MDP methodology to compute optimal manipulations in a blockchain protocol. MDPs are also used in [2, 7, 30], and similar convergence-based techniques are used in [16]. In comparison to these works, ours is the first to find optimal strategies for execution rewards beyond the uniform block rewards. In comparison to [2] specifically, our methodology can estimate ε-optimal strategies more efficiently and accommodate significantly more complex reward structures via sampling.

Maximal Extractable Value. Maximal extractable value (MEV) has been a large and growing part of reward modeling [4, 11]. Recently, there has been interest in understanding the game-theoretic implications of MEV on the incentive-compatibility of blockchain protocols [5, 17, 25]. MEV's interaction with various kinds of protocol deviations is an active area of research.

Validator Rewards and Slashing. Modifying validator rewards to disincentivize strategic deviations is a recurring theme in blockchain protocol design. Ethereum currently penalizes missed attestations uniformly via the inactivity leak [13], and there have been community discussions of introducing missed-slot penalties as a more targeted intervention [22]. A parallel line of work proposes time-sensitive or retroactive block rewards to disincentivize strategic delays in block proposal (timing games) [1, 24, 25]. We investigate a similar mitigation for RANDAO manipulation: we model tail slashing penalties as a reward function within our framework, and apply our methodology to determine slot miss penalties at the tail under which honest behavior is ε-optimal.

1.3 Main Contributions

- We analyze RANDAO manipulation under a wide array of stylized reward models (Section 4). This improves prior work by: (a) being the first to consider varying rewards such as execution layer rewards, and (b) allowing quick exploratory runs at coarse ε.
- We introduce a novel truncation of the RANDAO game that provides provable upper and lower bounds (Section 3), rendering many instances of the game with complicated reward functions computationally tractable.
- We prove robustness guarantees for our methodology, and provide a codebase[6] for easy replication of our results and running similar analyses for any reward function of interest.

[6] `https://github.com/kalpturer/randao-mev`.

- We apply our methodology to canonical settings of interest (Section 5, Figure 2), and conclude that the effect of RANDAO manipulation is highly dependent on the underlying reward model.
- We apply our framework to evaluate practical mitigations: epoch-length variation (Figure 4) and tail slashing penalties (Section 6). Modeling slashing as a reward function within our methodology, we compute, for each MEV reward model, minimal slashing curves that recover honest behavior at the equilibrium.

1.4 Roadmap

In Section 2, we model RANDAO manipulation with general rewards. In Section 3, we present our methodology of computing ε-optimal policies. We then proceed to analyze complex reward models in Section 4. In Section 5, we compare different reward models and show how much better a strategic proposer can do compared to honest proposers, including a sensitivity analysis on the epoch length. In Section 6, we analyze slashing-based mitigation approaches under which honest behavior is ε-optimal. Finally, Section 7 concludes with a summary and future work. We defer proofs to the appendix.

2 Modeling RANDAO

We now model RANDAO manipulation in its full generality, where the reward function is dynamic.

2.1 Background

Ethereum Consensus. Ethereum uses a Proof-of-Stake consensus mechanism which divides time into epochs with 32 slots, each of which lasts for 12 seconds. The participants in the protocol are called *validators* and stake ≥ 32 `ETH` each. For each epoch, 32 validators are sampled uniformly at random to propose the blocks in the epoch. The sampling procedure requires a reliable source of randomness, which is provided by the RANDAO mechanism [13].

RANDAO Mixing. Each block contains a 256-bit random seed called the RANDAO. The validator deterministically computes the block RANDAO by signing the current epoch number and then hashing the resulting signature. This hash is then mixed into the RANDAO value via an XOR operation. Note that this mixing is verifiable since the public key of the validator is known and the validator also includes the raw signature in the block. Thus, there is a unique value which a validator may use when updating the RANDAO. The future value of the RANDAO nonetheless cannot be computed by any third party ahead of time, since it relies upon both (a) the validator's private key, and (b) the RANDAO value as of the preceding block. The only exception to this rule occurs if and when a sequence of validators collude to reveal RANDAO values in advance. We model such a group of validators as the adversarial proposer.

Proposer Selection. The RANDAO value at the end of epoch E is used to determine the proposers for the next epoch by deterministically sampling 32 validators. The sampling protocol is complex, but can be accurately modeled by 32 independently and uniformly random selections from the pool of available validators.

Rewards. Ethereum employs a reward mechanism that incentivizes validators to propose, vote on blocks, and participate in the consensus protocol. Consensus-layer rewards are predictable and constitute a low-variance incentive for block proposal.[7] Execution-layer rewards, by contrast, are highly stochastic. The desire to maximize execution-layer rewards has driven the rise of a specialized and highly sophisticated ecosystem of Maximal Extractable Value[8] (MEV) searchers, block builders, and relays [21].

Adversary. We assume a single adversary that controls a fraction α of the total network stake. All other validators are assumed to be under the control of honest proposers and will therefore propose in each available slot. The adversary makes a strategic choice to propose or miss in each slot that they are assigned. We will hereafter refer to the consecutive proposal slots that the adversary is assigned at the end of an epoch as the *tail* of the epoch[9].

RANDAO Manipulation. Suppose the adversary controls a tail of length t in epoch E. Each subset of slots in which the adversary could propose will produce a unique RANDAO value at the end of the epoch.[10] Because the adversary controls the validators assigned to the tail, and hence knows their private keys, the RANDAO value associated with each slot can be deterministically precomputed. Thus the adversary may sample 2^t distinct future values of the RANDAO corresponding to each subset of tail slots in which they could propose.

Modeling Assumptions. We make some assumptions while modeling the RANDAO mechanism:

(a) Proposer sampling is uniformly random and the adversary gets selected to propose in a slot with α probability. Hence, each epoch assignment consists of 32 i.i.d. samples.

(b) We ignore interleaving of epochs and assume that the RANDAO value at the end of an epoch determines the proposers for the next epoch.[11]

(c) Honest proposers always propose and the adversary always chooses to propose when they are in non-tail slots as it is in their best interest to do so.

(d) We assume that the reward "resets" after an honest proposer, i.e., the reward distribution before and after an honest proposer is independent.

2.2 RANDAO Game

We now define the game induced by manipulating the RANDAO with an abstract reward model. Since general rewards may depend on the state in arbitrary ways, we begin by modeling the state space in its most general form as well. As an illustrative example, consider a scenario in which the adversary controls a tail of length 3. There are 2^3 different sequences of proposals that the adversary can make and one option is to do $\langle \mathsf{propose}, \mathsf{miss}, \mathsf{propose} \rangle$ which suppose results in a future epoch assignment with tail 30 of the form $\langle \mathsf{adversary}, \mathsf{honest}, \mathsf{adversary}, \dots \rangle$.

[7] The rewards are not fixed, but depend on the total number of validators in the network. However, this number is relatively stable.

[8] Maximal Extractable Value is defined as the reward available to a block proposer via their ability to selectively include, exclude, insert, and reorder transactions.

[9] For example, denoting adversarial slots by "A" and honest slots as "H", (...H) has tail 0, (...HA) has tail 1, (...HAA) has tail 2, and so on.

[10] Note that this is only the case because each slot in the tail is associated with a *unique* validator, each of which is under the control of the adversary. Thus the associated signatures are unique, meaning that the adversary has t distinct 256 bit strings which they may mix into the RANDAO.

[11] There is justification for this assumption in prior work [2], and it essentially does not affect the analysis for the adversarial stake range of interest ($\alpha < 1/2$) since each epoch contains at least one honest proposer with high probability.

In a reward model in which we only consider the number of blocks, this action will have a reward of 3 blocks plus getting to keep playing the game from a tail of 30. In the general case, however, there will be a known reward distribution that depends on the proposal sequence[12], and the adversary will first evaluate the expected reward of $\langle \mathsf{propose}, \mathsf{miss}, \mathsf{propose} \rangle$ followed by the expected reward of $\langle \mathsf{propose}, \mathsf{honest\ proposal} \rangle$ and the subsequent value of playing the game from a tail of length 30.

Proposal strings. At the core of our formulation is the definition of two types of proposal strings.

- *tail proposal string (TAIL):* $p_t \in \{0,1\}^k$ represents the event in which out of k consecutive slots all assigned to the adversary to propose, the adversary proposed in exactly the indices with 1.
- *non-tail proposal string (NEXT):* $p_n \in \{0,1\}^k$ represents the event in which out of k consecutive slots, the adversary proposed in exactly the indices with 1, while honest validators proposed in the remaining indices with 0 (recall that we assume all honest parties always propose).

Intuitively, when deciding what option to choose, the adversary will consider the expected reward of these two events and the resulting new tail of the action.

Epoch assignments. Let $\mathcal{E}$ be the distribution of epoch assignments of the form $\{0,1\}^{32}$ where each bit is identically and independently drawn with probability α. We will refer to the tail length of an epoch assignment $E \sim \mathcal{E}$ as $\mathsf{tail}(E)$ and the non-tail proposal string as $\mathsf{non\text{-}tail\text{-}proposals}(E)$.

Reward model. We define the rewards as an arbitrary function of the proposal strings representing the expected reward of the reward distribution. A reward model $\mathcal{R}$ is a pair $\langle \mathsf{propose}, \mathsf{next} \rangle$ where

- $\mathsf{propose} : \{0,1\}^* \to \mathbb{R}$ is a function that takes in a tail proposal string and outputs the expected reward of the adversary in taking this action.
- $\mathsf{next} : \{0,1\}^* \to \mathbb{R}$ is a function that takes in a non-tail proposal string and outputs the expected reward of the adversary in taking this action.

RANDAO game. Now, combining all the observations, we define the RANDAO manipulation game parameterized by adversarial stake α and reward model $\mathcal{R}$. The variables REWARD and ROUNDS count the accumulated reward by the adversary and the number of epochs that have passed. TAIL is used to keep track of the current tail controlled by the adversary.

(a) *Initialization.*
  1. Let REWARD := 0 and ROUNDS := 0.
  2. Let TAIL := 0.

(b) *Sampling.*
  3. Let $E_P \sim \mathcal{E}$ for all $P \subseteq \{0,1\}^{|\text{TAIL}|}$.
  4. Let N_P be $\mathsf{non\text{-}tail\text{-}proposals}(E_P)$ and similarly let T_P be $\mathsf{tail}(E_P)$.

[12] One way to think about this is to consider the distribution of rewards that a particular adversary is able to obtain. For example, if we consider an adversary that is really good at extracting multi-block MEV, the reward distribution will depend on whether proposals are consecutive. Alternatively if there is not much correlation between block rewards, we can use a per-slot independent model. We consider these in more detail in Section 4.

(c) *Strategic choice.*

5. The adversary chooses a subset P^* to propose in based on the reward model $\mathcal{R}$ and the set of sampled epochs.

(d) *Updates.*

6. Let $\textsc{Reward} := \textsc{Reward} + \mathsf{propose}_{\mathcal{R}}(P^*) + \mathsf{next}_{\mathcal{R}}(N_{P^*})$.
7. Let $\textsc{Tail} = T_{P^*}$.
8. Let $\textsc{Rounds} := \textsc{Rounds} + 1$.
9. Repeat from step 3.

Objective. The optimization objective is $\liminf\{\textsc{Reward}/\textsc{Rounds}\}$, the average reward in the infinite game.

3 Solving the RANDAO Game

In this section, we define the MDP formulation of the RANDAO game, using the state space reduction observation from [2]. The MDP formulation reduces the state space to just the tail lengths $\{0, 1, \ldots, 32\}$, pushing the complexity of the game into the action space. The MDP solver in [2], however, does not immediately generalize to our case.

Diverging from [2], to give probable bounds on our sampling-based approach, we observe that the rare states can be truncated without significantly affecting the optimal reward. Concretely, we will define two truncations of the MDP where we ignore states above a certain threshold: (a) an *upper truncation* that rewards large tails state transitions maximally, and (b) a *lower truncation* that rewards large-tail state transitions 0. This will allow us to bound the optimal reward of the full MDP between the optimal rewards of the two truncated MDPs.

Comparison with [2]. Our MDP uses the same state space as [2], the tail length $t \in \{0, \ldots, 32\}$. The main difference is in computing the transition probabilities. Computing each $P(t, t')$ requires summing, over the reward realizations the adversary could face from t, the probability that the adversary's optimal response leads to t'. Under uniform per-block rewards the sum has manageably few terms because the optimal response depends only on the *number of proposals*. Once the reward depends on the *pattern*, as in our examples, the sum has many more terms and quickly becomes intractable to evaluate. For example, the smooth multi-block model (Example 2) pays some function of k for each block of k consecutive proposals, so the sum has to enumerate over integer partitions of every proposal count $k \leq t$. Our methodology sidesteps this combinatorial issue by estimating the transitions via sampling, with an ε-approximation guarantee from a truncation argument and a concentration bound. As a practical benefit, new reward models can be plugged into the solver without re-deriving any closed forms.

3.1 Methodology

Observe that we can immediately write down the MDP for the RANDAO game with reward model $\mathcal{R}$. Let $\mathcal{S} = \{0, 1, \ldots, 32\}$ be the state space, $\mathcal{A}$ be the set of valuation functions $v : S \to \mathbb{R}$. Denote by $\pi : \mathcal{S} \to \mathcal{A}$ a policy. Let $P_\pi : \mathcal{S} \times \mathcal{S} \to [0, 1]$ be the transition probability such that $P_\pi(t, t')$ is the probability of transitioning from tail t to tail t' when the adversary picks the option that maximizes $\mathsf{propose}(P) + \mathsf{next}(N_P) + v(\mathsf{tail}(E_P))$. Let $R_\pi : \mathcal{S} \to \mathbb{R}$ also be the expected reward of taking the action given by π in the game. Lastly,

let $\mathsf{OPT}(\cdot)$ be the optimal reward of either the RANDAO MDP/game. Using a similar argument to [2], we know that we can reduce the state space to just the tail length by working with the RANDAO MDP.

▶ **Lemma 4.** $\mathsf{OPT}(G) = \mathsf{OPT}(M)$ *where G is the RANDAO manipulation game and M is the RANDAO MDP.*

As discussed at the start of this section, computing the transition probabilities for non-uniform rewards in closed form is not always tractable. We therefore propose sampling the transition probabilities to estimate them. While sampling transitions from large tails is expensive, we observe that we can truncate the MDP to small tails. Intuitively, as large tails are rare, ignoring them will not change the optimal reward by much.

Approximate RANDAO Solver.

1. Truncate the MDP to a state space $S^T = \{0, 1, \ldots, T\}$ such that all transitions that go to a tail $t > T$ now transition to T. The reward of the transition is either X for upper truncation or 0 for lower truncation. We call this the T-truncation of the model[a].
2. Initialize values $v(\cdot) = 0$ for each state.
3. Empirically estimate the truncated transition probabilities and reward vector with N samples for each $P(t, \cdot)$ and $R(\cdot)$ to get matrix $\hat{P}$ and vector $\hat{R}$.
4. Compute average reward γ for $\langle \hat{P}, \hat{R} \rangle$ using the stationary distribution of $\hat{P}$ and taking the product with $\hat{R}$.
5. Perform a policy improvement step by solving for values using $V + \gamma = \hat{R} + \hat{P}V$ where V is the value vector, and set $v := V$. If values did not change more than the tolerance, output the current valuation. Otherwise continue from step 3.

[a] We take $X = R_{\max} \cdot \mathbb{E}[L(T, \alpha)]$, an upper bound on the expected reward accrued during a large-tail sequence (see Lemma 5). As we show later, the upper and lower truncations give upper and lower bounds.

3.2 Reward Guarantees

We will now analyze the optimality loss of *truncation* and using the *approximate* model generated by samples. These bounds will ensure that we truncate conservatively and sample enough times to accurately approximate the optimal reward. Our analysis will depend on an upper bound on the maximum reward of a single transition $R_{\max}$. For example, if the reward is the number of blocks proposed, then $R_{\max} = 32$.

Truncation bound. Intuitively, we observe that any transition to a large tail $t > T$ must at some point transition back to a small tail $t \leq T$ for bounded α. Hence, we can bound the expected number of large tail transitions and assign the maximum reward $R_{\max}$ to each of these transitions to get an upper bound. To do this, we first need to bound the expected length of consecutive large tail transitions.

▶ **Lemma 5.** *For any policy π, let $L^\pi(T, \alpha)$ denote the number of consecutive rounds the adversary spends in tails greater than T, starting from $t = 32$. Let $L(T, \alpha)$ be the corresponding quantity under the policy that always picks the longest tail. Then $L(T, \alpha)$ stochastically dominates $L^\pi(T, \alpha)$ for every π, and*

$$\mathbb{E}[L(T, \alpha)] \leq X_{T,\alpha}$$

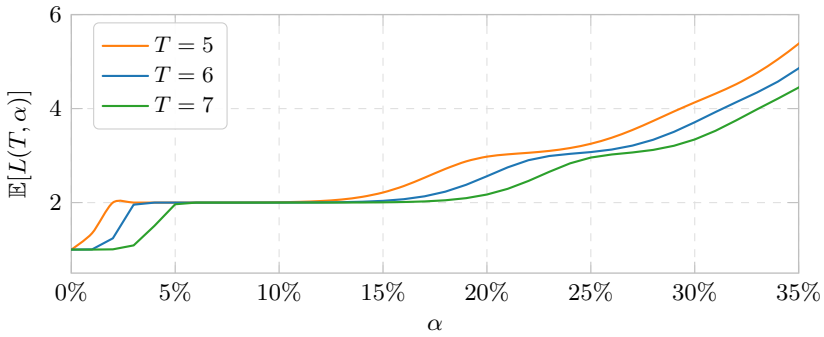


Figure 1 Bound on $\mathbb{E}[L(T,\alpha)]$ from Lemma 5, plotted versus stake fraction α for truncation thresholds $T \in \{5,6,7\}$.

where $X_{T,\alpha}$ is the last element of the vector $(I - P_T)^{-1}\mathbf{1}$ where P_T is the transition matrix of the Markov chain with state space $\{T+1,\ldots,32\}$ and transition probabilities of the strategy that always picks the longest tail.

▶ **Theorem 6.** *Let M^+ and M^- be the upper and lower T-truncations of the RANDAO MDP M. Then,* $\mathsf{OPT}(M^-) \le \mathsf{OPT}(M) \le \mathsf{OPT}(M^+)$

Sampling complexity. We now observe that by an application of the Chernoff-Hoeffding bound and a union bound over the $(T+1)^2$ entries of the transition matrix and the $(T+1)$ entries of the reward vector, we can derive the sufficient number of samples to estimate the probabilities and rewards up to an error of ε with probability $1-\delta$.

▶ **Lemma 7.** *Given $\varepsilon, \delta > 0$, let $\hat{P}$ and $\hat{R}$ be the empirical estimates of the transition probabilities and rewards using N independent samples per state from the truncated MDP M^T with a fixed policy where*

$$N \ge \max\left(\frac{(T+1)^2}{2\varepsilon^2}\ln\left(\frac{4(T+1)^2}{\delta}\right), \frac{R_{\max}^2}{2\varepsilon^2}\ln\left(\frac{4(T+1)}{\delta}\right)\right)$$

Then, with probability at least $1-\delta$, both of the following hold:

$$\sum_{t'} |P(t,t') - \hat{P}(t,t')| \le \varepsilon, \quad \forall t \in \{0,\ldots,T\}$$

$$|R(t) - \hat{R}(t)| \le \varepsilon, \quad \forall t \in \{0,\ldots,T\}$$

For $\varepsilon = 0.0085$ and $\delta = 0.01$, $N = 10^8$ samples per state is sufficient for $T = 7$ and the $R_{\max}$ of the reward models we consider. We will be using this setting in our experiments.

4 MEV Models and Results

In practice, MEV is a complex phenomenon that is affected by not just the number of blocks, but their successive combinations or the result of a stochastic process. We will therefore consider a few different MEV models that capture some of these phenomena. For all reported utilities, we plot the reward per slot (i.e. the average reward for a single slot) against α. We focus on stake fractions $\alpha \le 0.35$ throughout: at larger α the expected time spent in large tails $\mathbb{E}[L(T,\alpha)]$ (Lemma 5, Figure 1) grows rapidly, and the truncation bound $R_{\max} \cdot \mathbb{E}[L(T,\alpha)]$ separating our upper and lower estimates degrades.

4.1 Warm-up: Reproducing Uniform Reward Results

To validate our methodology, we first reproduce the results of [2] in the case of uniform rewards. In this case, the reward model $\mathcal{R}$ is defined as the number of 1's in the proposal string for both propose and next. The results are shown in Table 1.

Table 1 Comparison of the optimal rewards reported by [2] with our method's U (upper truncation) and L (lower truncation) bounds for uniform rewards (a single unit per proposed block). Each entry has an error of at most 0.0265% ($\varepsilon/32$).

α	5%	10%	15%	20%	25%	30%	35%
[2]	5.0483%	10.1881%	15.3996%	20.6777%	26.0247%	31.4516%	36.9734%
U	5.0483%	10.1879%	15.3998%	20.6783%	26.0323%	31.4916%	37.2290%
L	5.0483%	10.1878%	15.3997%	20.6776%	26.0250%	31.4487%	36.9581%

4.2 Per-slot MEV

We define the per-slot MEV reward model as follows (Example 1). In every slot, there is a base reward of 1 in two components:

- *Ephemeral reward.* This is the reward for proposing a block in that slot. If the block is not proposed, this reward is lost.
- *Rollover reward.* This is the reward that rolls over from the previous slot if it was not proposed. If the block is proposed, this reward is claimed. If the block is not proposed, this reward rolls over to the next slot.

Intuitively, there are several reasons why this model is reasonable. Certain consensus-layer block rewards are ephemeral and are lost if the current block is not proposed, while many execution-layer rewards (such as priority fees and arbitrage or liquidation opportunities) roll over to the next block if not claimed. Even some of the consensus rewards on Ethereum roll over: the block immediately following a missed slot contains approximately twice the proposer reward (see Figure 6) due to the additional attestations that can be included. We will use 1 reward per slot with 60% ephemeral and 40% rollover reward. Let $\mathsf{next}(n_t)$ be defined simply as the number of 1s in p_{t+1}. Let $\mathsf{propose}(p_t)$ be defined such that each proposed block gives a reward of $1.0 + r$, where r is the amount of rollover reward from previous slots. Each missed slot accumulates 0.4 rollover, and a proposal claims and resets it to 0. Using $R_{\max} = 32$ for this model, we run our approximate RANDAO solver and the results are shown in Table 2.

Table 2 Optimal rewards for the per-slot MEV reward model. U (upper truncation) and L (lower truncation) are compared with the honest strategy H. Each entry has an error of at most 0.0265% ($\varepsilon/32$).

α	5%	10%	15%	20%	25%	30%	35%
U	5.0493%	10.1922%	15.4107%	20.7007%	26.0734%	31.5628%	37.3514%
L	5.0493%	10.1922%	15.4106%	20.6999%	26.0660%	31.5194%	37.0754%
H	4.9999%	10.0009%	15.0015%	20.0017%	25.0016%	30.0018%	35.0016%

4.3 Multi-block MEV

We now define the multi-block MEV reward models as follows. In every epoch, the rewards scale with the number of consecutive blocks proposed. We will consider two variants of this model:

- **Smooth Model.** The reward for proposing k consecutive blocks is $k\sqrt{k}$, a smooth superlinear shape that captures phenomena such as arbitrage across multiple blocks (Example 2).
- **Threshold Model.** If k consecutive blocks are proposed, the reward is k for $k < 5$ and $k + 10$ for $k \geq 5$. This captures phenomena such as oracle manipulation, where a proposer with a long enough run can manipulate an oracle to their advantage and gain a large reward (Example 3).

Both multi-block reward models above are stylized illustrations rather than calibrated estimates. The $k^{3/2}$ shape is loosely inspired by CFAMM arbitrage with fees [19]: fees create a no-trade band that the cross-venue price gap must cross before arbitrage becomes profitable, and in the regime relevant to short multi-block sequences the resulting option-like payoff can grow superlinearly in k. The threshold choice ($k = 5$, bonus 10) is similarly illustrative. Documented TWAP attacks have their sharp discontinuity at $k = 2$ [18]. Our purpose is to showcase a methodology that can quantify the impact of any such reward structure, should one emerge in practice for multi-block strategies that are not yet exploited at scale. The results for both models are shown below in Table 3.

Table 3 Optimal rewards for multi-block MEV models. Results for the smooth reward model (top) and threshold reward model (bottom) are shown. U (upper truncation) and L (lower truncation) are compared with the honest strategy H.

Smooth Rewards							
α	5%	10%	15%	20%	25%	30%	35%
U	5.2616%	11.0913%	17.5533%	24.7437%	32.8185%	42.0835%	53.7479%
L	5.2616%	11.0913%	17.5529%	24.7392%	32.7769%	41.8354%	52.1490%
H	5.2072%	10.8585%	17.0013%	23.6931%	31.0040%	39.0198%	47.8438%

Threshold Rewards							
α	5%	10%	15%	20%	25%	30%	35%
U	5.0539%	10.2793%	15.8691%	22.1828%	29.7048%	38.9595%	50.7152%
L	5.0539%	10.2793%	15.8689%	22.1806%	29.6848%	38.8404%	49.9715%
H	5.0052%	10.0829%	15.3942%	21.1721%	27.6875%	35.2180%	44.0074%

5 Improvement over Honest Strategy

The severity of the incentive to manipulate RANDAO can be measured by the percentage increase in expected rewards per time over the honest strategy of proposing in every slot. We use this metric to compare the effects of different reward models on RANDAO manipulation incentives.

▶ **Definition 8** (Improvement). *Let $M^{\mathcal{R}}$ be a RANDAO MDP with reward model $\mathcal{R}$ and $h^{\mathcal{R}}$ be the average reward of the honest strategy in $M^{\mathcal{R}}$. Then, the improvement over the honest strategy is defined as:* $\frac{\mathsf{OPT}(M^{\mathcal{R}}) - h^{\mathcal{R}}}{h^{\mathcal{R}}}$

We find that this incentive is highly dependent on the reward model used as shown in Figure 2. This suggests that as the MEV extraction landscape evolves, the incentive to manipulate RANDAO may also change.

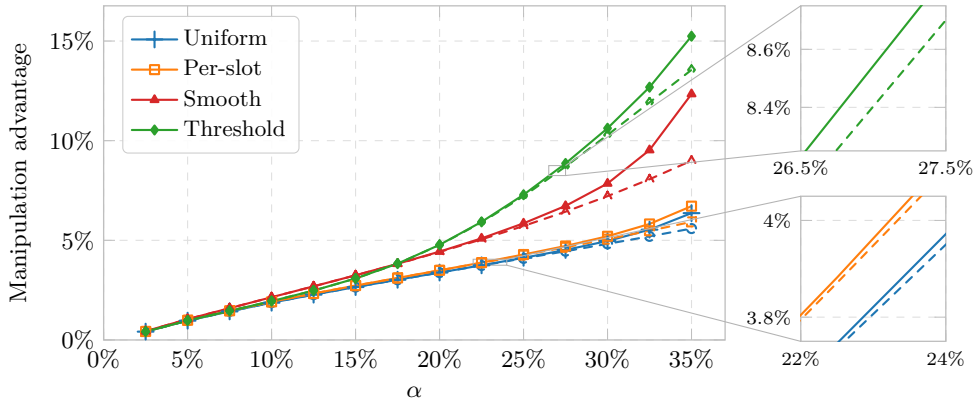


Figure 2 Manipulation advantage over the honest strategy across reward models, measured as $(\mathsf{OPT}(M^{\mathcal{R}}) - h^{\mathcal{R}})/h^{\mathcal{R}}$. Solid lines with filled markers show the upper-truncation bound. Dashed lines show the lower-truncation bound.

Error Analysis. We plot the absolute error between the upper and lower truncation in Figure 3. The error is small for small values of α, as expected, and the primary source of error is sampling. For larger values of α, the error increases as the optimal policy starts to depend more on the large tail states. However, even for $\alpha = 0.35$, the difference remains small. If higher precision is required, more samples and higher truncation T can be used.

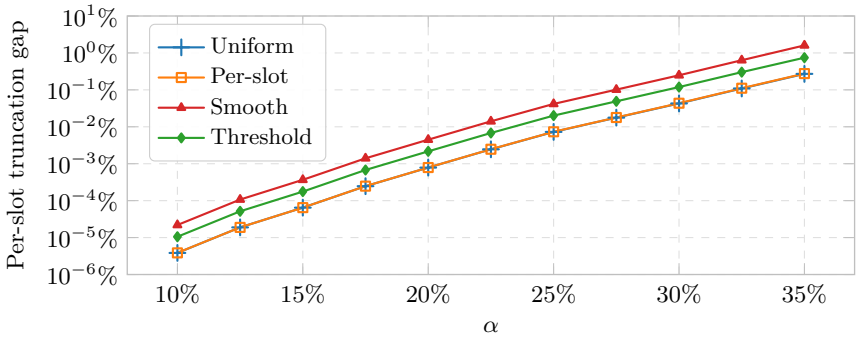


Figure 3 Per-slot gap between upper- and lower-truncation estimates of the optimal average reward at truncation tail length $T = 7$, across reward models and stake fractions α.

Epoch-Length Sensitivity. A natural protocol parameter to revisit is the epoch length ℓ itself, currently fixed at $\ell = 32$ slots on Ethereum mainnet. Let $M_\ell^{\mathcal{R}}$ denote the RANDAO MDP for reward model $\mathcal{R}$ under epoch length ℓ, and $h_\ell^{\mathcal{R}}$ the corresponding honest-strategy average reward. Figure 4 reports the relative manipulation gain $(\mathsf{OPT}(M_\ell^{\mathcal{R}}) - h_\ell^{\mathcal{R}})/h_\ell^{\mathcal{R}}$ as we sweep ℓ, holding α fixed and computed separately for each of the four MEV reward models of Section 4. Across all four models, we observe that the manipulation incentive decays with increased epoch length. Lengthening the epoch is therefore a partial mitigation, but for any concrete ℓ the manipulation incentive depends on the reward model.

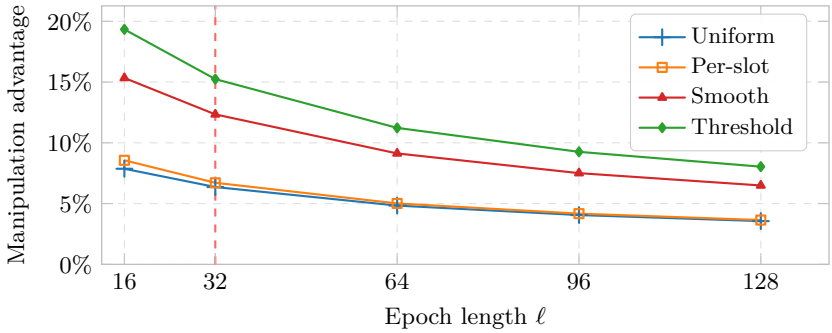


Figure 4 Relative manipulation gain $(\mathsf{OPT}(M_\ell^{\mathcal{R}}) - h_\ell^{\mathcal{R}})/h_\ell^{\mathcal{R}}$ versus epoch length ℓ at $\alpha = 0.35$, one curve per MEV reward model. The vertical red dashed line at $\ell = 32$ marks the current Ethereum value.

6 Tail Slashing

The preceding sections characterize how the incentive to manipulate RANDAO depends on the reward model. A natural next question is whether the reward structure itself can be modified to disincentivize manipulation. Introducing a missed-slot penalty in normal operation has been discussed within the Ethereum community [22]. In this section we apply our methodology in reverse: rather than computing the optimal adversarial strategy for a fixed reward model, we ask, for each MEV reward model, what is a minimal slashing curve that we can apply to the tail slots under which honest behavior becomes ε-optimal? We model tail slashing as a negative component of the reward function and reuse the solver.

6.1 Slashing as a Reward Model

We now pick an illustrative slashing scheme that penalizes tail misses. Let the epoch have length $\ell = 32$ and let $q \in \{0,1\}^t$ denote the adversary's proposal pattern over a tail of length t, with $q_i = 1$ if the slot at position i from the end of the epoch was proposed ($i = 0$ is the last slot). Manipulation is driven by control of the final slots of an epoch, so we concentrate the penalty there. The penalty is driven by two factors: how close the missed slot is to the end of the epoch, and how many other misses occur in the same tail.

Slashing scheme. Consider the following slashing scheme parameterized by a scaling constant κ. For a fixed window $W = 8$, in every epoch, consider the last W slots. For each missed slot i in the tail (numbered such that $i = 0$ is the last slot), we will assign a slashing penalty of $\kappa \cdot \frac{W-i}{W} \cdot D$ where D is an additional multiplier equal to $1 + (\#\text{ of missed slots in the window})/W$. We run our experiments and discuss manipulability for this particular formula. For the game where every honest player proposes against a single α-adversary, we can write this as the following.

▶ **Definition 9** (Tail slashing). *Fix a window size $W \in \mathbb{N}$ and a slashing magnitude $\kappa \geq 0$. Index tail slots so that $i = 0$ is the last slot of the epoch. For a tail-proposal string q, define*

$$S(q) = \sum_{i=0}^{\min(t,W)-1} \mathbf{1}[q_i = 0] \cdot \tfrac{W-i}{W}, \qquad C(q) = \big|\{i < \min(t,W) : q_i = 0\}\big|.$$

The slashing penalty for q is

$$s_\kappa(q) = -\kappa \cdot \left(1 + \tfrac{C(q)}{W}\right) \cdot S(q)$$

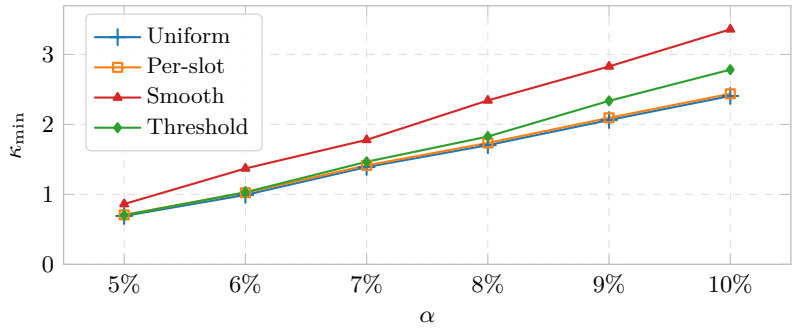


Figure 5 Slashing threshold $\kappa_{\min}$ versus stake fraction α, for each of the four MEV reward models of Section 4.

▶ **Example 10** (slashing with $W = 8$)**.** We list the penalty for several tail-proposal patterns. Each binary string shows the proposal pattern over the window, read so the rightmost bit is the last slot of the epoch ($i = 0$).

`11111110`	$: s_\kappa = -\kappa \cdot \left(1 + \frac{1}{W}\right) \cdot 1$	$\approx -1.125\,\kappa$	(miss at the last slot)
`01111111`	$: s_\kappa = -\kappa \cdot \left(1 + \frac{1}{W}\right) \cdot \frac{1}{W}$	$\approx -0.141\,\kappa$	(miss at $i = W-1$)
`11111100`	$: s_\kappa = -\kappa \cdot \left(1 + \frac{2}{W}\right) \cdot \frac{2W-1}{W}$	$\approx -2.344\,\kappa$	(two clustered misses at end)
`00000000`	$: s_\kappa = -\kappa \cdot \left(1 + \frac{W}{W}\right) \cdot \frac{W+1}{2}$	$= -9\,\kappa$	(all window slots missed)

Slashed reward model. Given a reward model $\mathcal{R} = \langle \mathsf{propose}, \mathsf{next} \rangle$, the slashed model is

$$\mathcal{R}^s_\kappa = \langle \mathsf{propose} + s_\kappa, \mathsf{next} \rangle.$$

Slashing only modifies the propose component not next. We write $M^{\mathcal{R}^s_\kappa}$ for the RANDAO MDP (Section 3) instantiated with $\mathcal{R}^s_\kappa$.

▶ **Definition 11** (Slashing threshold)**.** *For reward model $\mathcal{R}$, stake fraction α, and tolerance $\varepsilon > 0$, the slashing threshold $\kappa_{\min}$ is the minimum slashing magnitude $\kappa \geq 0$ for which honest behavior is ε-optimal in the slashed MDP $M^{\mathcal{R}^s_\kappa}$. That is, the optimal reward $\mathsf{OPT}(M^{\mathcal{R}^s_\kappa})$ is at most ε above the honest average reward $h^{\mathcal{R}}$.*

The following monotonicity ensures that $\kappa_{\min}$ is well-defined.

▶ **Lemma 12** (Monotonicity in κ)**.** $\mathsf{OPT}(M^{\mathcal{R}^s_\kappa})$ *is non-increasing in κ.*

To certify that honest behavior is ε-optimal under a given κ, we need an upper bound on $\mathsf{OPT}(M^{\mathcal{R}^s_\kappa})$. We therefore use only the upper truncation M^+ when computing $\kappa_{\min}$, and run at $N = 10^7$ samples per state.

6.2 Slashing Magnitudes

Figure 5 reports $\kappa_{\min}$ for the four MEV reward models of Section 4 across stake fractions α from 0.05 to 0.10. We observe that the ordering of $\kappa_{\min}$ at fixed α across MEV reward models matches the ordering of the manipulation gain $\mathsf{OPT}(M^{\mathcal{R}}) - h^{\mathcal{R}}$ from Section 5: reward models that incentivize manipulation more strongly require larger penalties to neutralize as expected. Similarly, $\kappa_{\min}$ is monotonically increasing in α, since larger adversaries gain more from manipulation and require correspondingly larger slashing magnitudes.

Solo-staker friendliness. One concern for introducing a missed slot penalty is the effect on solo stakers. Solo stakers in practice miss only a small fraction of their assigned slots (roughly 1–2%), as i.i.d. accidents rather than coordinated drops at the tail. Under such a miss pattern, our slashing curve is far weaker on such validators than a fixed per-slot penalty that is independent of tail position and clustering. Misses early in the epoch incur little or no penalty under the decaying weight S, and the multi-miss multiplier stays close to 1 for isolated misses. A uniform per-slot penalty, by contrast, would charge the same κ for every accidental miss regardless of where in the epoch it falls or how it clusters with other misses.

7 Conclusion

We have shown that the incentive to manipulate RANDAO is highly dependent on the MEV reward model. In particular, partial reward rollover and multi-block MEV both amplify the manipulation gain, while lengthening the epoch is a partial mitigation across all reward models we considered. We also showed that tail slashing, modeled as a reward function within our methodology, can make honest behavior ε-optimal. Our methodology can be used to analyze the incentives around RANDAO in a variety of settings. For instance, recent work [8] incorporates a RANDAO bias analysis in the design of a protocol that uses randomness. Our codebase and general approach can streamline such analyses for other applications with complex reward models.

Future work. Three directions remain open. First, Section 6 analyzes a single tail-slashing curve built from a decaying penalty and a multi-miss multiplier. A natural extension is to explore alternative slashing curves, which may achieve smaller costs for honest validators under the same deterrence guarantee. Second, our framework works at the epoch level and we assume that the adversary commits to an action for the entire epoch at once. However, in practice, the adversary may do even better by adapting their strategy based on the blocks they see in the current epoch online. Analyzing this adaptive strategy is an interesting direction for future work. Third, the empirical gap between our upper and lower bounds is much smaller than our sample-complexity bound predicts, suggesting room for a tighter analysis.

References

1 Kaya Alpturer, Kushal Babel, and Aditya Saraf. Timing games in responsive consensus protocols, 2025. doi:10.48550/arXiv.2510.25144.

2 Kaya Alpturer and S. Matthew Weinberg. Optimal RANDAO Manipulation in Ethereum. In *AFT 2024*, 2024.

3 Musab A Alturki and Grigore Roşu. Statistical model checking of randao's resilience to pre-computed reveal strategies. In *Formal Methods – FM 2019 International Workshops*, pages 337–349. Springer, 2019.

4 Kushal Babel, Philip Daian, Mahimna Kelkar, and Ari Juels. Clockwork finance: Automated analysis of economic security in smart contracts. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 2499–2516. IEEE, 2023. doi:10.1109/SP46215.2023.10179346.

5 Maryam Bahrani, Pranav Garimidi, and Tim Roughgarden. Transaction fee mechanism design in a post-mev world. In *6th Conference on Advances in Financial Technologies (AFT 2024)*, volume 316 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 29:1–29:24, 2024. doi:10.4230/LIPIcs.AFT.2024.29.

6 Maryam Bahrani, Michael Neuder, and S. Matthew Weinberg. Selfish mining under general stochastic rewards, 2025. doi:10.48550/arXiv.2502.20360.

7 Roi Bar-Zur, Ameer Abu-Hanna, Ittay Eyal, and Aviv Tamar. Werlman: To tackle whale (transactions), go deep (rl). In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 93–110. IEEE, 2023. doi:10.1109/SP46215.2023.10179444.

8 Bhargav Nagaraja Bhatt, Fatemeh Shirazi, and Alistair Stewart. Trustless bridges via random sampling light clients. AFT 2025, 2025. URL: https://eprint.iacr.org/2025/057.

9 Linda Cai, Jingyi Liu, S. Matthew Weinberg, and Chenghan Zhou. Profitable Manipulations of Cryptographic Self-Selection Are Statistically Detectable. In Rainer Böhme and Lucianna Kiffer, editors, *6th Conference on Advances in Financial Technologies (AFT 2024)*, volume 316 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:23, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.AFT.2024.30.

10 Miles Carlsten, Harry Kalodner, S. Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 154–167, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2976749.2978408.

11 Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE symposium on security and privacy (SP)*, pages 910–927. IEEE, 2020. doi:10.1109/SP40000.2020.00040.

12 Data Always. mevboost-data: MEV-Boost relay bid archive. https://github.com/dataalways/mevboost-data.

13 Ben Edgington. *Upgrading Ethereum.* eth2book.info, Capella edition, 2023. URL: https://eth2book.info/.

14 Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, pages 436–454. Springer, 2014. doi:10.1007/978-3-662-45472-5_28.

15 Matheus V. X. Ferreira, Ye Lin Sally Hahn, S. Matthew Weinberg, and Catherine Yu. Optimal strategic mining against cryptographic self-selection in proof-of-stake. In David M. Pennock, Ilya Segal, and Sven Seuken, editors, *EC '22: The 23rd ACM Conference on Economics and Computation, Boulder, CO, USA, July 11 - 15, 2022*, pages 89–114. ACM, 2022. doi:10.1145/3490486.3538337.

16 Matheus V.X. Ferreira, Aadityan Ganesh, Jack Hourigan, Hannah Huh, S. Matthew Weinberg, and Catherine Yu. Computing optimal manipulations in cryptographic self-selection proof-of-stake protocols. In *EC '24: The 25th ACM Conference on Economics and Computation*. ACM, 2024. doi:10.1145/3670865.3673602.

17 Flashbots and Robert Miller. Mev and the limits of scaling. https://writings.flashbots.net/mev-and-the-limits-of-scaling, 2025.

18 Torgin Mackinga, Tejaswi Nadahalli, and Roger Wattenhofer. Twap oracle attacks: Easier done than said? In *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–8. IEEE, 2022. doi:10.1109/ICBC54727.2022.9805499.

19 Jason Milionis, Ciamac C Moallemi, and Tim Roughgarden. Automated market making and arbitrage profits in the presence of fees. In *International Conference on Financial Cryptography and Data Security*, pages 159–171. Springer, 2024. doi:10.1007/978-3-031-78676-1_9.

20 Ábel Nagy, János Tapolcai, István András Seres, and Bence Ladóczki. Forking the RANDAO: Manipulating ethereum's distributed randomness beacon. Cryptology ePrint Archive, Paper 2025/037, 2025. URL: https://eprint.iacr.org/2025/037.

21 Burak Öz, Danning Sui, Thomas Thiery, and Florian Matthes. Who Wins Ethereum Block Building Auctions and Why? In Rainer Böhme and Lucianna Kiffer, editors, *6th Conference on Advances in Financial Technologies (AFT 2024)*, volume 316 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:25, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.AFT.2024.22.

22 Max Resnick. Missed slot penalties: temperature check, 2024. ethresear.ch forum post. URL: https://ethresear.ch/t/missed-slot-penalties-temperature-check/18713.

23 Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *Financial Cryptography and Data Security - 20th International Conference, FC 2016, Christ Church, Barbados, February 22-26, 2016, Revised Selected Papers*, pages 515–532, 2016. doi:10.1007/978-3-662-54970-4_30.

24 Caspar Schwarz-Schilling. Retroactive proposer rewards. Ethereum notes. URL: https://notes.ethereum.org/@casparschwa/S1vcyXZL9.

25 Caspar Schwarz-Schilling, Fahad Saleh, Thomas Thiery, Jennifer Pan, Nihar Shah, and Barnabé Monnot. Time is money: Strategic timing games in proof-of-stake protocols. In *5th Conference on Advances in Financial Technologies (AFT 2023)*, volume 282 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:17, 2023. doi:10.4230/LIPIcs.AFT.2023.30.

26 Pascal Stichler. Does multi-block MEV exist? Analysis of 2 years of MEV data, August 2024. URL: https://ethresear.ch/t/does-multi-block-mev-exist-analysis-of-2-years-of-mev-data/20345.

27 János Tapolcai, Bence Ladóczki, and Ábel Nagy. Slot a la carte: Centralization issues in ethereum's proof-of-stake protocol. Cryptology ePrint Archive, Paper 2025/219, 2025. URL: https://eprint.iacr.org/2025/219.

28 Toni Wahrstätter. Selfish mixing and randao manipulation, July 2023. URL: https://ethresear.ch/t/selfish-mixing-and-randao-manipulation/16081.

29 Tao Yan, Shengnan Li, Benjamin Kraner, Luyao Zhang, and Claudio J. Tessone. Replication Data for: "Analyzing Reward Dynamics and Decentralization in Ethereum 2.0: A Data Engineering Workflow and Datasets". Harvard Dataverse, V1, 2025. doi:10.7910/DVN/HG36LO.

30 Roi Bar Zur, Ittay Eyal, and Aviv Tamar. Efficient mdp analysis for selfish-mining in blockchains. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, AFT '20, pages 113–131, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3419614.3423264.

A Proofs for Section 3 (Solving the RANDAO Game)

▶ **Lemma 4.** $\mathsf{OPT}(G) = \mathsf{OPT}(M)$ *where* G *is the RANDAO manipulation game and* M *is the RANDAO MDP.*

Proof. We first observe that the game is ergodic, since each epoch has a non-zero probability of having samples with the same fixed epoch (or tail length). Hence, the optimal average reward is independent of the starting state.

Now consider the RANDAO game G. Note that every epoch has a unique tail length. For any strategy playing the game, there is a corresponding strategy for the MDP that induces the same sequence of tails. As the game is ergodic, each state pair will be visited infinitely often, and in expectation these strategies will have the same average reward. Similarly, any strategy playing the MDP has a corresponding strategy in the game as the actions are the same. Hence, the optimal average reward must be the same. ◀

▶ **Lemma 5.** *For any policy* π*, let* $L^\pi(T, \alpha)$ *denote the number of consecutive rounds the adversary spends in tails greater than* T*, starting from* $t = 32$*. Let* $L(T, \alpha)$ *be the corresponding quantity under the policy that always picks the longest tail. Then* $L(T, \alpha)$ *stochastically dominates* $L^\pi(T, \alpha)$ *for every* π*, and*

$$\mathbb{E}[L(T, \alpha)] \leq X_{T,\alpha}$$

where $X_{T,\alpha}$ is the last element of the vector $(I - P_T)^{-1}\mathbf{1}$ where P_T is the transition matrix of the Markov chain with state space $\{T+1, \ldots, 32\}$ and transition probabilities of the strategy that always picks the longest tail.

Proof. We start by observing that the strategy that always picks the longest tail stochastically dominates any other strategy in terms of the sequence of rounds spent in tails greater than T. Consider the next epoch samples across strategies: the tail-maximizing choice realizes a next-state tail length that is at least as large as any other strategy's. Moreover, choosing the max tail dominates returning to smaller states. Hence, the expected length of consecutive rounds in large tails of the tail-maximizing strategy is an upper bound on the expected length of consecutive rounds in large tails of any strategy. Lastly, we observe that the length of this path is maximized by starting from the largest tail 32. Hence, as the expected length of rounds in large tails starting from each large state is given by the vector $(I - P_T)^{-1}\mathbf{1}$, the expected length starting from 32 is the last element of this vector. ◀

▶ **Theorem 6.** *Let M^+ and M^- be the upper and lower T-truncations of the RANDAO MDP M. Then,* $\mathsf{OPT}(M^-) \leq \mathsf{OPT}(M) \leq \mathsf{OPT}(M^+)$

Proof. M^- differs from M only by replacing the reward on each transition to a large tail with 0, so any policy's average reward in M^- is at most its average reward in M. Hence, $\mathsf{OPT}(M^-) \leq \mathsf{OPT}(M)$. Now consider the optimal policy π^* for M. Observe that whenever π^* transitions to a large tail $t > T$, it must eventually transition back to a small tail $t \leq T$. As this will last less than $L(T, \alpha)$ rounds in expectation, we can upper bound the expected reward of this path by $R_{\max} \cdot \mathbb{E}[L(T, \alpha)]$. Therefore, the corresponding policy in M^+ that takes the same actions as π^* in small tails and transitions to T with reward $R_{\max} \cdot \mathbb{E}[L(T, \alpha)]$ in large tails will have an expected reward at least as much as π^*. Moreover, since the M^+ policy collapses each large-tail sequence into a single transition while π^* takes $\mathbb{E}[L(T, \alpha)]$ steps in expectation, the M^+ policy accumulates at least as much reward over strictly fewer transitions, so its long-run average reward is at least that of π^*. Hence, $\mathsf{OPT}(M) \leq \mathsf{OPT}(M^+)$ as desired. ◀

▶ **Lemma 7.** *Given $\varepsilon, \delta > 0$, let $\hat{P}$ and $\hat{R}$ be the empirical estimates of the transition probabilities and rewards using N independent samples per state from the truncated MDP M^T with a fixed policy where*

$$N \geq \max\left(\frac{(T+1)^2}{2\varepsilon^2} \ln\left(\frac{4(T+1)^2}{\delta}\right), \frac{R_{\max}^2}{2\varepsilon^2} \ln\left(\frac{4(T+1)}{\delta}\right)\right)$$

Then, with probability at least $1 - \delta$, both of the following hold:

$$\sum_{t'} |P(t, t') - \hat{P}(t, t')| \leq \varepsilon, \quad \forall t \in \{0, \ldots, T\}$$

$$|R(t) - \hat{R}(t)| \leq \varepsilon, \quad \forall t \in \{0, \ldots, T\}$$

Proof. We proceed with applying a Chernoff-Hoeffding bound on each estimate and then taking a union bound over all the estimates. For each state t, we have N independent samples of the next state t'. Let X_i be the indicator random variable for the event that the ith sample transitioned to t'. Then, $\hat{P}(t, t') = \frac{1}{N}\sum_{i=1}^{N} X_i$ and $\mathbb{E}[\hat{P}(t, t')] = P(t, t')$. By the Chernoff-Hoeffding bound, we have

$$\Pr\left[|\hat{P}(t, t') - P(t, t')| \geq \frac{\varepsilon}{T+1}\right] \leq 2\exp\left(-\frac{2N\varepsilon^2}{(T+1)^2}\right)$$

Now, we know that $\sum_{t'} |P(t,t') - \hat{P}(t,t')| \le \varepsilon$ if $|P(t,t') - \hat{P}(t,t')| \le \varepsilon/(T+1)$ for all t'. Therefore, by a union bound over all t', we have

$$\Pr\left[\bigcup_{t'}\left[|\hat{P}(t,t') - P(t,t')| \ge \frac{\varepsilon}{T+1}\right]\right] \le 2(T+1)\exp\left(-\frac{2N\varepsilon^2}{(T+1)^2}\right)$$

Hence, by another union bound over all t, we have

$$\sum_{t'} |P(t,t') - \hat{P}(t,t')| \le \varepsilon, \quad \forall t \in \{0,\dots,T\}$$

with probability at least $1 - 2(T+1)^2\exp(-2N\varepsilon^2/(T+1)^2)$. As $N \ge \frac{(T+1)^2}{2\varepsilon^2}\ln\left(\frac{4(T+1)^2}{\delta}\right)$, this is $\ge 1 - \delta/2$.

Now consider the reward estimates. Let Y_i be the random variable representing the reward of the ith sample from state t. Then, $\hat{R}(t) = \frac{1}{N}\sum_{i=1}^{N} Y_i$ and $\mathbb{E}[\hat{R}(t)] = R(t)$. By the Chernoff-Hoeffding bound, using the fact that $Y_i \in [0, R_{\max}]$, we have

$$\Pr\left[|\hat{R}(t) - R(t)| \ge \varepsilon\right] \le 2\exp\left(-\frac{2N\varepsilon^2}{R_{\max}^2}\right)$$

Since there are $T+1$ states, by a union bound over all t, we have

$$|R(t) - \hat{R}(t)| \le \varepsilon, \quad \forall t \in \{0,\dots,T\}$$

with probability at least $1 - 2(T+1)\exp(-2N\varepsilon^2/R_{\max}^2)$. As $N \ge \frac{R_{\max}^2}{2\varepsilon^2}\ln\left(\frac{4(T+1)}{\delta}\right)$, this is $\ge 1 - \delta/2$.

Therefore, by a final union bound, we have that both properties hold with probability at least $1 - \delta$ as desired. ◀

B Proofs for Section 6 (Tail Slashing)

▶ **Lemma 12** (Monotonicity in κ). $\mathsf{OPT}(M^{\mathcal{R}^s_\kappa})$ *is non-increasing in* κ.

Proof. Fix $\kappa < \kappa'$ and let π be optimal for $M^{\mathcal{R}^s_{\kappa'}}$. On every transition where π skips a tail slot, the slashing penalty under κ' is at least as large as under κ. On transitions with no skip, the reward is identical. Since both MDPs are ergodic, the long-run average reward of π is well-defined and starting-state independent in each. Hence the average reward of π in $M^{\mathcal{R}^s_\kappa}$ is at least its average reward in $M^{\mathcal{R}^s_{\kappa'}}$, giving $\mathsf{OPT}(M^{\mathcal{R}^s_\kappa}) \ge \mathsf{OPT}(M^{\mathcal{R}^s_{\kappa'}})$. ◀

C Reward plots

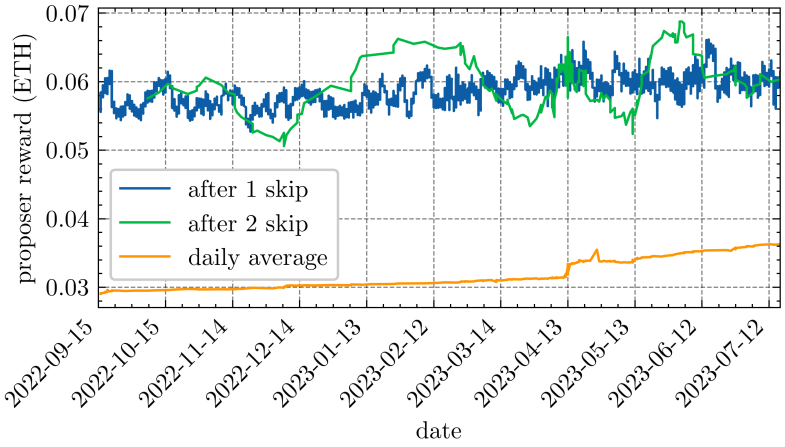


Figure 6 Consensus-layer rewards over time, excluding execution-layer rewards. Data from [29]. After a single missed slot, the next block earns roughly twice the consensus reward. Two or more consecutive missed slots show no significant further increase.

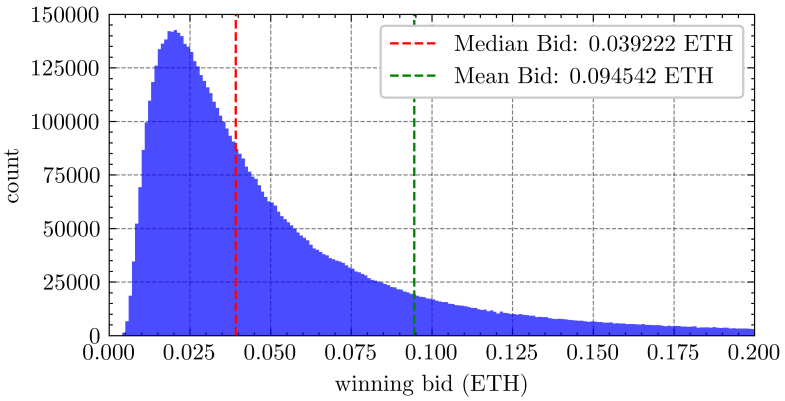


Figure 7 Histogram of bids from the MEV-Boost dataset [12].

To Wait or to Probe: Arbitrage Competition on High-Throughput Blockchains

Fei Wu* 
King's College London, London, UK

Burak Öz 
Flashbots, Berlin, Germany

Abstract

Maximal Extractable Value (MEV) on high-throughput blockchains can be captured through *targeted search*, where bots identify opportunities off-chain and submit route-committed transactions, or through *probabilistic search*, where bots submit repeated attempts that discover opportunities during on-chain execution. This distinction has direct implications for spam, blockspace consumption, and protocol revenue. We model how ordering granularity, fee floors, and opportunity-access shocks shape competition between these architectures. Using cyclic arbitrage data on Base from June 2025 to February 2026, we develop a trace-level classifier for search architectures and show that the resulting labels correspond to distinct execution behavior. In our sample, probabilistic search accounts for only 23% of arbitrage activity but produces 95% of spam and consumes 20% of Base gas. We test the model across three episodes: Flashblocks selects against broad on-chain probabilistic scanners; token-launch opportunity shocks temporarily revive probabilistic search; and higher fee floors select against probabilistic bots whose opportunity flow cannot sustain repeated attempts. After Base's configuration changes, protocol revenue shifts toward successful arbitrages and away from spam, probabilistic bots pay higher priority fees, and spam consumes a smaller share of blockspace.

2012 ACM Subject Classification Applied computing → Electronic commerce

Keywords and phrases Maximal Extractable Value, cyclic arbitrage, spam, Base

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.24

Related Version *Full Version*: `https://arxiv.org/abs/2606.00720` [35]

1 Introduction

A book collector offers $500 for a first-edition copy of *The Hobbit*. A local bookstore will receive the book at an unknown time and sell it for $400, with customers allowed to add a gratuity for priority service. The book becomes purchasable as soon as it enters the bookstore's inventory, but the public website updates only every ten minutes. Alice waits for the public update, then submits a purchase offer with a high gratuity. Bob repeatedly queries live inventory and buys if one check reveals the book first, imposing load on the store's system. Who wins the arbitrage? What would the bookstore prefer: targeted purchase attempts with higher gratuities, or repeated checks that might reveal opportunities earlier but consume system resources? Would the answer change if updates became more frequent or repeated checks became more costly?

This toy example captures a central design problem for Maximal Extractable Value (MEV) search on high-throughput blockchains: should searchers *wait* for information and submit targeted execution attempts, or *probe* the system with transactions whose profitability is resolved during execution? On Ethereum, MEV strategies are typically associated with targeted search: searchers identify and simulate an opportunity *off-chain*, then submit a

*Fei Wu performed work in part during an internship at Flashbots in Spring 2026.


© Fei Wu and Burak Öz;
licensed under Creative Commons License CC-BY 4.0

transaction committed to a specific execution route [7, 14, 26, 27, 34, 36]. On high-throughput blockchains, fast block production, low fees, and limited visibility into pending transactions can make a second architecture viable: searchers submit many transactions that read live state and resolve opportunity discovery *on-chain*, proceeding to further execution only when a profitable route is found [6, 11, 16, 28, 32].

We refer to these two architectures as *targeted search* and *probabilistic search*. Because opportunity discovery occurs at different stages, the architectures differ in their sensitivity to ordering rules and fees, and consequently in their attempt efficiency, blockspace usage, and contribution to protocol revenue. Probabilistic searchers submit speculative attempts that resolve opportunity discovery during execution; many fail without producing useful state changes and are therefore treated as *spam*. These attempts still consume blockspace and pay fees, so failed search can be both a congestion externality and a source of protocol revenue. Targeted searchers instead submit route-committed transactions against identified opportunities with potentially higher fees, but can also fail if the opportunity disappears before execution. The design question is therefore which architecture a protocol encourages, and how the resulting bot composition shapes blockspace consumption and protocol revenue.

This question has become increasingly relevant as high-throughput chains experiment with ordering rules and fee designs to reduce spam externalities and improve protocol revenue [4, 5, 21, 22]. However, the equilibrium effect of such changes is not obvious. Existing work characterizes probabilistic search timing [16] and models equilibrium spam volume as a function of block capacity, gas fee floor, and fee-based ordering [32], but typically focuses on a single search architecture. In this paper, we formalize the distinction between targeted and probabilistic search and study how protocol design shapes competition between them.

We develop a reduced-form equilibrium model of architecture competition in which ordering granularity, fee floors, and opportunity access shape the relative viability of targeted and probabilistic search (cf. Section 3). The model highlights two margins through which protocol changes affect spam: which architectures remain active, and how intensively active bots submit attempts.

We test these predictions on Base, which has documented spam activity [11, 28, 32] and underwent major configuration changes during our June 2025–February 2026 sample period [3]. We identify cyclic arbitrages and associated spam, and develop a trace-level classifier that distinguishes search architectures by the extent to which route discovery is observable before swap execution in successful arbitrage traces (cf. Section 4).

Our empirical analysis proceeds in three steps. We first validate that the classifier separates economically meaningful search architectures by showing that they differ in execution behavior, attempt efficiency, and spam production (cf. Section 5.1). We next study how arbitrage competition and the architecture relative viability evolve around three episodes: Flashblocks, the September surge, and the minimum-base-fee escalation (cf. Section 5.2-5.4). Finally, we aggregate bot-level responses to chain-level outcomes, including priority-fee revenue and blockspace consumption (cf. Section 5.5). We summarize our contributions as follows:

1. We formalize the distinction between *targeted* and *probabilistic* search architectures and develop a reduced-form equilibrium model in which ordering granularity, fee floors, and opportunity access shape the relative viability of the two architectures.
2. We develop a trace-level classifier that distinguishes the architectures through observable route discovery within successful arbitrages, and validate that the labels correspond to coherent transaction designs, spam behavior, and efficiency profiles.
3. We test the model's predictions on Base. Flashblocks selects against scan-intensive probabilistic bots and toward leaner survivors; a temporary opportunity shock revives probabilistic activity through short-lived entrants; and fee floor escalation selects against high-fee-exposure bots whose opportunity flow no longer supports repeated attempts.

4. We quantify the chain-level consequences. After configuration changes, priority-fee revenue shifts away from failed attempts toward realized arbitrages and higher-bidding probabilistic search attempts, while spam consumes a smaller share of blockspace.

1.1 Related work

MEV competition by targeted search has been documented on Ethereum Layer-1 [7,14,17,26, 27,29,33,34,36], alternate Layer-1s [31,38], Layer-2 networks [8], and cross-network [12,20,25]. Recent work documents probabilistic search on high-throughput blockchains: [6] introduces it as a viable strategy on Solana under faster block times and lower fees, and [11,28] show that similar strategies consume substantial blockspace on Base and Optimism, though [28] notes a different pattern on Arbitrum without formally distinguishing the two architectures. A related strand studies reverts and mechanisms for reducing failed MEV attempts [13,18,37]. Beyond empirical analysis, [32] derives equilibrium spam volume under varying block capacity, minimum gas price, and priority-fee ordering, while [16] analyzes the timing game underlying probabilistic search competition. These models focus on probabilistic search in isolation. Our paper distinguishes targeted and probabilistic search architectures, connects failed attempts to architecture choice, and studies how ordering granularity, fee floors, and opportunity access shift competition between them.

2 Preliminaries: Arbitrage Search Architectures

Section 1 introduced targeted and probabilistic search as two economic architectures. A central setting for our analysis is cyclic arbitrage, where a bot routes a token through two or more decentralized exchange (DEX) pools and returns to the starting token with a positive net profit. On high-throughput chains, bots often submit candidate transactions before knowing with certainty whether execution will be profitable. Failed attempts can therefore arise under both architectures: a route-committed transaction may find that the targeted opportunity has disappeared, while a probabilistic search transaction may fail to find any profitable route during execution. We use *failed attempts* as an umbrella term for arbitrage attempts that do not execute a profitable swap sequence. Within this category, *probes* terminate after state reads without executing swaps, while *reverts* revert during execution. Both consume execution resources and are commonly treated as spam. We next describe how these architectures appear at the *implementation level*, focusing on where route discovery occurs and what evidence is visible in transaction traces.

Off-chain discovery with direct execution. The cleanest implementation of targeted search is a transaction that proceeds directly from entry to swap execution with no pool-state read before the first swap.[1] The route, DEX venues, and input amount are embedded in the transaction calldata, indicating that the opportunity was discovered, simulated, and parameterized off-chain before submission.

Route-confined on-chain evaluation. In the second implementation, the bot also begins with a pre-selected route embedded in the calldata, but reads live state from the relevant pools before committing to swaps.[2] These reads can be interpreted as a profitability check:

[1] Consider example tx hash: `https://basescan.org/tx/0x1d93f62a009974da3ae575e7c623d88f09693b3e4e64ff0eabbac821f3fa091e`.
[2] Consider example tx hash: `https://basescan.org/tx/0x47a91fe3b9c31818f6c7f6699f6675297a8b926af48084e0c17cf8504a44df14`.

if the route remains profitable, the transaction proceeds to swap execution; otherwise, it terminates early or reverts. Importantly, the pre-swap reads are confined to the venues that the transaction later executes against. This pattern is economically closer to targeted search than to probabilistic search, but it is intrinsically ambiguous from traces alone: it could also reflect discovery over a confined venue set that happens to coincide with the executed venues.

Broad on-chain discovery. The canonical implementation of probabilistic search is a transaction that scans a wider set of pools than it ultimately executes on.[3] The bot reads state from candidate venues to identify which subset offers a profitable route, then executes the selected route, or terminates or reverts if no profitable route is found. This strict superset relation is the trace-level signature of route discovery during execution.

Bots read pool state through several trace-visible mechanisms, including direct pool reads, balance-based reads, revert-based quotes, and helper-contract calls. We use these mechanisms to identify pre-swap state reads in transaction traces (see full version [35]).

3 A reduced-form equilibrium model

This section presents a stylized equilibrium model of how ordering granularity, transaction costs, and opportunity access shape the relative competitiveness of arbitrage bot architectures. Entry adjusts until expected profits are driven to zero.

Ordering regimes and bot architecture. We consider two ordering regimes $r \in \{B, F\}$, where B denotes the baseline regime and F denotes a finer ordering regime. The relevant ordering unit is a regular full block in regime B, but a smaller sub-block slice in regime F.

We next describe two representative bot architectures. Following the implementation-level vocabulary of Section 2, we refer to them as *off-chain discovery*, denoted by O, and *on-chain discovery*, denoted by C, corresponding to targeted and probabilistic search, respectively. Bots compete over candidate arbitrage opportunities. We assume that under the chain's ordering regime, an opportunity is exhausted by the first valid claimant in the earliest profitable ordering unit.

Same-slice competitiveness. On-chain discovery transactions perform state reads to evaluate candidate routes, so they consume more blockspace than route-committed off-chain transactions targeting the same opportunity. Under finer ordering, early sub-block slices have smaller effective capacity, so heavier on-chain transactions are more likely to be displaced, allowing a lean off-chain transaction to capture the opportunity in the earlier slice.[4]

We summarize this disadvantage by a reduced-form *same-slice competitiveness* function

$$\eta_r \in [0, 1), \qquad r \in \{B, F\},$$

where η_r is the probability that an on-chain discovery transaction remains competitive in the same earliest profitable slice as an off-chain discovery transaction in regime r. The key restriction is $\eta_F < \eta_B$: on-chain discovery is less likely to remain competitive in the same profitable slice under a finer ordering regime than under the baseline regime.

[3] Consider example tx hash: `https://basescan.org/tx/0xc020e534c39e203e9f3a42ea5f89ca023e486c38e15ca55e2a183ead1f4c885d`.

[4] Under Flashblocks on Base [2], transactions exceeding 14M gas cannot fit in the first Flashblock.

Opportunity access and free-entry payoffs. The two architectures differ in opportunity access. An off-chain discovery transaction may encode several pre-identified routes, but its candidate route set is fixed before submission. An on-chain discovery transaction instead reads the current state and can select among a state-contingent set of candidate routes during execution. We summarize this potential breadth advantage through architecture-specific opportunity-access rates. Let λ_i denote the expected number of candidate opportunities per model period accessible to architecture $i \in \{O, C\}$. With average gross value V conditional on successful capture, $\lambda_i V$ is the gross opportunity value flow accessible to architecture i, before competition and transaction costs.[5] We allow $\lambda_C \geq \lambda_O$ to represent the potentially broader access of on-chain discovery.

Let m_i denote the effective mass of independent competing strategies using architecture i. Here, "mass" is the standard continuum-entry measure of participation rather than a literal count of unique addresses or contracts observed on-chain. In a fraction η_r of competitive states, an on-chain discovery transaction remains in the earliest profitable ordering slices and competes with off-chain discovery transactions. Conditional on this case, the opportunity is allocated uniformly across the total active mass $m_O + m_C$. In the remaining states, the on-chain discovery transaction is displaced, and only off-chain discovery bots compete, uniformly across m_O. Each bot incurs an expected cost $K_i(b) > 0$ per modeled period, where $b > 0$ is the minimum base fee. Applying this reduced form of the sharing rule to each architecture's accessible opportunity flow yields the payoffs below.

The expected payoff of an off-chain discovery bot in regime r is

$$\Pi_O^r(m_O, m_C; b) = \eta_r \frac{\lambda_O V}{m_O + m_C} + (1 - \eta_r) \frac{\lambda_O V}{m_O} - K_O(b).$$

The first term is the off-chain bot's share when both architectures remain competitive in the earliest profitable slice; the second is its share when on-chain discovery is displaced; the final term is its expected cost. An on-chain-discovery bot earns opportunity value only in the fraction η_r of states in which its heavier transaction remains competitive in the earliest profitable slice:

$$\Pi_C^r(m_O, m_C; b) = \eta_r \frac{\lambda_C V}{m_O + m_C} - K_C(b).$$

A competitive equilibrium in regime r is a pair $(m_O^{*,r}, m_C^{*,r})$ such that no bot architecture earns positive expected profit, and any active bot architecture earns zero expected profit:

$$\Pi_i^r(m_O^{*,r}, m_C^{*,r}; b) \leq 0 \quad \text{for } i \in \{O, C\}, \quad m_i^{*,r} > 0 \;\Rightarrow\; \Pi_i^r(m_O^{*,r}, m_C^{*,r}; b) = 0.$$

Thus, active architectures earn zero profit, and inactive architectures cannot enter profitably.

▶ Remark 1 (Blockspace competitiveness versus opportunity creation). The model isolates the blockspace competitiveness channel rather than the opportunity-creation channel. Finer ordering may also change the value or arrival rate of arbitrage opportunities by reducing the order flow aggregated into each interval. In our notation, such effects would enter through V or λ_i. We hold V and λ_i fixed to focus on architecture composition.

For simplicity, we next focus on equilibria with active off-chain discovery architecture. Other equilibrium configurations can be derived analogously.

[5] We assume V to be common across architectures. In reality, off-chain discovery may extract more value through optimized off-chain simulation; this asymmetry would only reinforce the qualitative results.

Define the effective value-cost ratios

$$\kappa_O(b) := \frac{\lambda_O V}{K_O(b)}, \qquad \kappa_C(b) := \frac{\lambda_C V}{K_C(b)}.$$

We assume that, over the parameter region of interest, $\kappa_C(b) > \kappa_O(b) > 0$. The condition is imposed to make coexistence possible before applying the ordering competitiveness penalty η_r. If the condition fails, on-chain discovery is already dominated on the branch considered here. The observed coexistence of both architectures in our sample motivates studying this parameter region, but we do not estimate the inequality structurally.

▶ **Proposition 2** (Within-regime equilibrium). *Fix $b \geq 0$, $r \in \{B, F\}$, and $\kappa_C(b) > \kappa_O(b) > 0$. On the equilibrium branch with active off-chain discovery, the unique equilibrium is*

$$(m_O^{*,r}, m_C^{*,r}) = \begin{cases} \left(\dfrac{(1-\eta_r)\kappa_O(b)}{1-\kappa_O(b)/\kappa_C(b)}, \dfrac{\eta_r\kappa_C(b) - \kappa_O(b)}{1-\kappa_O(b)/\kappa_C(b)} \right), & \eta_r\kappa_C(b) > \kappa_O(b), \\ (\kappa_O(b), 0), & \eta_r\kappa_C(b) \leq \kappa_O(b). \end{cases}$$

Thus, on-chain discovery survives iff $\eta_r\kappa_C(b) > \kappa_O(b)$, i.e., its access-adjusted value-cost advantage is large enough to overcome its ordering disadvantage.

Next, we show how a decline in same-slice competitiveness, from η_B to η_F, shifts equilibrium participation away from on-chain discovery and toward off-chain discovery.

▶ **Theorem 3** (Finer ordering reduces the competitiveness of on-chain discovery). *Fix $b \geq 0$, and suppose $\eta_F < \eta_B$ and both regimes are on the coexistence branch. Then*

$$m_C^{*,F}(b) < m_C^{*,B}(b), \qquad m_O^{*,F}(b) > m_O^{*,B}(b).$$

Switching to the finer ordering regime reduces on-chain participation and increases off-chain participation on the coexistence branch.

To capture selection within on-chain discovery, we allow subtypes to differ in their *per-attempt scan footprint*, which measures how much venue-search work a single attempt performs. It is distinct from *attempt intensity*, i.e., how many attempts a bot sends. A larger scan footprint may broaden the candidate set evaluated within an attempt, but also makes the attempt heavier, reducing same-slice competitiveness under finer ordering and raising gas cost. This trade-off creates a selection margin within the on-chain discovery category.

▶ **Proposition 4** (Finer ordering selects against high-footprint on-chain discovery). *Under the regularity conditions defined in Appendix A.2 in the full version [35], let on-chain discovery subtypes be indexed by a per-attempt scan-footprint parameter x, and let $x_r^*(b)$ denote the largest viable footprint in regime r. Then $x_F^*(b) < x_B^*(b)$, so finer ordering selects against more scan-intensive on-chain-discovery strategies.*

This selection result implies that, among surviving on-chain discovery bots, observable scan-footprint measures, such as the number of pre-swap state reads, should weakly fall under finer ordering. We formalize this prediction in Appendix A.2 in the full version [35].

▶ Remark 5 (Off-chain discovery reliability under finer ordering). Finer ordering also shortens the interval between opportunity observation and execution, reducing the stale-state risk faced by off-chain discovery. In the model, this channel would raise the effective value-cost ratio of off-chain discovery by reducing failed attempts. This stale-state channel and the

blockspace-competitiveness channel both predict a shift away from broad on-chain discovery toward route-committed execution, so the composition test in Section 5.2 cannot separately identify them. However, the stale-state channel more directly predicts improved off-chain attempt efficiency. The rise in off-chain discovery success rates in Table 4 is therefore suggestive evidence for this margin, although not separately identified from changes in blockspace competition.

The theorems above isolate the blockspace-competitiveness channel under a finer ordering regime. We then turn to the cost channel. A higher minimum base fee raises the cost of every attempt, so it selects against architectures with higher fee exposure.

▶ **Proposition 6** (A higher minimum base fee can disproportionately hurt on-chain discovery)**.** *Define architecture i's fee exposure and on-chain discovery effective advantage as*

$$\chi_i(b) := \frac{\partial \log K_i(b)}{\partial b}, \qquad \rho(b) := \frac{\kappa_C(b)}{\kappa_O(b)}.$$

If $\chi_C(b) > \chi_O(b)$, then $\rho(b)$ is strictly decreasing in b. Equivalently, the coexistence condition $\eta_r \kappa_C(b) > \kappa_O(b)$ becomes harder to satisfy as the fee floor rises.

The compositional prediction requires asymmetric fee exposure. If the fee floor scaled every architecture's cost by the same proportional factor, then $\chi_C(b) = \chi_O(b)$, and a higher fee floor would reduce total entry without changing architecture composition. Empirically, this condition is consistent with on-chain discovery consuming more gas per attempt and sending more attempts per successful arbitrage (cf. Section 5.1).

The same logic applies within on-chain discovery: holding opportunity access and same-slice competitiveness fixed, higher fee floors reduce the advantage of on-chain subtypes with more fee-sensitive cost functions. We formalize this in the full version [35].

▶ Remark 7 (What fee exposure means empirically). Fee exposure depends not only on per-attempt scan footprint, but also on attempt intensity. Thus, higher fee floors should be interpreted as raising the hurdle rate for repeated probing, not as mechanically eliminating the most scan-intensive bots. High-exposure bots may remain viable if their opportunity flow is valuable enough, while lower-exposure bots may exit if their opportunity flow is weak.

The blockspace and cost channels above hold the opportunity environment fixed. Token launches and fresh DEX deployments can temporarily raise λ_C by creating routes that off-chain discovery architectures have not yet indexed or parameterized. Such shocks can revive on-chain discovery participation even under otherwise unfavorable policy conditions.

▶ **Proposition 8** (Opportunity-access shocks can revive on-chain discovery)**.** *On the coexistence branch, $m_C^{*,r}$ is strictly increasing and $m_O^{*,r}$ strictly decreasing in λ_C. There exists a viability threshold $\bar{\lambda}_C(r, b)$ such that on-chain discovery is active iff $\lambda_C > \bar{\lambda}_C(r, b)$, and this threshold is strictly increasing in policy frictions. Hence, a temporary increase in λ_C can revive on-chain discovery even when the policy conditions render it inactive.*

▶ Remark 9 (Participation and observed spam). Free entry links the model's equilibrium masses to empirical architecture prominence: when a policy change lowers an architecture's expected payoff, the equilibrium supports a smaller mass of participants using that architecture. Empirically, lower architecture mass can be reflected in fewer active bots or a lower architecture share of arbitrage count or volume. Aggregate spam, however, also depends on attempt intensity per bot. Thus, spam may not fall proportionally if surviving or entering bots submit more attempts.

4 Data and Methodology

We use Base as a case study to measure arbitrage competition between search architectures and their response to protocol changes. Base is a natural setting because it has substantial DEX activity and spam volume [11, 28, 32], and because it implemented major configuration changes, most notably Flashblocks and subsequent increases in the minimum base fee [3]. This section describes the data construction and bot-architecture classification methodology.

4.1 Data collection

We collect cyclic arbitrages and associated spam activity on Base from June 1, 2025, through February 28, 2026, covering the periods of major configuration changes on Base [3].

Successful cyclic arbitrages. We extend the cyclic arbitrage detection method of [28]. Using a custom Dune query, we detect 21,374,434 cyclic arbitrages by 4,365 unique arbitrage bot addresses over the observed period. For each transaction, we collect gas usage, fees, USD notional volume, and the realized arbitrage route, including traded tokens and DEX pools.

Failed arbitrage attempts (spam). We separately query reverted and probe transactions that are attributed to the same arbitrage bot addresses. Importantly, the bot does not need to interact with the DEX pool contract to read its state. To capture this broader class of probing behavior, we classify as probes successful transactions that emit no logs from external contracts other than the bot itself and carry ETH value ≤ 1 WEI.[6]

This yields 985,559,360 spam transactions from these cyclic arbitrage bots over the observed period, including 35,653,523 reverts and 949,905,837 probes. Because transaction-level analysis of this full spam set is infeasible, we aggregate it at the bot-week level: for each bot in each week, we collect the count of probes and reverts, corresponding total gas usage and fees, and share of transaction count and gas usage out of the chain's throughput.

4.2 Bot architecture identification

Transaction-level labeling. Because failed-attempt outcomes can arise from any of the three implementation patterns, distinguishing architectures requires trace-level evidence rather than transaction outcomes alone. We identify architecture from successful arbitrage transactions, where the executed route is observable. The objective is to distinguish whether a transaction arrives with a route already committed, evaluates only the route it eventually executes, or scans multiple candidate venues before choosing an execution route.

Since tracing all 21.37M arbitrages is infeasible, for each bot in each week during the observed period, we randomly sample up to 50 arbitrage transactions, yielding 359,752 sampled transactions across 4,365 bots and 39 weeks. We label these transactions using a trace-level classifier applied to `debug_traceTransaction` call traces with an Alchemy RPC node. The classifier distinguishes three transaction-level labels:

`OFFCHAIN_DIRECT` transactions show no meaningful pre-swap state read, or use only helper contract or revert-based quoting on the eventual execution route.

`ONCHAIN_EVAL` transactions perform on-chain evaluation before execution, but that evaluation is confined to the route that is eventually executed.

`ONCHAIN_BROAD_SCAN` transactions evaluate venues beyond the eventual execution set before committing to the winning route.

[6] Simple ETH transfer also emits no external logs. Certain arbitrage bots typically attach 1 WEI to their transactions, such as `0xA4B9eDB99806bFB8BbEE89D13F4c296DAF225dEE`. Thus, we set 1 WEI as a buffer.

The classifier's core is to identify if state reads exist and compare the venues read before swap with the venues used in the successful route. Reads of executed venues are treated as route-confined evaluation; reads of non-executed venues are treated as route discovery.

From transaction labels to bot architecture. We aggregate transaction-level labels into weekly bot architectures using the implementation patterns introduced in Section 2. A bot-week is labeled *on-chain discovery* if its sample contains at least one manually validated `ONCHAIN_BROAD_SCAN` transaction. It is labeled *off-chain discovery* if it contains `OFFCHAIN_DIRECT` behavior and no credible broad-scan evidence after manual validation. It is labeled *on-chain evaluation* only if all sampled transactions are `ONCHAIN_EVAL`. This asymmetry reflects the fact that direct execution and broad venue expansion are clear endpoint behaviors, while confined on-chain evaluation can arise under either off-chain discovery with on-chain re-validation or narrow on-chain discovery.

This rule assumes architecture is persistent within a bot-week, rather than switching transaction by transaction. Broad on-chain scanning requires different execution logic from route-committed execution, so genuine changes to off-chain architecture are more plausibly associated with redeployment or a new bot identity. Because labels are assigned each week, the same bot can still receive different labels across weeks if its observed behavior changes over longer horizons. We then project weekly labels onto the full arbitrage dataset and weekly spam datasets, producing a unified bot-week panel with architecture, successful arbitrage activity, and aggregate spam behavior.

Mixed `ONCHAIN_BROAD_SCAN` and `OFFCHAIN_DIRECT` cases within the same bot-week are manually reviewed to exclude false `ONCHAIN_BROAD_SCAN` classifications. Full classifier details, edge cases, and manual-validation rules are specified in the full version [35].

Scope and limitations. We infer architecture from successful transactions because failed attempts often do not reveal a realized execution route, making route-confined evaluation and broad scanning indistinguishable. Projecting the weekly label onto spam therefore assumes that successful and failed attempts within a bot-week are generated by the same underlying architecture. While this cannot be easily validated transaction by transaction, the distinct architecture behavior we identified in the later sections provides indirect support for the projection. The classifier is conservative in two further ways: singleton PoolManager designs can hide venue expansion, and sampling up to 50 transactions may miss rare direct-execution or broad-scan transactions. We quantify these effects in Appendix B.2 in the full version [35].

5 Empirical Results

We organize the empirical results around the model's predictions and Base configuration changes during the sample period [3]. We first establish that the three architecture labels correspond to distinct successful arbitrage and spam profiles. We then study how the prominence of each architecture changes around three empirical episodes: the introduction of Flashblocks, the September on-chain-discovery surge, and the escalation of the minimum-base-fee. Finally, we quantify how each architecture's response to these episodes translates into chain-level outcomes, including priority-fee payments and blockspace consumption.

Table 1 Bot activity by weekly architecture label, June 1, 2025–February 28, 2026. Bot counts are unique addresses observed with each label at least once; categories are not mutually exclusive because labels can vary across weeks.

Bot label	Bots	Arbs	Spam	Reverts	Probes	Success rate	Median / P95 avg. spam tx gas	Avg. cost per success
Off-chain discovery	1,634	7.67M	23.63M	16.86M	6.78M	42.51%	352K / 1.02M	0.000095 ETH
On-chain evaluation	798	8.83M	27.63M	12.51M	15.12M	18.49%	64.8K / 371K	0.000045 ETH
On-chain discovery	2,049	4.87M	934.30M	6.29M	928.01M	4.58%	277K / 1.39M	0.000253 ETH

5.1 Bot architecture characteristics

We begin by documenting how the three bot architectures differ in their observable behavior, both in successful arbitrages and in spam during our sample period. These differences validate that the classifier separates economically distinct architectures rather than spurious trace clusters, and they support the model's key primitives: on-chain discovery bots operate over broader venue sets with lower success rates, and have higher fee exposure.

Spam character. The three architectures produce strikingly different spam activities. Table 1 summarizes full-sample activity by weekly bot label. The table reports the number of bots, successful arbitrages, spam transactions, reverts, probes, average bot-level success rate, spam transaction gas usage, and average cost per arbitrage opportunity. Success rate is computed at the bot level as successful arbitrages divided by total attempts, where total attempts include successful arbitrages and spam transactions. Spam gas is computed as each bot-week's average gas per spam transaction and then summarized across bot-weeks. Average cost per successful arbitrage is total gas cost across successful and failed attempts divided by successful arbitrages.

Off-chain discovery bots produce the least spam and have the highest average success rate at 42.51%. The spam they produce consists predominantly of reverts rather than probes. Because routes are pre-committed, their transactions typically proceed directly to swap execution and revert only when profitability conditions fail close to the end of execution. The resulting reverts traverse part or all of the swap logic, which explains their highest median average spam gas usage (352K) of the three architectures. Their average cost per successful arbitrage is 0.000095 ETH – moderate despite heavy per-spam gas usage, because relatively few failed attempts accompany each successful arbitrage.

On-chain evaluation bots produce spam split roughly evenly between reverts and probes, which we interpret as an implementation choice rather than a meaningful architectural distinction: both outcomes terminate a failed attempt after verifying a confined route set. Their evaluation is lightweight and confined to a narrow candidate venue set, so their average spam gas usage is the lowest of the three architectures (median 64.8K, P95 371K). Their average cost per captured opportunity is correspondingly the lowest at 0.000045 ETH, reflecting both cheap spam and a modest success rate.

On-chain discovery bots generate the largest spam activity with 934.30M total spam transactions, almost all of which are probes. This matches their architecture: the bot reads state across many venues to discover whether an opportunity exists at all, and most such reads find none, leaving a probe transaction on-chain. The median average spam gas usage (277K) is higher than that of on-chain evaluation, consistent with broader venue scanning, and the P95 (1.39M) reflects a tail of very wide route searches. Their average cost per opportunity is 0.000253 ETH, much higher than other architectures – the compound effect of heavy spam transactions and a large number of them per successful arbitrage. The average success rate is correspondingly low at 4.58%.

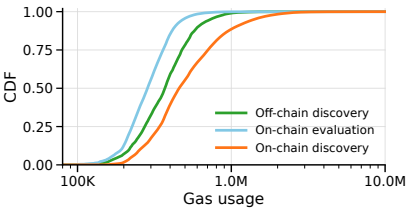


Figure 1 Arbitrage gas usage CDF.

Table 2 Sampled arbitrage trace-level metrics.

	Bot-level architecture		Transaction-level footprint	
Bot label	Median calldata length	Average / Median venue commitment	Median pre-reads	oracle usage
Off-chain disc.	516	0.607 / 0.971	0	22.61%
On-chain eval.	150	0.535 / 0.886	3	4.61%
On-chain disc.	91	0.184 / 0.000	40	0.08%

Successful arbitrage fingerprints. The architectural distinction also shows up in how successful arbitrage transactions are structured. Figure 1 plots the cumulative distribution function of successful arbitrage gas usage by bot label. Successful arbitrages by on-chain discovery bots are more gas-intensive, consistent with broad venue scanning making these transactions heavier. We also observe that off-chain discovery transactions use more gas than on-chain evaluation transactions. This does not contradict their route-committed classification: off-chain discovery bots may execute more complex routes, use flash loans,[7] or route through venues such as Balancer, where execution may involve a vault and oracle calls.

Table 2 reports complementary architecture and footprint measures from the sampled successful arbitrage transactions. Calldata length and venue commitment are summarized at the bot level to avoid overweighting high-activity bots; pre-execution reads and oracle usage are summarized at the transaction level because they measure per-transaction footprint. Venue commitment is a conservative address-level proxy for route pre-commitment, defined as the fraction of executed venue addresses embedded in calldata.

Off-chain discovery bots have the longest median calldata and the highest venue commitment, and their transactions have a median of zero pre-execution reads, consistent with pre-committed routes carried in the transaction itself and direct execution behavior. On-chain discovery bots show the opposite pattern with the shortest median calldata, zero median venue commitment, and 40 median pre-execution reads in their transactions, indicating that the route is resolved during execution through dense state reads across many venues. On-chain evaluation bots sit between the two, with modest pre-reads against the confined set of venues they subsequently execute on.

Time-series overview. We next document the time-series evolution of the prominence of each architecture during the observed period. Figure 2 presents the weekly successful arbitrage, USD volume, and spam activity by architecture. Across the full sample, on-chain discovery bots account for 22.8% of successful arbitrages and 28% of volume, but produce nearly 95% of the total spam activity. Figure 3A plots the weekly count of active bots by architecture. In most weeks, 30–50 active on-chain discovery bots generate roughly 20M spam transactions per week. By contrast, off-chain discovery and on-chain evaluation have larger active populations, but generate substantially less spam while accounting for a larger share of successful arbitrages. Figure 3B–C tracks active bots and volume shares by architecture. The series highlights three episodes we study below: Flashblocks on July 7, 2025; a temporary on-chain discovery surge in early September 2025; and the minimum-base-fee escalation beginning December 4, 2025. These episodes correspond to the model's three comparative statics: finer ordering, opportunity-access shocks, and fee-floor increases.

[7] Consider example tx hash: `https://basescan.org/tx/0x05d5aeccc0ecc11013b62481ea92f47213c200e9ec6bad0c72598d306be40526`.

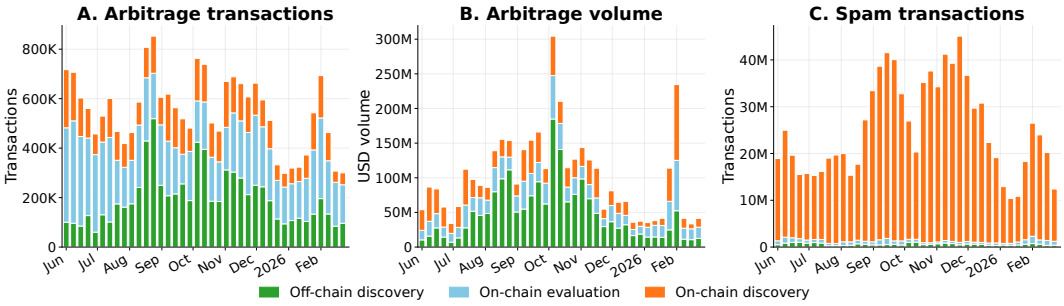


Figure 2 Weekly successful arbitrage transaction count, USD volume, and spam transaction count by bot architecture label.

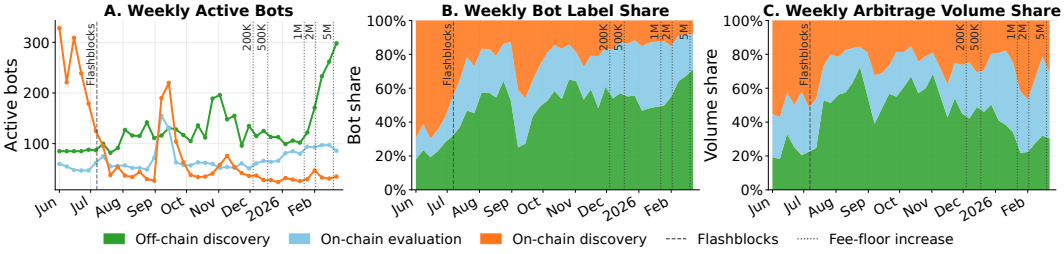


Figure 3 Weekly number of active bots per label, weekly active bot label share, and weekly volume share per bot label. Vertical lines mark configuration changes of Flashblocks and fee floors.

5.2 Flashblocks

The Flashblocks activation on July 7, 2025, introduced 200ms sub-block pre-confirmations within Base's 2-second block cycle, and each Flashblock has an incrementally increasing space capacity [2], substantially finer than the full-block ordering. Our model predicts two patterns around this transition: a decline in on-chain discovery participation (cf. Theorem 3); selection against more complex on-chain discovery strategies (cf. Proposition 4) and towards simpler on-chain discovery survivors. We empirically validate these predictions using a four-week symmetric event window around the rollout: pre weeks June 8 – July 6, 2025, post weeks July 13 – August 10, 2025, with the event week itself excluded.

Population and volume shift. The composition of active bots changes sharply after the Flashblocks introduction. The number of total distinct active bots declines by 53%, falling from 983 in the pre-window to 462 in the post-window. The contraction is concentrated almost entirely in on-chain discovery architecture: its share of active bots falls from 63.1% to 25.1%, while off-chain discovery rises from 23.4% to 46.7% and on-chain evaluation rises from 13.4% to 28.2%. As illustrated in Figure 3A, the on-chain discovery population peaks above 300 weekly active in mid-June, declines through the pre-window, and stabilizes around 30 weekly active bots in the post-window. The off-chain discovery population, in contrast, expands steadily from roughly 85 to over 150 weekly active bots by early August.

Figure 3C presents a consistent story in terms of arbitrage volume. On-chain discovery's share of weekly arbitrage USD volume falls from roughly 50-60% pre-Flashblocks to around 30% post-Flashblocks and remains at that level for the remainder of the observation window. Off-chain discovery expands from roughly 20% to over 50% of volume, and on-chain evaluation remains broadly stable. These movements match Theorem 3: finer ordering reduces participation from on-chain discovery architecture and increases off-chain participation.

Table 3 On-chain discovery cohort composition around Flashblocks. Bot-level metrics first summarize each bot within the relevant window and then summarize across bots. Transaction-level metrics summarize sampled arbitrage transactions and spam bot-week medians within the same four-week pre- and post-event windows.

Cohort	Bots	Bot-level architecture			Transaction-level footprint		
		Median calldata length	Average venue commitment	Median venue commitment	Median pre-reads	Median arbitrage tx gas	Median of avg. spam tx gas
Selected out (pre only)	741	36	0.044	0.000	57	855K	671K
Entrants (post only)	120	244	0.258	0.000	46	630K	272K
Survivors (pre → post)	37	163 → 302	0.345 → 0.340	0.000 → 0.000	39 → 12	662K → 437K	226K → 103K

A natural concern is that on-chain discovery was already declining before Flashblocks, falling from 309 weekly active bots by June 15 to 125 by July 6. This decline likely reflects the unwinding of a May-June opportunity cycle around AI-agent tokens on Base.[8] However, a purely demand-side explanation is less plausible: on-chain discovery's volume share remains stable during the pre-window, and around Flashblocks, aggregate arbitrage activity rebounds (cf. Figure 2), yet on-chain discovery bots continue to contract from 179 in the week before Flashblocks to 38 three weeks after.

Selection against on-chain discovery. The composition shift around Flashblocks is driven primarily by selection rather than within-bot label switching: among the 174 bots active in both pre- and post-windows, 167 retain the same label, so the shift mainly reflects differential exit and replacement. The selection margin is sharpest within on-chain discovery. To characterize it, we compare three cohorts within the on-chain discovery architecture: bots active only in the pre-window (*selected out*), only in the post-window (*entrants*), and in both windows under the same label (*survivors*). Table 3 reports the bot-level architecture and transaction trace-level footprint metrics calculated from sampled arbitrages for three cohorts, using only their bot-weeks labeled as on-chain discovery.

The 741 selected-out on-chain discovery bots have the highest scan footprint in the dataset. Their median bot-level calldata length is only 36 bytes, median venue commitment is zero, and their transaction-level footprint is the heaviest, with 57 median pre-reads and the highest gas usage for both arbitrage and spam transactions. In other words, the bots most heavily dependent on short-calldata, low-commitment, high-read scanner logic are precisely the ones that vanish after Flashblocks.

Relative to the selected-out bots, the entrant cohort is markedly less scan-intensive – longer calldata, higher (average) venue commitment, fewer pre-reads, and lower gas usage on successful arbitrages and spam transactions. The survivors are simpler still. Even in the pre-window, surviving on-chain discovery bots exhibit the least search behavior across the three cohorts, suggesting that survival is concentrated among bots whose on-chain search is already relatively lean. Moreover, the survivors become even less scan-intensive after Flashblocks, indicating within-bot adaptation toward leaner scanning.

The same selection and adaptation patterns hold in a two-week window around Flashblocks, suggesting that the result is not driven by the choice of event window or by later post-event weeks. By contrast, the corresponding architecture changes for off-chain discovery and on-chain evaluation population are modest around Flashblocks (see Appendix C.1 in the full version [35]). These results confirm Proposition 4 that finer ordering selects against on-chain discovery with higher per-attempt scan-footprint and towards less scan-intensive ones.

[8] Examples include KTA, RDAC, VIRTUAL, and AIXBT. These provide context for a contemporaneous opportunity cycle; the empirical results do not depend on this attribution.

Table 4 Attempt efficiency and spam production around Flashblocks. Success rates and gas cost per successful arbitrage are bot-level averages within each window. Spam per active bot-week normalizes spam count by the number of labeled bot-week observations. Bot counts are not mutually exclusive because labels may vary across weeks.

Bot label	Success rate pre → post	Gas cost / success pre → post	Total spam txs pre → post	Spam per active bot-week	Active bots pre → post
Off-chain disc.	35.8% → 43.6%	0.000140 → 0.000113	3.73M → 1.78M (-52.3%)	11.13K → 4.83K	134 → 207
On-chain eval.	26.1% → 23.1%	0.000015 → 0.000027	3.66M → 2.54M (-30.7%)	18.70K → 10.45K	86 → 113
On-chain disc.	5.7% → 3.8%	0.000293 → 0.000350	68.43M → 70.50M (+3.0%)	72.49K → 314.71K	782 → 157

Attempt efficiency and spam concentration. The selection documented above concerns the transaction-level footprint of arbitrage attempts. It does not imply that aggregate spam activity should fall mechanically, because spam also depends on how many failed attempts each bot sends. Table 4 compares spam production and attempt efficiency by architecture over the four-week pre- and post-Flashblocks windows.

The spam response is highly asymmetric. Off-chain discovery and on-chain evaluation both reduce total spam and spam per active bot-week. On-chain discovery, however, remains spam-intensive: despite active bots falling from 782 to 157, total spam rises from 68.43M to 70.50M, and spam per active bot-week rises from 72.49K to 314.71K. Because on-chain discovery dominates spam in both windows, aggregate spam is nearly flat.

Attempt efficiency moves in a similar direction. Flashblocks improves attempt efficiency for off-chain discovery: average success rate rises from 35.8% to 43.6%, and gas cost per successful arbitrage falls from 0.000140 to 0.000113 ETH. This is consistent with reduced stale-state risk between off-chain opportunity discovery and execution (cf. Remark 5). On-chain discovery moves in the opposite direction, with lower success rates and higher gas cost per success. Despite fewer failed attempts, the attempt efficiency of on-chain evaluation declines, as their successful arbitrages fall (1.403M pre-window → 0.855M post-window).

The persistence of on-chain discovery spam reflects a response in attempt intensity among the bots that remain. Flashblocks selects against many broad scanner bots and leaves less scan-intensive survivors, but it does not directly price attempt frequency. By shortening the window in which each opportunity remains available for on-chain discovery, finer ordering can lower per-attempt yield; surviving discovery bots can respond by probing more frequently. In this sense, Flashblocks changes who participates in on-chain discovery and narrows the form of surviving search, but does not by itself eliminate high-frequency probing.

Summary. The Flashblocks event study supports the model's central mechanism: finer ordering reduces the competitiveness, and therefore the participation, of on-chain discovery, especially among the most scan-intensive bots. The dominant margin is selection: broad scanners exit, same-label survivors are already narrower and simplify further, and the market shifts toward route-committed off-chain discovery. Off-chain discovery becomes more attempt-efficient after Flashblocks, consistent with the stale-state channel: shorter ordering intervals make route-committed execution more reliable. However, this architectural selection does not translate into a proportional reduction in total spam. Aggregate spam remains nearly flat because on-chain discovery spam becomes concentrated among fewer, higher attempt-intensity entrants and survivors. Thus, Flashblocks reduces broad on-chain discovery and improves off-chain execution efficiency, but its effect on aggregate spam is partially offset by higher attempt intensity among the remaining on-chain discovery bots.

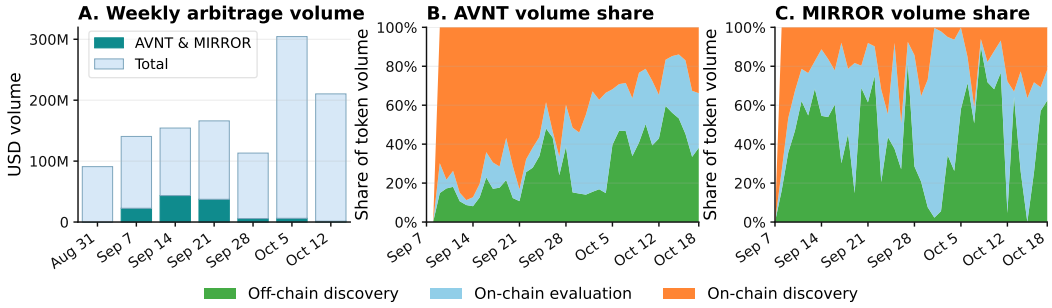


Figure 4 AVNT and MIRROR arbitrage activity around the surge window. Panel A shows weekly AVNT/MIRROR volume relative to total volume, with the x-axis showing the week-start date. Panels B and C show AVNT and MIRROR daily volume share, respectively, by bot architecture.

5.3 The September surge

The post-Flashblocks on-chain discovery population stabilizes at roughly 30-50 weekly active bots through August 2025. However, this number spikes sharply in early September: active on-chain discovery bots rise to 220 in the week beginning September 14, before quickly falling back to 62 by September 28 and stabilizing around 35 in early October (cf. Figure 3A). A smaller spike also appears among on-chain evaluation, while off-chain discovery remains comparatively stable. Because the aggregate time series does not show a broad, sustained market-wide expansion (cf. Figure 2), we interpret the episode as a localized entry wave around newly attractive venues rather than a broad expansion in arbitrage opportunities. For the remainder of this section, we refer to the four weekly buckets from September 7–13 through September 28–October 4 as the September surge window.

A short-lived entrant wave. The entrant composition shows that the surge is bimodal: a few serious entrants and a much larger mass of short-lived triers. Across the September surge window, 504 unique on-chain discovery bots are active, of which 464 (92.1%) are both first and last seen within these weeks. Most are active only briefly: 426 bots appear only in one week, and 470 appear in at most two weeks. The same pattern appears in transaction intensity: median transient on-chain discovery bots execute only 29 surge-window arbitrages, compared with 262.5 for non-transient bots (see Appendix C.2 in the full version [35]).

AVNT/MIRROR opportunity wave. The entrant wave is centered on two newly launched tokens: AVNT, launched on September 9 [1], and MIRROR, launched on September 8 [19]. Figure 4A shows that AVNT and MIRROR form a visible share of arbitrage volume from September 7 to September 28, and account for much of the contemporaneous increase in total volume. After September 28, AVNT/MIRROR arbitrage volume falls quickly, coinciding with the disappearance of the temporary on-chain discovery entrant population.

AVNT and MIRROR account for a large share of arbitrage activity during the surge window. From September 7 to October 4, AVNT or MIRROR appears in 420,541 arbitrages (19.3% of all arbitrages) and $109.1M in volume (19.0%). The concentration is much stronger within on-chain discovery: 42.8% of arbitrages and 50.2% of volume involve AVNT/MIRROR, spanning 412 of the 504 active on-chain discovery bots. Most of this activity is driven by AVNT ($79.1M vs. $0.5M for MIRROR).

The concentration is especially strong among short-lived triers. Of the 464 transient on-chain discovery bots, 84.3% execute at least one AVNT/MIRROR arbitrage, and 54.7% of them have a majority (>50%) of their activity in these tokens during the surge window. By contrast, persistent on-chain discovery bots are more diversified: median AVNT/MIRROR share is only 2.0%, and 35 of 40 have a majority of their activity outside these tokens.

The swarm does not persist. Figure 4B–C show that on-chain discovery's share of AVNT/MIRROR volume falls quickly post-surge, while the other two architectures absorb the bulk of the flow. The opportunity itself does not disappear – AVNT continues to generate arbitrage activity post-surge (Oct 5, 2025 to Feb 28, 2026), with 120,872 transactions across 308 bots, compared with 396,957 transactions across 619 bots during the surge weeks. What disappears is the surge-window on-chain discovery cohort: only 13 of the 504 September on-chain discovery bots remain active on post-surge AVNT flow, and all on-chain discovery bots account for 29.5% of post-surge AVNT volume, while off-chain discovery bots capture the largest share of post-surge AVNT volume (41.4%). The MIRROR pattern is similar, with only the 7 surge-window cohort bots surviving post-surge. Off-chain discovery bots capture the largest 60.9% post-surge MIRROR volume, while on-chain discovery bots fall to 24%.

We interpret this transition as evidence that on-chain discovery is temporarily valuable when new token routes are not yet routinized. This need not mean that existing on-chain discovery contracts discover new pools faster. Instead, early token-launch markets can generate dispersed retail flow, uncertain venue quality, and many short-lived route candidates, making probing attractive. Bots may pass venue parameters through calldata, use generic scanning logic, or deploy cloned contracts specialized to new route families [28]. As the routes become easier to monitor and parameterize off-chain, activity shifts toward on-chain evaluation and off-chain discovery, while most short-lived on-chain discovery entrants exit.

Spam burden. The entrant wave also generates a temporary spam shock: on-chain discovery bots produce 147M spam transactions during the surge window (cf. Figure 2C). Most of the level comes from short-lived entrants. The 464 transient bots generate 105.7M spam transactions, or 72% of on-chain discovery spam. The remaining 40 non-transient bots generate 41.3M spam transactions, but are far more intensive on a per-bot basis and account for more arbitrage activity per bot (403,464 arbitrages across 40 bots, versus 185,997 across 464 transient bots). Thus, the surge combines a mass-entry spam shock from transient bots with a smaller persistent core of high-intensity scanners.

Summary. The September surge provides an example of the model's opportunity-shock channel (cf. Proposition 8). On-chain discovery briefly revives around two newly launched token families in an otherwise stable post-Flashblocks regime, then recedes as launch-specific activity fades and the remaining flow is increasingly handled by other architectures. The associated spam surge comes mostly from short-lived triers that exit afterward. Since the episode overlaps the September 17 per-transaction gas-limit change [3], we do not interpret it as a clean event study of that policy. A later November rebound contrasts with September: volatility in already active token families mainly expands off-chain discovery, while a smaller on-chain discovery response still generates a visible spam increase (see full version [35]).

5.4 Minimum base fee escalation

Base introduced a minimum base fee on December 4, 2025 and increased it in five steps: 200K WEI on December 4, 500K WEI on December 18, 1M WEI on January 22, 2M WEI on February 2, and 5M WEI on February 19. In the model, the fee floor operates through

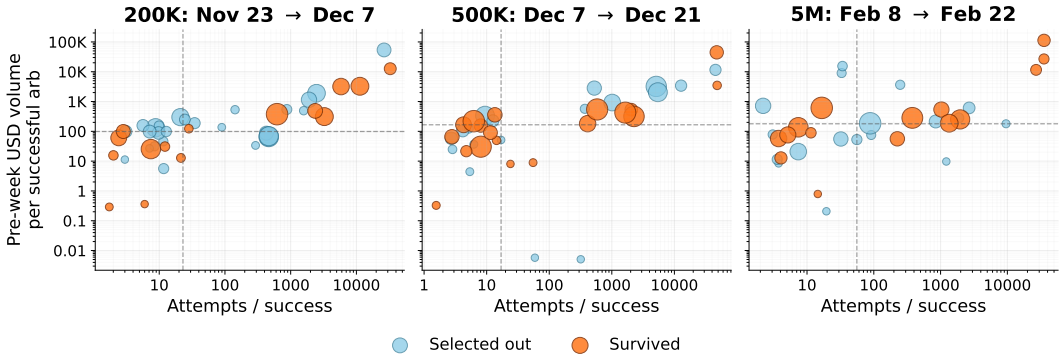


Figure 5 On-chain discovery bots active in the pre-step window for each fee increase, plotted by attempts per successful arbitrage and USD volume per successful arbitrage on log scales. Color indicates whether the bot remains active in the post-step window (orange) or is absent (blue). Marker size scales with total volume. The vertical and horizontal dashed lines mark cohort medians. Leftward indicates lower attempt intensity; upward indicates higher value per success, so the upper-left region corresponds to stronger value-cost margins.

the cost channel: architectures with greater fee exposure per captured opportunity become less viable as the fee floor rises. We expect the fee ramp to disproportionately affect on-chain discovery, where repeated probing and low success rates make the value-cost ratio especially fee-sensitive (cf. Proposition 6 and Remark 7). Because the fee changes are tightly spaced, we analyze the overall weekly population shift and use one-week local comparisons around each step to study selection on value-cost economics.

Population shift. The fee ramp coincides with a gradual reallocation toward off-chain discovery. Figure 3B shows on-chain discovery's active-bot share falling steadily across the ramp, while off-chain discovery rises, especially from late January 2026 onward. By the final sample week, on-chain discovery accounts for 8.4% of active bots and off-chain discovery for 71.1%, with on-chain evaluation comparatively stable. The shift is mainly selection and replacement rather than label switching: only three surviving bots switch labels during the fee escalation regime. The remaining on-chain discovery bots still capture non-trivial arbitrage flow, indicating that a smaller high-productivity core remains viable.

Selection on value-cost margin. Remark 7 predicts that fee floors raise the hurdle rate for on-chain discovery participation: bots whose opportunity access does not justify repeated attempts at higher cost should exit, regardless of their attempt intensity in isolation. Figure 5 plots pre-step on-chain discovery bots in attempts per success and USD volume per success at the 200K, 500K, and 5M fee steps, with marker size proportional to each bot's total pre-step arbitrage volume. Volume per success proxies for the value side of the value-cost ratio: large-size successful arbitrages imply more gross opportunity value available to cover failed-attempt costs. We discuss the 1M and 2M steps separately because they overlap a mid-ramp opportunity shock (see Appendix C.3 in the full version [35]).

The selection pattern is two-dimensional: neither attempt intensity nor volume per success alone predicts survival, with both surviving and selected-out bots present across most regions of each panel. Bots with high volume per success and low attempts per success tend to survive: they generate value without incurring high failed-attempt costs. Bots with low volume per success and high attempts per success are disproportionately selected out because

their value-cost ratio is unfavorable, and the fee floor pushes them below viability. Bots with both high attempts and high volume per success survive only when their successful opportunities are valuable enough to offset the cumulative cost of repeated attempts; in the 5M panel, this region is anchored by a small cluster of markers operating at \$10,000–100,000 per successful arbitrage. Thus, the fee floor does not select against attempt intensity alone; it selects against attempt intensity unsupported by sufficient opportunity value.

A cohort-level value-per-attempt comparison before and after each fee step, reported in Appendix C.3 in the full version [35], supports this interpretation. At the clean 200K and 5M steps, surviving bots' aggregate USD volume per attempt rises from 0.36 to 0.87 and from 0.46 to 1.31, respectively, consistent with pruning low-yield attempts or shifting toward higher-value opportunities. The 500K step moves in the opposite direction, but this window coincides with a broad market contraction in late December.

Unlike Flashblocks, the fee ramp does not appear to select for less scan-intensive on-chain discovery attempts at the trace-level. Therefore, broad scanning can still be profitable when opportunities are valuable enough, but low-yield probing becomes harder to sustain. The endpoint trace-level comparison is reported in Appendix C.3 in the full version [35].

Summary. These results support the model's cost-channel in Proposition 6. As fee floors raise per-attempt costs, on-chain discovery, the highest-exposure architecture, contracts from 22.0% to 8.4% of active bots. The selection is against repeated attempts unsupported by sufficient opportunity value, while broad scanning remains viable when opportunity value is high. Thus, higher fee floors shift the on-chain discovery viability frontier along the value-cost margin, distinct from Flashblocks' blockspace channel, which selects against heavy venue scanning in early ordering slices.

5.5 Protocol revenue and blockspace efficiency

The previous sections documented how bot architectures responded to Base's configuration changes. We now ask whether these responses improved chain-level outcomes, including protocol revenue composition and blockspace consumption.[9] These outcomes speak to both protocol revenue quality and the blockspace externality caused by spam.

Priority fees by architecture. Across the full sample, off-chain discovery successful arbitrages pay the highest median priority fee per transaction, at 8.66×10^{-7} ETH, compared with 4.72×10^{-7} ETH for on-chain evaluation and 5.02×10^{-7} ETH for on-chain discovery. This ordering is consistent with the economic role of off-chain route commitment. Because these bots can simulate routes and optimize input sizes before submitting, they are better positioned to identify higher-value opportunities and bid more aggressively for prioritized inclusion. Thus, the higher priority fees likely reflect both better opportunities and competition for more valuable arbitrages.

Figure 6 plots priority-fee distributions for successful arbitrages before and after Flashblocks by architecture. We note that the pre-Flashblocks window contains 5 weeks, while the post-Flashblocks window contains 33 weeks. The largest distributional change occurs for on-chain discovery. Before Flashblocks, successful on-chain discovery arbitrages exhibit a pronounced low-priority-fee mass, which we interpret as low-willingness-to-pay probabilistic search. After Flashblocks, this low-fee mass shrinks, and the on-chain discovery distribution

[9] We note that the base fees on Base accrue to the sequencer rather than being burned as on Ethereum.

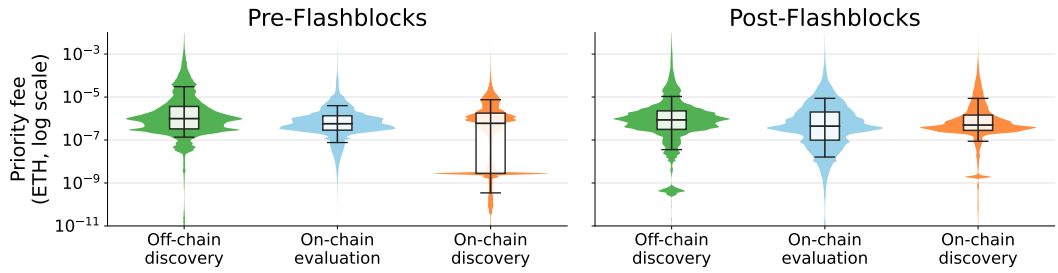


Figure 6 Priority-fee distributions for successful arbitrages by architecture before and after Flashblocks (3.04M arbitrages pre, 17.40M arbitrages post).

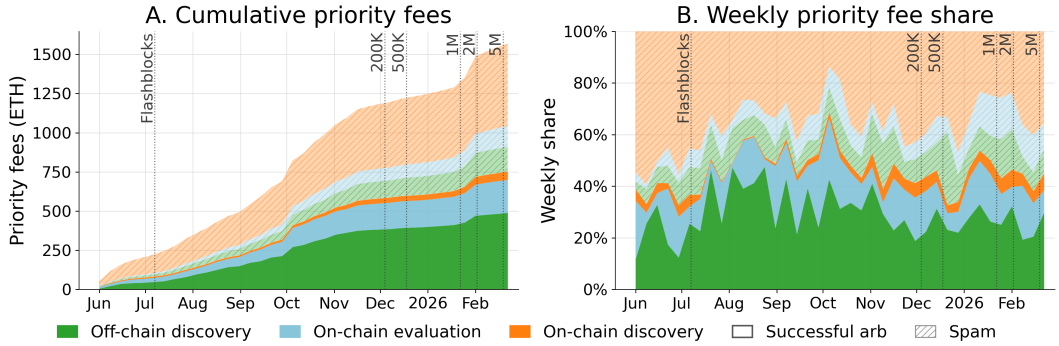


Figure 7 Priority-fee payments by bot architecture and transaction outcome. Panel A reports cumulative priority fees from arbitrage attempts. Panel B reports weekly priority-fee shares. Colors denote bot architecture; solid areas denote successful arbitrages and hatched areas denote spam.

moves closer to the other architectures. This pattern is consistent with Flashblocks selecting out the broadest low-revenue scanners and leaving a smaller population whose attempts are more targeted (cf. Table 3). The same qualitative pattern appears on the spam side (see Appendix C.4 in the full version [35]).

Taken together, the priority-fee evidence suggests that Flashblocks changes not only which bots remain active, but also how surviving on-chain discovery bots bid for inclusion. The low-priority-fee mass of on-chain discovery shrinks after Flashblocks, and the remaining on-chain discovery transactions pay priority fees closer to the other architectures. Thus, the post-Flashblocks on-chain discovery population is smaller but bids more aggressively.

A similar, though less clean, pattern appears across the minimum-base-fee ramp. Median priority fees for successful on-chain discovery arbitrages rise from 4.58×10^{-7} ETH in the pre-ramp endpoint week to 2.69×10^{-6} ETH in the post-ramp endpoint week, consistent with low-value (and therefore low-fee) attempts becoming less viable after repeated attempts become more costly (see Appendix C.4 in the full version [35]). Because we observe only one post-ramp week, we interpret it cautiously.

Protocol revenue composition. We next aggregate priority fees across arbitrage attempts to examine protocol revenue. Figure 7 decomposes cumulative and weekly priority fees by architecture and attempt outcome. The level series in Figure 7A is highly opportunity-driven: protocol revenue accumulates faster during high-volume periods, especially around the October and late-January episodes (cf. Figure 2B). For off-chain discovery and on-chain evaluation bots, cumulative priority fees come predominantly from successful arbitrages

Table 5 Gas usage by architecture.

Architecture	Arbitrage gas share	Spam gas share	Total gas share
Off-chain disc.	0.47%	0.99%	1.45%
On-chain eval.	0.41%	0.34%	0.75%
On-chain disc.	0.44%	19.59%	20.04%
All bots	1.31%	20.92%	22.23%

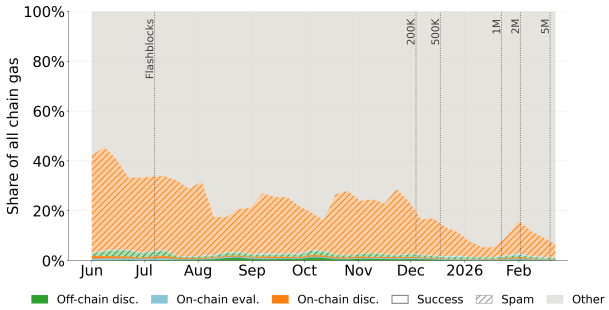


Figure 8 Arbitrage-related weekly gas usage share.

despite spam transactions being roughly three times more frequent (cf. Table 1). This reflects that their high-value successful captures pay large tips and account for a disproportionate share of cumulative fees. For on-chain discovery, by contrast, spam contributes a large share of cumulative priority fees because the architecture generates so many failed attempts.

Figure 7B shows a post-Flashblocks shift in arbitrage-related priority-fee revenue away from spam and toward successful arbitrages. Before Flashblocks, spam accounts for more than 60% of weekly priority fees; afterward, it falls to roughly 40–45%, even during the September entrant surge when spam transactions nearly double relative to pre-Flashblocks weeks (cf. Figure 2C). The successful-arbitrage share is correspondingly higher after Flashblocks, but remains opportunity-driven: it rises during high-volume periods, especially around the October episode, and declines as aggregate arbitrage volume falls (cf. Figure 2B). Thus, spam continues to contribute to nontrivial protocol revenue, but it becomes a less dominant component after the ordering granularity changes.

The fee-ramp period is harder to interpret because policy changes overlap with low-activity periods and a mid-ramp opportunity shock (cf. Figure 2B). Even so, the successful-arbitrage share rises following the 500K step, despite volume contraction, consistent with selection against low-value, high-attempt-intensity bots. The pattern becomes less clear around the 1M and 2M steps, when the opportunity shock temporarily revives on-chain discovery.

Blockspace efficiency. Despite protocol revenue contributions, failed attempts impose a burden on blockspace. Table 5 documents the gas usage by architecture as a share of total Base gas over the sample period. Although on-chain discovery bots only capture roughly half of successful arbitrages of the other two architectures (cf. Table 1), their arbitrages consume comparable blockspace because they are heavier (cf. Figure 1). Successful arbitrage transactions across all architectures consume only 1.31% of total gas on Base. The blockspace burden is instead driven almost entirely by spam, especially on-chain discovery spam, which accounts for roughly 20% of total Base gas.

Figure 8 decomposes weekly gas usage into arbitrage-related gas and other chain activity. Before Flashblocks, on-chain discovery spam alone consumes roughly one-third to nearly one-half weekly gas. Flashblocks reduces this burden, although not immediately. As shown in Section 5.2, surviving and entering on-chain discovery bots become more attempt-intensive in the four-week post window. The gas share declines further after that window to 15.2%, as a few extremely intensive bots disappear or downscale (see Appendix C.4 in the full version [35]).

Later episodes show the same distinction between architecture selection and opportunity shocks. During the September surge and the November rebound period, spam transaction counts rise substantially, but spam does not return to its early-sample blockspace share.

Through the fee ramp, spam gas share declines further, consistent with higher fees making low-yield probing harder to sustain. We interpret this cautiously because December and early January are periods of low market-wide activity (cf. Figure 2B), and the late-January opportunity shock temporarily raises spam gas even under a higher fee floor.

Taken together, Base's configuration changes improve blockspace efficiency in a narrow but important sense. Spam continues to occur, while its share of chain gas falls after Flashblocks and declines further through the fee-ramp period, aside from temporary opportunity shocks.

6 Concluding Discussion

Our paper shows that arbitrage competition on high-throughput blockchains involves economically distinct search architectures with different responses to protocol design. A reduced-form model studies how ordering granularity, fee floors, and opportunity-access shocks affect the relative viability of targeted and probabilistic search. Empirical evidence from Base supports these predictions, and traces how protocol changes reshape bot composition, execution behavior, priority-fee payments, and blockspace consumption.

The broader lesson is that protocol design affects not only how much arbitrage MEV is extracted, but also how it is pursued on-chain. Failed attempts are not merely isolated transaction-level inefficiencies; they arise from equilibrium search behavior under incomplete information, low attempt costs, and specific ordering environments. Opportunity shocks further show that protocol changes do not mechanically guarantee permanent reductions in failed attempts. The externality these attempts impose is blockspace congestion that displaces other users. Understanding how search architectures emerge across protocol designs is important for reducing externalities without suppressing useful economic activity such as arbitrage. We conclude by outlining several open questions for future work.

Does the search-architecture framework apply to other forms of MEV? Our analysis focuses on cyclic arbitrage, where the route discovery is visible in transaction traces. Other MEV forms may admit the same dichotomy. Liquidations can be targeted, when searchers wait for oracle updates against known positions, or probabilistic, when they repeatedly probe for liquidatable positions. Cross-domain arbitrage may similarly vary in how information is gathered and how transactions are submitted across domains. Future work could test whether these MEV forms exhibit analogous search architectures, how they respond to protocol changes, and what externalities they impose.

Can the model's predictions be tested on other blockchains? A natural next step is to compare chains with different execution environments, fees, and information availability. Ethereum likely favors targeted search because failed attempts are costly, and mempool visibility enables off-chain simulation. Arbitrum may also favor targeted search, but for a different reason: its fast block time reduces stale-state risk, consistent with evidence that Arbitrum bots appear less probe-like than bots on Base and Optimism [28]. Optimism is a closer Base comparison because both use the OP Stack [23] and share similar ordering and configuration choices, including Flashblocks. Outside the Ethereum ecosystem, Solana offers a useful contrast as a high-throughput chain with active probabilistic searching [6] and a different information environment, where transactions are forwarded directly to validators who may share pending-transaction information with searchers.

How would priority-access mechanisms affect architecture competition? Our framework can also be extended to ordering mechanisms that allocate priority explicitly. Such mechanisms change the margin of competition from simply reaching the earliest ordering unit to deciding whether priority access is worth paying for and how aggressively to use it. Arbitrum's Timeboost [21], which sells priority access, tests one such margin: priority access may favor targeted searchers that can forecast and monetize latency advantages, while leaving probabilistic search viable only for operators with enough opportunity flow to justify the cost. Existing evidence that express-lane access is concentrated and that roughly 22% of time-boosted transactions revert suggests that priority access alone does not eliminate spam [18]. Optimism's recent experiment with stake-based priority ordering [22] provides another useful setting for studying how stake requirements, first-come-first-served ordering, and priority-fee multipliers affect the relative viability of targeted and probabilistic search.

How should spam mitigation differ across architectures? Failed attempts from different architectures call for different mitigation tools. Revert protection can reduce failed route-committed attempts [9, 30], but it does not eliminate early-terminating targeted profitability checks or probabilistic probe spam, where transactions themselves are used to discover opportunities. Reducing probe-like spam requires improving information availability or reducing the incentive to use transactions as information-gathering devices, for example, through richer real-time information channels [15], protocol-run backrunning [24] and privacy-preserving backrunning auctions [10], and charging based on reserved execution gas [32]. On high-throughput chains, these mechanisms face a tight tradeoff between information freshness, auction efficiency, latency, and user protection.

References

1 AVANTIS. Avantis launch. URL: https://x.com/avantisfi/status/1965415885243556092.

2 Base. Base flashblocks configurations, 2025. URL: https://docs.base.org/base-chain/network-information/block-building#flashblocks.

3 Base. Configuration changelog, 2026. URL: https://docs.base.org/base-chain/network-information/configuration-changelog.

4 Base. Flashblocks Overview. https://docs.base.org/base-chain/flashblocks/overview, 2026. Base Documentation. Accessed: 2026-05-12.

5 Base. Tweet announcing base minimum fee increase. https://x.com/buildonbase/status/2024306165711327610, 2026.

6 Eugene Chen and @0xEdgar. MEV on Solana. https://www.umbraresearch.xyz/writings/mev-on-solana, 2023. Umbra Research. Accessed: 2026-05-10.

7 Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 910–927, 2020. doi:10.1109/SP40000.2020.00040.

8 Christof Ferreira Torres, Albin Mamuti, Ben Weintraub, Cristina Nita-Rotaru, and Shweta Shinde. Rolling in the shadows: Analyzing the extraction of mev across layer-2 rollups. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, pages 2591–2605, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3658644.3690259.

9 Flashbots. MEV Protection Overview. https://docs.flashbots.net/flashbots-protect/overview, 2025. Flashbots Documentation. Last updated: 2025-11-04. Accessed: 2026-05-13.

10 Flashbots. MEV-Share Introduction. https://docs.flashbots.net/flashbots-mev-share/introduction, 2025. Flashbots Documentation. Last updated: 2025-11-04. Accessed: 2026-05-13.

11 Flashbots and Robert Miller. Mev and the limits of scaling. URL: https://writings.flashbots.net/mev-and-the-limits-of-scaling.

12 Krzysztof Gogol, Johnnatan Messias, Deborah Miori, Claudio Tessone, and Benjamin Livshits. Cross-rollup mev: Non-atomic arbitrage across l2 blockchains, 2024. arXiv:2406.02172.

13 Krzysztof Gogol, Manvir Schneider, and Claudio Tessone. When priority fails: Revert-based mev on fast-finality rollups, 2025. arXiv:2506.01462.

14 Lioba Heimbach, Vabuk Pahari, and Eric Schertenleib. Non-Atomic Arbitrage in Decentralized Finance . In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 3866–3884, Los Alamitos, CA, USA, May 2024. IEEE Computer Society. doi:10.1109/SP54263.2024.00256.

15 Quintus Kilbourn. Dealing with spam caused by on-chain searching. https://collective.flashbots.net/t/dealing-with-spam-caused-by-on-chain-searching/3381, 2024. The Flashbots Collective, Research forum post. Accessed: 2026-05-13.

16 Bruno Mazorra, Christoph Schlegel, and Akaki Mamageishvili. Timing games: Probabilistic backrunning and spam, 2026. doi:10.48550/arXiv.2602.22032.

17 Robert McLaughlin, Christopher Kruegel, and Giovanni Vigna. A large scale study of the ethereum arbitrage ecosystem. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 3295–3312, Anaheim, CA, August 2023. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/mclaughlin.

18 Johnnatan Messias and Christof Ferreira Torres. The express lane to spam and centralization: An empirical analysis of arbitrum's timeboost, 2025. doi:10.48550/arXiv.2509.22143.

19 MIRROR. Mirror tokenomics & distribution. URL: https://x.com/blackmirror_xp/status/1964605903824703813.

20 Alexandre Obadia, Alejo Salles, Lakshman Sankar, Tarun Chitra, Vaibhav Chellani, and Philip Daian. Unity is strength: A formalization of cross-domain maximal extractable value, 2021. arXiv:2112.01472.

21 Offchain Labs. A Gentle Introduction: Timeboost. https://docs.arbitrum.io/how-arbitrum-works/timeboost/gentle-introduction, 2026. Last updated: 2026-02-17; Accessed: 2026-05-08.

22 Optimism. Stake-based priority ordering on OP Mainnet. https://docs.optimism.io/notices/stake-based-priority-ordering, 2026. Accessed: 2026-05-08.

23 Optimism. The OP Stack. https://docs.optimism.io/op-stack/introduction/op-stack, 2026. Optimism Documentation. Accessed: 2026-05-21.

24 Osmosis. Protorev. https://docs.osmosis.zone/overview/features/protorev/, 2026. Osmosis Docs. Accessed: 2026-05-13.

25 Burak Öz, Christof Ferreira Torres, Christoph Schlegel, Bruno Mazorra, Jonas Gebele, Filip Rezabek, and Florian Matthes. Cross-chain arbitrage: The next frontier of mev in decentralized finance. *Proc. ACM Meas. Anal. Comput. Syst.*, 9(3), December 2025. doi:10.1145/3771566.

26 Kaihua Qin, Liyi Zhou, Pablo Gamito, Philipp Jovanovic, and Arthur Gervais. An empirical study of defi liquidations: incentives, risks, and instabilities. In *Proceedings of the 21st ACM Internet Measurement Conference*, IMC '21, pages 336–350, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3487552.3487811.

27 Kaihua Qin, Liyi Zhou, and Arthur Gervais. Quantifying blockchain extractable value: How dark is the forest? In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 198–214, 2022. doi:10.1109/SP46214.2022.9833734.

28 Ozan Solmaz, Lioba Heimbach, Yann Vonlanthen, and Roger Wattenhofer. Optimistic MEV in Ethereum Layer 2s: Why Blockspace Is Always in Demand. In Zeta Avarikioti and Nicolas Christin, editors, *7th Conference on Advances in Financial Technologies (AFT 2025)*, volume 354 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:24, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.AFT.2025.28.

29 Christof Ferreira Torres, Ramiro Camino, and Radu State. Frontrunner jones and the raiders of the dark forest: An empirical study of frontrunning on the ethereum blockchain. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1343–1359. USENIX Association, August 2021. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/torres.

30 Uniswap Labs. Live on Unichain: Fair Transaction Ordering and MEV Protection. https://blog.uniswap.org/rollup-boost-is-live-on-unichain, 2025. Accessed: 2026-05-08.

31 Qin Wang, Ruiqiang Li, Guangsheng Yu, Vincent Gramoli, and Shiping Chen. Mev in binance builder, 2026. doi:10.48550/arXiv.2602.15395.

32 Wenhao Wang, Aditya Saraf, Lioba Heimbach, Kushal Babel, and Fan Zhang. Blockspace under pressure: An analysis of spam mev on high-throughput blockchains, 2026. doi:10.48550/arXiv.2604.00234.

33 Ye Wang, Yan Chen, Haotian Wu, Liyi Zhou, Shuiguang Deng, and Roger Wattenhofer. Cyclic arbitrage in decentralized exchanges. In *Companion Proceedings of the Web Conference 2022*, WWW '22, pages 12–19, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3487553.3524201.

34 Fei Wu, Danning Sui, Thomas Thiery, and Mallesh Pai. Measuring CEX-DEX Extracted Value and Searcher Profitability: The Darkest of the MEV Dark Forest. In Zeta Avarikioti and Nicolas Christin, editors, *7th Conference on Advances in Financial Technologies (AFT 2025)*, volume 354 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:23, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.AFT.2025.26.

35 Fei Wu and Burak Öz. To wait or to probe: Arbitrage competition on high-throughput blockchains, 2026. doi:10.48550/arXiv.2606.00720.

36 Liyi Zhou, Kaihua Qin, Christof Ferreira Torres, Duc V Le, and Arthur Gervais. High-frequency trading on decentralized on-chain exchanges. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 428–445, 2021. doi:10.1109/SP40001.2021.00027.

37 Brian Z. Zhu, Xin Wan, Ciamac C. Moallemi, Dan Robinson, and Brad Bachu. Quantifying the value of revert protection. In Christina Garman and Pedro Moreno-Sanchez, editors, *Financial Cryptography and Data Security*, pages 72–88, Cham, 2026. Springer Nature Switzerland.

38 Burak Öz, Jonas Gebele, Parshant Singh, Filip Rezabek, and Florian Matthes. Playing the mev game on a first-come-first-served blockchain. In *2024 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 220–224, 2024. doi:10.1109/ICBC59979.2024.10634397.

The Price of Decentralization in Block Building

Burak Öz[*]
Flashbots, Berlin, Germany

Fei Wu[*†]
King's College London, London, UK

Luis Correia
Flashbots, London, UK

Sen Yang
Yale University, New Haven, CT, USA
IC3, New York, NY, USA

Bruno Mazorra
Flashbots, Barcelona, Spain

Stefanos Leonardos
King's College London, London, UK

Abstract

Decentralized block building mechanisms replace the monopoly of a single proposer with multiple builders. However, their censorship-resistance and fair-access benefits depend not only on the number of builders, but also on where builders are geographically positioned to provide transaction coverage. We study this tension between builder location choice, user transaction coverage, and utility concentration by modeling decentralized block building as a stochastic coverage game. Builders choose regions, information sources emit transactions over a block construction round, and latency determines whether a transaction is received before the deadline.

We show that the game is an exact potential game and admits a pure Nash equilibrium. We prove an asymptotically tight factor-2 bound on the Price of Anarchy, which we interpret as the *price of decentralization* from uncoordinated builder placement. We also study builder utility concentration, showing that the lowest-utility builder earns at least half of the highest-utility builder's payoff, and the utility-share HHI is at most 12.5% above the egalitarian benchmark. We complement the theory with simulations under richer latency and source environments and show that slot times, builder participation, and reward-sharing rules are important protocol design choices that can shape the price of decentralization.

2012 ACM Subject Classification Theory of computation → Algorithmic game theory; Computing methodologies → Modeling and simulation

Keywords and phrases decentralization, block building, potential game, price of anarchy

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.25

Related Version *Full Version*: `https://arxiv.org/abs/2606.01874` [37]

Supplementary Material *Software (Source Code)*: `https://github.com/boz1/db-sims` [1]
archived at `swh:1:dir:afa10f961c32d9645eb1b3fad7a89764a79699aa`

[*]These authors contributed equally to this work.
[†]Fei Wu performed work in part during an internship at Flashbots in Spring 2026.

© Burak Öz, Fei Wu, Luis Correia, Sen Yang, Bruno Mazorra, and Stefanos Leonardos;
licensed under Creative Commons License CC-BY 4.0

1 Introduction

Many blockchain protocols assign the block proposal right to a single leader in each slot. This temporary monopoly gives the proposer control over transaction inclusion and ordering, creating opportunities for Maximal Extractable Value (MEV) [8]. Moreover, proposers located in particular jurisdictions may face censorship pressures, such as those induced by OFAC sanctions in the U.S. [33], and geographic concentration can create potential liveness risks in the presence of regional outages. Recent multiple-proposer or decentralized block-building proposals seek to reduce this monopoly and improve censorship resistance by allowing multiple builders to construct candidate blocks or sub-blocks [16, 22].[1] If a user transaction can be included by multiple builders, who typically share the fees paid by the transaction, this increases the cost of censorship: excluding a transaction may require bribing multiple entities rather than a single proposer.

However, multiple builders do not automatically imply geographically broad transaction coverage. In permissionless blockchains, the consensus protocol typically does not mandate where builders should operate. Because latency affects whether transactions reach builders before the block construction deadline, self-interested builders may prefer regions that provide access to the same high-value transaction sources. Recent work shows that Ethereum's block-building protocol design is not geographically neutral and can create co-locating incentives to minimize latency to payoff-critical parties [36]. If builders concentrate in a small number of regions, the benefits of having multiple builders can diminish, as users located farther away may face weaker inclusion guarantees due to latency effects. In auction-like settings, such latency differences can give colocated parties an informational advantage [24].

This raises a critical design question: if decentralized block building mechanisms rely on multiple self-interested builders who choose locations to maximize individual utility, will their location choices actually improve access for geographically distributed users and transaction sources? How inefficient and concentrated can the resulting placements be? These questions matter because the censorship-resistance and fair-access benefits of multi-builder designs depend not only on the number of builders, but also on whether their locations are distributed enough to provide timely coverage to users and transaction sources across regions over the existing Internet infrastructure with latency constraints.

In this paper, we model decentralized block building as a stochastic coverage game (cf. Section 2). Builders choose geographic regions, while information sources emit transactions over the course of a block-construction round. A builder's region determines the probability that a transaction emitted by a given source reaches the builder before the block deadline. Since multiple builders may receive and include the same transaction, decentralized placement can create redundant coverage: several builders may cover the same high-value sources, while other sources remain weakly covered.

We study the welfare loss induced by this decentralized builder region game (cf. Section 3). Relative to a social planner who coordinates builder locations to maximize expected transaction coverage, selfish builders may over-concentrate in regions that provide access to the same valuable information sources. We measure this efficiency loss through the Price of Anarchy (PoA), which we interpret in this setting as the *price of decentralization*, and derive an asymptotically tight worst-case bound on it using the smoothness framework [31]. The key observation is that when a transaction is already covered by several builders, an additional

[1] In the paper, we use *builder* as the generic term for an entity that constructs and proposes a candidate (sub-)block.

builder receives only a reduced share of its value under the equal-split sharing rule, while not improving welfare. This allows us to relate unilateral deviations to aggregate welfare. We also prove the existence of a pure Nash equilibrium by constructing an exact potential function, and establish bounds on utility concentration among builders at equilibrium. In our model, welfare captures the total transaction value covered by at least one builder, while utility concentration captures how the economic benefits of block building are distributed.

The theoretical results characterize worst-case properties: whether stable placements exist, how inefficient they can be, and how concentrated builder utilities can become. To complement this analysis, we run simulations in richer latency and source environments. These simulations show when welfare losses arise in concrete instances, how they are driven by under-coverage of peripheral sources, and why geographic concentration and builder utility concentration need not coincide (cf. Sections 4 and 5). The results highlight how protocol designs can shape welfare in decentralized block building (cf. Section 6). We summarize our contributions as follows.

- We model decentralized block building as a latency-induced stochastic coverage game in which builders choose regions and transaction sources generate value over time. We analyze the equilibrium and efficiency properties of the game under the equal-split sharing rule. We show that the game admits a pure Nash equilibrium through an exact potential function and prove a tight factor-2 PoA bound.
- We study utility concentration among builders in equilibrium. We show that the lowest-utility builder earns at least half of the highest-utility builder's payoff, and that the resulting utility-share HHI is at most 12.5% above the egalitarian benchmark. Both bounds are tight.
- We use simulations [1] to study the game under richer latency and source environments. We find that welfare losses are largest in intermediate regimes where peripheral sources are reachable, but selfish incentives still favor high-value source regions. We also show that geographic concentration and utility concentration need not align: planner allocations may improve coverage by placing builders in lower-payoff peripheral regions, while equilibrium outcomes can be more geographically centralized but more utility-balanced.

1.1 Related work

Geographic (de)centralization. Prior measurement studies show that Ethereum nodes are geographically concentrated, with deployments repeatedly skewed toward a small number of countries, particularly in Europe and North America [6, 11, 19–21]. These studies establish geographic concentration as an empirical phenomenon, while recent work studies the incentives and mechanisms that shape geographic decentralization. Yang et al. [36] show that Ethereum's block building regimes create location-dependent validator rewards, allowing latency advantages and asymmetric access to information sources to reinforce concentration. Roeschlin et al. [30] study how to enforce geography-aware incentives when nodes may misreport their locations, using RTT-based localization to support truthful reporting. Our work is complementary to both lines. We study equilibrium outcomes in decentralized block building when builders strategically choose geographic regions, and analyze how latency-sensitive placement affects welfare, geographic concentration, and builder utility concentration.

Decentralized block building. Decentralized block building aims to reduce centralized control over transaction inclusion and ordering, especially under MEV-driven incentives, and to improve censorship resistance. Industry discussions have advocated distributed, parallel, and global block building architectures to counter latency moats and censorship risks [13, 23].

Recent work proposes Multiple Concurrent Proposers (MCP) designs to reduce the serial monopoly of a single proposer over inclusion and ordering [15, 16, 22]. Complementary work studies transaction assignment in such protocols, showing that stronger censorship resistance often requires more transaction duplication and hence lower throughput [10]. Our work complements these works through placement equilibria: we study how builders' strategic geographic choices affect welfare, geographic concentration, and builder utility concentration.

Coverage games and equilibrium inefficiency. Vetta introduced valid utility games, where agents' utilities are tied to a monotone submodular social objective and Nash equilibria admit constant welfare guarantees [32]. Gairing studied covering games, where players select subsets of weighted elements and covered value is allocated through utility-sharing rules [14]. Our equal-split rule corresponds to symmetric equal sharing, but coverage is stochastic and induced by heterogeneous source-to-builder reception probabilities.

Prior work studies the distributional properties of equilibria through notions such as utility uniformity [12] and fairness ratio for costs [3]. Our utility dispersion and HHI results complement this literature by studying reward concentration in a stochastic coverage game.

Beyond worst-case PoA guarantees, related work examines inefficiency in particular parameter regimes and among outcomes reached in practice or through specified response dynamics [4, 7, 26]. Our simulations take a similar instance-level perspective by evaluating equilibria selected by asynchronous better-response dynamics across sampled environments and initial conditions.

2 Model

2.1 Sources, regions, and probabilistic reception

We study a decentralized block-building network in which, in every round of a blockchain protocol, a new block containing transactions is appended. Each round has a duration $\Delta > 0$. Denote the set of builders by $\mathcal{B} := \{1, \dots, K\}$ and the set of geographical regions in the network by $\mathcal{R} := \{1, \dots, R\}$.

In each round, every builder constructs a sub-block from the transactions it receives before the round deadline. We assume that all received transactions are included and impose no block-capacity constraint. We discuss the implications of block-capacity constraints in Section 6.

Let $\mathcal{I}$ be a finite set of information sources. Each source $I \in \mathcal{I}$ is associated with a fixed region $r(I) \in \mathcal{R}$. For each source I, let $\mathcal{J}_I$ denote the random set of transactions emitted by source I during one round. Each transaction $j \in \mathcal{J}_I$ has an emission time $t_j \in [0, \Delta]$ and a positive value V_j.

We assume that source I has finite expected emitted value in a round. Specifically, there exist a scalar $\Lambda_I > 0$ and a density $\rho_I : [0, \Delta] \to \mathbb{R}_+$ with $\int_0^\Delta \rho_I(t)\, dt = 1$ such that, for every bounded function $h : [0, \Delta] \to \mathbb{R}$,

$$\mathbb{E}\left[\sum_{j \in \mathcal{J}_I} V_j\, h(t_j)\right] = \Lambda_I \int_0^\Delta h(t)\rho_I(t)\, dt.$$

Thus, Λ_I is the expected total value emitted by source I in one round, and ρ_I describes how this expected value is distributed over time within this round. This formulation lets us work directly with expected value flow rather than specifying the full arrival process of transactions and values.

For each ordered region pair $(i, k) \in \mathcal{R} \times \mathcal{R}$, let $D_{k\to i}$ denote the propagation latency from a source in region k to a builder located in region i, and let $F_{k\to i}$ be its cumulative distribution function. We allow the latency distribution to depend on the ordered region pair, so different source-builder pairs may exhibit different propagation characteristics. This captures persistent geographical differences in network connectivity while allowing for random latency fluctuations.

A strategy for builder b consists of selecting a region $s_b \in \mathcal{R}$, and a strategy profile is written as $\mathbf{s} := (s_1, \ldots, s_K) \in \mathcal{R}^K$. Region selection is modeled as a one-shot simultaneous-move game: builders choose regions based on the known source environment, but not on the contemporaneous choices of the other builders.

Consider a transaction emitted at time $t \in [0, \Delta]$ by a source located in region k. If builder b is located in region $i = s_b$, then the transaction reaches builder b before the block deadline with probability

$$q_k(i, t) := \Pr(D_{k\to i} \le \Delta - t) = F_{k\to i}(\Delta - t).$$

Thus, for a source I and a builder b, the corresponding reception probability is obtained by setting $k = r(I)$ and $i = s_b$, giving $q_{r(I)}(s_b, t)$. Since reception probabilities depend on builders only through their chosen regions, all builders located in the same region have identical reception probabilities.

Let $\mathcal{T}_b$ denote the random set of transactions received by builder b in a round, and write $\mathcal{T}_b^I := \mathcal{T}_b \cap \mathcal{J}_I$ for the transactions from source I received by b. Conditional on the realized transactions and their emission times, receptions are independent across builders. Thus, two builders located in the same region have the same reception probability for a given transaction, but need not have the same realized reception outcome.

2.2 Value sharing rule and welfare

After a round, the sub-blocks of all builders are aggregated into a final block. When multiple builders include the same transaction, its value is shared among them. For each transaction j, let

$$n_j(\mathbf{s}) := |\{b \in \mathcal{B} : j \in \mathcal{T}_b\}|$$

denote the number of builders that received and included transaction j under profile $\mathbf{s}$. When the profile is clear from context, we write n_j.

We assume an *equal-split sharing rule*: if $n_j \ge 1$, then each builder that included transaction j receives the fraction $1/n_j$ of its value. Builders that did not include j receive zero from it. Equivalently, the reward assigned to builder b from transaction j is

$$V_j \cdot \frac{\mathbf{1}\{j \in \mathcal{T}_b\}}{n_j},$$

with the convention that this term is zero when $n_j = 0$.

We use equal splitting as a simple symmetric and budget-balanced sharing rule: it depends only on the number of builders that include the transaction, treats all including builders equally, and distributes exactly V_j when the transaction is covered by at least one builder. Thus, the total reward distributed for transaction j is $V_j \mathbf{1}\{n_j \ge 1\}$. This rule is consistent with Shapley-style distribution designs for multi-proposer transaction fee mechanisms, where the reward from a transaction is shared among the proposers that include it [15]. We abstract from the bidding layer and treat V_j as the transaction payment to be shared; alternative reward-sharing rules are discussed in Section 6.

The realized reward of builder b under profile $\mathbf{s}$ is

$$\Pi_b(\mathbf{s}) = \sum_{I \in \mathcal{I}} \sum_{j \in \mathcal{J}_I} V_j \cdot \frac{\mathbf{1}\{j \in \mathcal{T}_b\}}{n_j},$$

again with the convention that the summand is zero when $n_j = 0$.

Since transaction generation, including values and emission times, and reception outcomes are random, $\Pi_b(\mathbf{s})$ is a random variable. The payoff-relevant quantity for builder b is its expected reward

$$u_b(\mathbf{s}) := \mathbb{E}[\Pi_b(\mathbf{s})],$$

where the expectation is taken over transaction generation and reception outcomes. Given the other builders' region choices, each builder chooses s_b to maximize its expected reward.

We define the *social welfare* of profile $\mathbf{s}$ as the expected total value of transactions received by at least one builder:

$$W(\mathbf{s}) := \mathbb{E}\left[\sum_{I \in \mathcal{I}} \sum_{j \in \mathcal{J}_I} V_j \, \mathbf{1}\{n_j \geq 1\}\right].$$

Thus, $W(\mathbf{s})$ measures the total user value realized by transactions that are successfully covered under profile $\mathbf{s}$. Following the standard welfare view in prior work [2], we treat transfers between users and builders as redistributing value rather than creating it. Therefore, in our stylized model, welfare equals the total value of included transactions.

The equal-split sharing rule implies a useful accounting identity: aggregate builder utility equals welfare. We state this formally in Lemma 5.

2.3 Equilibrium and social optimum

The preceding definitions induce a finite game in which builders are the players, regions are the actions, and expected rewards u_b are the payoffs. We refer to this game as the *builder region game*.

A pure strategy profile $\mathbf{s}$ is a Nash equilibrium of the builder region game if no builder can improve its expected reward through a unilateral deviation. That is, for every $b \in \mathcal{B}$ and every $s'_b \in \mathcal{R}$,

$$u_b(\mathbf{s}) \geq u_b(s'_b, \mathbf{s}_{-b}),$$

where $\mathbf{s}_{-b}$ denotes the strategies of all builders other than b.

A social planner chooses the entire profile $\mathbf{s}$ jointly to maximize welfare. We denote a welfare-maximizing profile by

$$\mathbf{s}^* \in \arg\max_{\mathbf{s} \in \mathcal{R}^K} W(\mathbf{s}),$$

and call $W(\mathbf{s}^*)$ the optimal welfare.

3 Equilibrium, Efficiency, and Utility Concentration

In this section, we analyze the equilibrium, efficiency, and utility-concentration properties of the builder region game. We first show that the equal-split sharing rule induces an exact potential function, which implies the existence of a pure Nash equilibrium. We then bound the welfare loss in equilibrium using the PoA. Finally, we study concentration in builder utilities through the Herfindahl–Hirschman Index (HHI). We provide the proofs in the appendix of the full version [37].

3.1 Pure Nash equilibrium via an exact potential

We first show that the builder region game admits a pure Nash equilibrium by constructing an exact potential function. Under equal sharing, welfare itself is not a potential function: if a transaction is already covered by other builders, an additional builder can still obtain positive utility by also receiving it, while welfare does not increase. The key observation is instead that the marginal payoff from joining a transaction already received by m other builders is exactly $V_j/(m+1)$, which corresponds to the increment of a harmonic potential.

▶ **Proposition 1** (Exact potential under equal sharing). *Consider the builder region game under the equal-split sharing rule. For $m \geq 1$, let*

$$H_m := \sum_{\ell=1}^{m} \frac{1}{\ell}, \qquad H_0 := 0$$

denote the harmonic numbers. For a realized transaction j, let

$$n_j(\mathbf{s}) := |\{b \in \mathcal{B} : j \in \mathcal{T}_b\}|$$

denote the number of builders that receive j under profile $\mathbf{s}$. Define

$$\Phi(\mathbf{s}) := \mathbb{E}\left[\sum_{I \in \mathcal{I}} \sum_{j \in \mathcal{J}_I} V_j \, H_{n_j(\mathbf{s})}\right].$$

Then Φ is an exact potential for the expected-utility game. That is, for every builder $b \in \mathcal{B}$, every profile $\mathbf{s}$, and every deviation $s'_b \in \mathcal{R}$,

$$\Phi(s'_b, \mathbf{s}_{-b}) - \Phi(\mathbf{s}) = u_b(s'_b, \mathbf{s}_{-b}) - u_b(\mathbf{s}).$$

Hence, under equal sharing, the builder region game is a finite exact potential game.

▶ **Corollary 2** (Pure equilibria and better-response convergence). *Under the equal-split sharing rule, the builder region game admits a pure Nash equilibrium. Moreover, every asynchronous sequence of strict better responses terminates in finite time at a pure Nash equilibrium.*

Proof. By Proposition 1, the game is a finite exact potential game. Hence, any maximizer of Φ over the finite pure profile space $\mathcal{R}^K$ is a pure Nash equilibrium [25].

Now consider an asynchronous sequence of strictly better responses. At each step, a single builder changes its region in a way that strictly increases its expected utility. By exact potentiality, this strictly increases the potential Φ. Since the pure profile space $\mathcal{R}^K$ is finite, such strict improvements cannot continue indefinitely. The process must therefore terminate at a profile with no profitable unilateral deviation, i.e., a pure Nash equilibrium. ◀

In the deterministic-reception special case, the game admits a direct congestion-game representation, where transaction sources are resources and a source of value Λ_I gives payoff Λ_I/m to each of its m covering builders, who dilute each others rewards. With probabilistic reception, expected payoff depends on the full vector of heterogeneous probabilities, so this compact source-level representation is less direct. Abstractly, every finite exact potential game can be mapped to a congestion game [25].

3.2 Price of Anarchy in decentralized block building

In the builder region game, builders may over-concentrate in regions that provide access to valuable information sources. This can lead multiple builders to include the same transactions, creating redundant coverage, while other sources remain weakly covered. The resulting mismatch between private incentives and aggregate coverage can reduce welfare. We use PoA to measure the welfare loss caused by selfish region selection relative to the welfare-maximizing profile chosen by a social planner.

Let $\mathcal{E}$ denote the set of pure Nash equilibria of the builder region game, and let $\mathbf{s}^*$ be a welfare-maximizing profile. We define PoA as

$$\text{PoA} := \frac{W(\mathbf{s}^*)}{\min_{\mathbf{s}\in\mathcal{E}} W(\mathbf{s})}.$$

The bound we derive follows the standard PoA argument for valid-utility games [32], a class that includes coverage and location-game variants, expressed through the smoothness framework [31]. The smoothness framework bounds inefficiency by comparing the total payoff players would obtain by unilaterally deviating from a current profile toward a benchmark profile with the welfare of the two profiles. Our contribution is to verify that the builder region game satisfies the required structure after taking expectations over transaction value flows and latency-induced probabilistic reception. Specifically, we show that welfare admits a source-level coverage representation, that equal-split sharing makes aggregate builder utility equal welfare, and that deviation payoffs can be lower-bounded by exclusive coverage. These properties allow the standard smoothness proof to carry over to our stochastic block-building setting.

A payoff-maximization game is (λ, μ)-smooth if, for every pair of profiles $\mathbf{s}$ and $\mathbf{s}^*$,

$$\sum_{b\in\mathcal{B}} u_b(s_b^*, \mathbf{s}_{-b}) \geq \lambda W(\mathbf{s}^*) - \mu W(\mathbf{s}).$$

If a game is (λ, μ)-smooth and aggregate utility equals welfare, then every pure Nash equilibrium has welfare at least a $\lambda/(1+\mu)$ fraction of the optimal welfare. Equivalently,

$$\text{PoA} \leq \frac{1+\mu}{\lambda}.$$

▶ **Definition 3** (Coverage). *A transaction from source I emitted at time t is* covered *under profile $\mathbf{s}$ if at least one builder receives it before the deadline. The corresponding coverage probability is*

$$f_I(t; \mathbf{s}) := 1 - \prod_{b\in\mathcal{B}} \big(1 - q_{r(I)}(s_b, t)\big).$$

The expected coverage of source I under profile $\mathbf{s}$, averaged with respect to the source's emission-time profile ρ_I, is

$$\bar{f}_I(\mathbf{s}) := \int_0^\Delta \rho_I(t) f_I(t; \mathbf{s})\, dt.$$

▶ **Lemma 4** (Coverage representation of welfare). *For every profile $\mathbf{s}$, total expected welfare satisfies*

$$W(\mathbf{s}) = \sum_{I\in\mathcal{I}} \Lambda_I \bar{f}_I(\mathbf{s}).$$

The proof applies the expected value-flow assumption with $h(t) = f_I(t; \mathbf{s})$ and then sums over sources.

Interpretation. The lemma shows that welfare depends only on whether each transaction is covered by at least one builder. Additional builders receiving the same transaction do not create additional social value. Thus, welfare losses arise from inefficient region allocation: if too many builders cover the same high-value sources, other sources may remain weakly covered relative to the social optimum.

▶ **Lemma 5** (Aggregate utility identity). *Under the equal-split sharing rule, aggregate builder utility equals welfare. That is, for every profile* $\mathbf{s}$*,*

$$\sum_{b\in\mathcal{B}} u_b(\mathbf{s}) = W(\mathbf{s}).$$

This follows because a covered transaction distributes exactly its full value across the builders that include it, while an uncovered transaction distributes zero.

▶ **Lemma 6** (Smoothness inequality). *The builder region game is* $(1,1)$*-smooth. That is, for all profiles* $\mathbf{s}$ *and* $\mathbf{s}^*$*,*

$$\sum_{b\in\mathcal{B}} u_b(s_b^*, \mathbf{s}_{-b}) \geq W(\mathbf{s}^*) - W(\mathbf{s}).$$

The proof lower-bounds each builder's deviation payoff by the value it would cover exclusively under the deviation. Summing these exclusive-coverage terms across builders and applying a union-bound argument shows that they cover the welfare gap $W(\mathbf{s}^*) - W(\mathbf{s})$.

▶ **Theorem 7** (Price of Anarchy bound). *The PoA of the builder region game is at most* 2*. Equivalently, for every pure Nash equilibrium* $\mathbf{s} \in \mathcal{E}$ *and every welfare-maximizing profile* $\mathbf{s}^*$*,*

$$W(\mathbf{s}) \geq \frac{1}{2} W(\mathbf{s}^*).$$

▶ Remark 8 (Broader equilibrium notions). The factor-2 guarantee follows from the standard smoothness argument for valid utility games [32]. The block-building-specific content is the verification that a latency-induced stochastic region-selection game preserves this structure in expectation. Although Theorem 7 is stated for pure Nash equilibria, the same smoothness argument implies the factor-2 welfare guarantee for mixed Nash, correlated, and coarse correlated equilibria, with welfare evaluated in expectation [31].

We next construct an example to show that the factor-2 bound is asymptotically tight. The worst case in this game arises when all builders crowd into a region with a single highly valuable source that is already fully covered. A unilateral move to an uncovered secondary source's region would improve social coverage but reduce builder's utility. In contrast, a social planner would keep only one builder at the dominant source and allocate the remaining builders to uncovered source regions.

▶ **Proposition 9** (Tightness of the factor-2 bound). *For every* $K \geq 2$ *and every* $\varepsilon \in (0,1)$*, there exists an instance of the builder region game with* K *builders, such that some pure Nash equilibrium* $\mathbf{s}$ *satisfies*

$$\frac{W(\mathbf{s}^*)}{W(\mathbf{s})} = 2 - \frac{1}{K} - \frac{K-1}{K}\varepsilon.$$

In particular, both the PoA and PoS can be arbitrarily close to 2*.*

▶ Remark 10 (Price of Stability). The same construction also implies that the best and worst equilibria coincide, and the game admits a unique pure Nash equilibrium. Therefore, the factor-2 bound is asymptotically tight for the Price of Stability.

3.3 Builder utility dispersion and concentration

We next study how dispersed builder utilities can be at equilibrium. Our utility-dispersion measure coincides with the notion of *utility uniformity* studied in [12]. We then use this dispersion bound to quantify utility concentration among builders under decentralized block building with the equal-split sharing rule.

▶ **Lemma 11** (Equilibrium utility dispersion)**.** *Under the equal-split sharing rule, let* $\mathbf{s}$ *be a pure Nash equilibrium. Define*

$$u_{\min}(\mathbf{s}) := \min_{b\in\mathcal{B}} u_b(\mathbf{s}), \qquad u_{\max}(\mathbf{s}) := \max_{b\in\mathcal{B}} u_b(\mathbf{s}).$$

Then

$$u_{\min}(\mathbf{s}) \geq \frac{1}{2} u_{\max}(\mathbf{s}).$$

▶ **Proposition 12** (Tightness of the utility dispersion bound)**.** *The factor* $1/2$ *in Lemma 11 is tight: there exists an instance and a pure Nash equilibrium* $\mathbf{s}$ *such that*

$$u_{\min}(\mathbf{s}) = \frac{1}{2} u_{\max}(\mathbf{s}).$$

▶ **Definition 13** (Utility shares and concentration)**.** *For any profile* $\mathbf{s}$ *with* $\sum_{b\in\mathcal{B}} u_b(\mathbf{s}) > 0$*, define the utility share of builder* b *by*

$$x_b(\mathbf{s}) := \frac{u_b(\mathbf{s})}{\sum_{c\in\mathcal{B}} u_c(\mathbf{s})}.$$

The Herfindahl–Hirschman Index (HHI) of the utility profile is

$$\mathrm{HHI}(\mathbf{s}) := \sum_{b\in\mathcal{B}} x_b(\mathbf{s})^2.$$

Its minimum possible value over K *builders is attained at equal utilities:*

$$\mathrm{HHI}^{\mathrm{egal}} = \frac{1}{K}.$$

▶ **Proposition 14** (Equilibrium bound on utility concentration)**.** *Assume the equal-split sharing rule, and let* $\mathbf{s}$ *be a pure Nash equilibrium with* $\sum_{b\in\mathcal{B}} u_b(\mathbf{s}) > 0$*. Then*

$$\mathrm{HHI}(\mathbf{s}) \leq \frac{9}{8K} = \frac{9}{8}\,\mathrm{HHI}^{\mathrm{egal}}.$$

In addition, the factor $9/8$ *is tight as a universal bound.*

Interpretation. This proposition shows that, under decentralized block building with the equal-split sharing rule, utility shares among builders cannot be much more concentrated than the egalitarian benchmark. At any pure Nash equilibrium, the utility-share HHI is at most 12.5% above the value attained when all builders earn equal utility. This follows from the equilibrium dispersion bound: the lowest-utility builder must earn at least half of the highest-utility builder's payoff.

4 Simulation Setup

The theoretical results provide worst-case guarantees that hold uniformly across model instances. Simulations allow us to ask a complementary, instance-level question: how do welfare, transaction coverage, and concentration vary with concrete source-value, latency, and builder-participation configurations? To answer this question, we compare self-interested equilibrium outcomes with centralized planner benchmarks across a set of controlled simulation environments. For reproducibility, we open-source our simulator [1].

Regions and information sources. We instantiate the region set $\mathcal{R}$ using 24 regions from Google Cloud Platform (GCP), partitioned into two source-location pools (see the appendix of the full version [37]). The high-value pool contains regions near major financial or technology centers, such as North America, Western Europe, and East Asia[2]. The peripheral pool contains regions that are geographically further from these centers. For each simulation instance, we sample the same number of source regions from the high-value and peripheral pools, without replacement, and place one information source in each selected region. Source locations remain fixed throughout a run. We denote the high-value and peripheral source clusters by $\mathcal{I}_H$ and $\mathcal{I}_P$, respectively. We normalize the total expected emitted value to $\sum_{I\in\mathcal{I}} \Lambda_I = 10$ in all reported experiments. Source-value asymmetry is parameterized by the high-to-low value share ratio γ, defined as the ratio between the total expected value emitted by $\mathcal{I}_H$ and $\mathcal{I}_P$. Thus, the two clusters receive total expected values $10 \cdot \frac{\gamma}{1+\gamma}$ and $10 \cdot \frac{1}{1+\gamma}$, respectively, divided evenly within each cluster.

Transaction and latency. To generate realized transactions, the simulator uses a Poisson arrival process with emission times sampled uniformly in $[0, \Delta]$. In the reported experiments, transaction values are deterministic and chosen so that each source has the desired expected emitted value Λ_I. As in previous literature [34, 36], the simulator also supports lognormal transaction values, but we set the value noise to zero to isolate latency and source-value effects. When sweeping Δ, we hold each Λ_I fixed, so slot-duration experiments isolate the deadline effect rather than changing total emitted value per slot.

We calibrate the propagation delays using an empirical GCP pairwise latency matrix. For a source region k and builder region i, latency $D_{k\to i}$ is modeled as lognormal with mean equal to the calibrated empirical latency and standard deviation 15% of the mean, with a small lower bound to avoid degeneracy. The corresponding reception probability is $q_k(i,t) = \Pr(D_{k\to i} \le \Delta - t)$.

Equilibrium profiles. Builders choose regions from the same 24-region set and are initialized uniformly at random. We use asynchronous better-response dynamics (ABR), which is guaranteed to converge to a pure Nash equilibrium (PNE) under the exact-potential structure of the game (cf. Corollary 2). Builders update one at a time in round-robin order. When builder b is selected, all other builders' regions are fixed; candidate regions are considered in random order, and b moves to the first region that strictly improves its expected utility. If no improving region exists, it stays.

In the simulator, utilities are evaluated numerically, and convergence is declared when a full round-robin sweep completes without any builder changing regions. In all reported experiments, every simulation run converged to a profile with no utility-improving unilateral deviation within 250 builder updates. We denote the equilibrium profile by $\mathbf{s}^{\mathrm{PNE}}$.

[2] This classification follows recent observations that Ethereum latency competition depends on relay geography and price-signal paths across North America, Europe, and East Asia [9].

To compute a builder's utility at a candidate region, we use the equal-split sharing rule directly. If builder b considers moving to region $i \in \mathcal{R}$, then for a transaction from source I emitted at time t, builder b receives the transaction with probability $q_{r(I)}(i,t)$. If $X_{I,t}^{-b}$ other builders also receive it, then builder b's share is $\frac{1}{1+X_{I,t}^{-b}}$. Thus,

$$u_b(i, \mathbf{s}_{-b}) = \sum_{I \in \mathcal{I}} \Lambda_I \int_0^{\Delta} \rho_I(t)\, q_{r(I)}(i,t)\, \mathbb{E}\left[\frac{1}{1+X_{I,t}^{-b}}\right] dt.$$

Here, $X_{I,t}^{-b}$ is the random number of builders other than b that receive a transaction from source I emitted at time t, under the profile $\mathbf{s}_{-b}$. Since receptions are independent across builders, $X_{I,t}^{-b}$ follows a Poisson-binomial distribution with parameters $\{q_{r(I)}(s_{b'}, t)\}_{b' \neq b}$. We compute the probability mass function by dynamic programming, evaluate $\mathbb{E}[1/(1+X_{I,t}^{-b})]$, and approximate the time integral using a uniform grid of points. For each parameter setting, we sample five random source layouts. For each source layout, we simulate three independently sampled initial builder profiles, giving 15 runs per parameter setting.

4.1 Benchmarks and metrics

Planner benchmark. For each simulation instance, we compare the equilibrium profile with a centralized social-planner benchmark. When exact optimization is tractable, the planner benchmark is $\mathbf{s}^* \in \arg\max_{\mathbf{s} \in \mathcal{R}^K} W(\mathbf{s})$, and $W^* := W(\mathbf{s}^*)$. We assume builders are symmetric, so exact search can be performed over occupancy vectors rather than over all labeled profiles in $\mathcal{R}^K$. The number of such occupancy vectors is $\binom{|\mathcal{R}|+K-1}{K}$. We use exact exhaustive search in the experiments with fixed number of builders $K = 5$. For experiments that sweep over larger builder counts, exact search becomes computationally expensive, so we use a greedy welfare-maximization routine as the planner benchmark. Across all tested instances, the worst observed greedy-to-exact welfare ratio is 0.999129 (see the appendix of the full version [37]). In experiments using the greedy benchmark, welfare ratios should be interpreted relative to the greedy planner benchmark rather than the exact optimum.

Metrics. The primary efficiency metric is *welfare ratio*, calculated as $\mathrm{WR} := W_{\mathrm{PNE}}/\widehat{W}^*$, where $\widehat{W}^* = W^*$ for exact planner search and equals the greedy planner welfare when the greedy benchmark is used. When $\widehat{W}^* = W^*$, the theoretical PoA ≤ 2 bound corresponds to $\mathrm{WR} \geq 1/2$.

We report utility concentration with *utility HHI* following Definition 13. This metric measures how concentrated expected builder utilities are at the selected PNE.

Geographical concentration of builder locations is measured with *geographic HHI*. Let $g_i(\mathbf{s}) := |\{b \in \mathcal{B} : s_b = i\}|/K$ be the fraction of builders located in region $i \in \mathcal{R}$. We define

$$\mathrm{HHI}_{\mathrm{geo}}(\mathbf{s}) = \sum_{i \in \mathcal{R}} g_i(\mathbf{s})^2.$$

When $K \leq |\mathcal{R}|$, its minimum value is $1/K$, attained when all builders occupy distinct regions. Its maximum value is 1, attained when all builders co-locate in a single region.

Finally, for a source cluster $G \subseteq \mathcal{I}$, we define *value-weighted cluster coverage* by

$$\mathrm{Coverage}(G; \mathbf{s}) = \frac{\sum_{I \in G} \Lambda_I \bar{f}_I(\mathbf{s})}{\sum_{I \in G} \Lambda_I}.$$

We report this metric for $G = \mathcal{I}_H$ and $G = \mathcal{I}_P$. Higher coverage means that transactions from that cluster are more likely to reach at least one builder before the deadline.

5 Simulation Results

We now study how the equilibria behave across concrete source-value and latency environments. The goal is to understand how self-interested builder location choice affects welfare, transaction coverage, geographic centralization, and builder utility concentration in equilibrium, and how these outcomes compare with those in the welfare-optimal planner benchmark. The experiments vary the two main forces in the model. Source-value asymmetry changes builders' private incentives by making some sources more attractive than others. The slot duration controls how strongly latency affects whether transactions arrive before the deadline.

Experiment 1: source-value asymmetry. We isolate the effect of source-value asymmetry by sweeping the high-to-low source value ratio at a fixed slot duration and builder count.

Experiment 2: latency deadline effect. We isolate the latency deadline effect by sweeping the slot duration while holding value asymmetry fixed.

Experiment 3: interaction effect. We study the interaction between source-value asymmetry and latency sensitivity by jointly varying the value ratio and the slot duration.

Experiment 4: builder participation. We vary the number of builders to test whether additional builders expand coverage or mainly duplicate coverage around attractive sources.

The main figures in these experiments report aggregate metrics across randomized source layouts and initializations. These summaries are useful for comparing welfare, coverage, and concentration, but they do not show the underlying builder placements in individual simulation runs. To make the placement mechanism more concrete, we visualize a representative builder placement in equilibrium and the planner benchmark for each experiment in the appendix of the full version [37].

5.1 Experiment 1: source-value asymmetry effect

This experiment isolates the effect of source-value asymmetry. We test whether increasing the value gap between high-value and peripheral sources causes builders to concentrate near high-value sources, creating redundant coverage and reducing peripheral-source coverage.

Setup. We fix $K = 5$ builders and $\Delta = 50$ms slot duration.[3] Each simulation instance contains five high-value sources and five peripheral sources, sampled from their respective region pools. We vary the high-to-low source value ratio γ from 1 to 20, while keeping the total expected emitted value fixed. Thus, increasing the ratio reallocates value toward the high-value sources without changing the total value generated.

Results. In the top-left panel of Figure 1, the welfare in equilibrium approaches that in the planner benchmark as the high-to-low value ratio increases. The welfare ratio rises toward one, indicating that builder placement in equilibrium recovers most of the optimum welfare achieved by the planner when high-value sources dominate the total value.

This improvement is not driven by region-level co-location. The bottom-left panel shows that geographic HHI remains essentially at the uniform baseline $1/K$ throughout the value-ratio sweep. In this experiment, there are exact $K = 5$ builders and five high-value sources, so builders can cover the high-value cluster in equilibrium by spreading across those

[3] We use $K = 5$ as a baseline, motivated by the fact that the current Ethereum builder market is concentrated among a few builders [28, 29, 35]. 50ms slot duration is motivated by Solana Constellation's configuration, in which proposers submit transactions and the leader assembles a batch in each cycle [22].

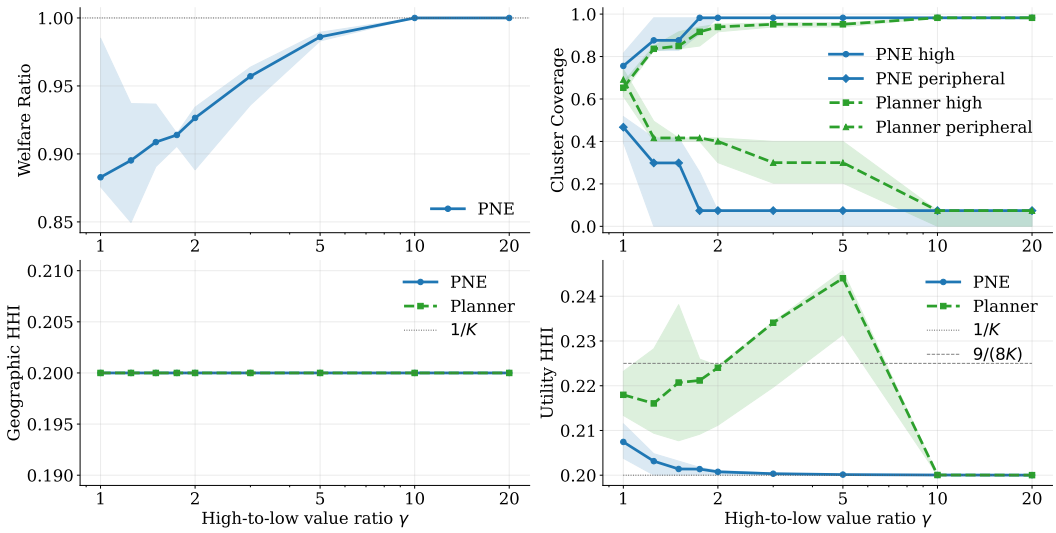


Figure 1 *Effect of source-value asymmetry.* Lines show medians across runs; shaded bands show inter-quartile ranges. As the high-to-low source value ratio increases, equilibrium welfare approaches the optimal welfare while geographic HHI remains exactly $1/K$. The main effect is coverage reallocation: at intermediate value ratios, coverage already concentrates on high-value sources in equilibrium, whereas the planner maintains peripheral source coverage and induces higher utility concentration.

regions rather than co-locating. The main effect of source-value asymmetry is therefore the reallocation of coverage. In the top-right panel, as the high-to-low value ratio increases, builders become increasingly incentivized to cover high-value sources in equilibrium: high-value cluster coverage quickly approaches one, while peripheral cluster coverage drops sharply. In the planner benchmark, the planner keeps peripheral coverage higher at intermediate value ratios, but this comes with a higher builder utility HHI (bottom-right panel), indicating that welfare-improving coverage can require allocating builders to lower-payoff regions.

Overall, in this balanced-capacity setting, source-value asymmetry reallocates coverage toward high-value sources without inducing region-level co-location: with as many high-value source regions as builders, builders can spread across distinct high-value regions. This should not be interpreted as saying that value asymmetry cannot create geographic concentration. If high-value sources are themselves geographically clustered, the same incentive to cover high-value sources may instead lead builders to concentrate in a small number of regions.

5.2 Experiment 2: slot-duration effect

Next, we fix the source-value distribution and vary the slot duration, isolating the other main force in the model: deadline-induced latency sensitivity. This experiment tests how the block-construction round duration changes latency-sensitive builder placement. When the duration is short, propagation latency has a stronger effect on whether transactions reach builders before the deadline. When the duration is long, more source-builder pairs can receive transactions in time, and geographic placement should matter less for coverage.

Setup. We fix $K = 5$ builders and high-to-low value ratio $\gamma = 10$. Each instance contains five high-value sources and five peripheral sources. We vary slot durations from 10ms to 12s. The lower end is tighter than the proposed 50ms Solana Constellation cycle [22], while the upper end matches the 12s slot duration used by Ethereum Proof-of-Stake [5]. When

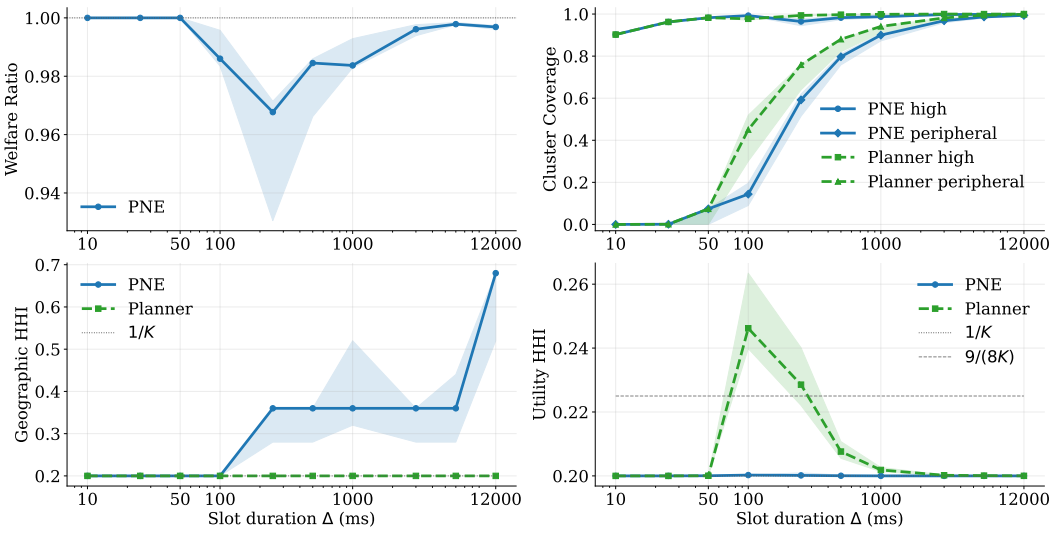


Figure 2 *Effect of slot duration.* Larger slot durations relax the latency deadline and increase source-cluster coverage. The welfare gap is largest at intermediate durations, where peripheral coverage is feasible in the planner benchmark but not reliably supplied in equilibrium. At long durations, coverage is nearly saturated, so geographic concentration has limited welfare cost even when the builder placement in equilibrium remains more concentrated than in the planner benchmark.

sweeping slot durations Δ, we hold each source's expected emitted value Λ_I fixed, so the experiment isolates the effect of the deadline on transaction reception rather than changing the amount of value generated per round.

Results. Figure 2 shows a non-monotone effect of slot duration. In the top-left panel, when slots are very short, the welfare ratio is close to one, but this does not indicate broad coverage. Rather, the deadline is so tight that builders can effectively capture value only from nearby sources. Since high-value sources dominate the value distribution in this experiment, both the planner and the equilibrium incentive place builders near high-value source regions, leaving peripheral sources mostly uncovered.

At intermediate slot durations, peripheral sources become reachable for some regions, but coverage still depends strongly on builder placement. This is where the welfare ratio falls: the planner spreads builders across regions to improve coverage of different source clusters, while in equilibrium, builders are concentrated around regions with better access to high-value sources. The top-right cluster-coverage panel shows the resulting mechanism: high-value coverage is already near saturation, whereas peripheral coverage in equilibrium lags behind that in the planner benchmark. Thus, the welfare gap comes primarily from under-coverage of peripheral sources in equilibrium, induced by redundant coverage around already attractive high-value sources. The utility HHI panel indicates that this welfare gap is not primarily an effect of payoff inequality: builder utilities remain relatively balanced even when peripheral coverage is under-provided.

This transition is best interpreted relative to the latency scale of the underlying region graph (see the appendix of the full version [37]). The largest welfare losses arise in the intermediate regime, where Δ is comparable to cross-region latencies: peripheral coverage is feasible, but only for appropriately placed builders. When Δ is much smaller than typical cross-region propagation delays, builders can capture value only from nearby sources, so

builders have limited ability to cover peripheral sources in both the equilibrium and the planner benchmark. When Δ is much larger than these latencies, most source-builder pairs can communicate before the deadline, so transaction coverage becomes less sensitive to builder geography. In this long-slot regime, geographic centralization may persist, but it no longer produces a large welfare loss in this model because coverage is close to saturated.

We repeat this experiment with a lower value ratio of $\gamma = 2$, where peripheral sources contribute to a larger share of the total value. The qualitative patterns are similar, but the welfare gap is larger at intermediate slot durations, where peripheral sources are valuable enough to matter and reachable enough for the planner to cover. We provide the figure and discussion in the appendix of the full version [37].

5.3 Experiment 3: value-asymmetry–slot-duration interaction effect

Having isolated source-value asymmetry and slot duration separately, we now study their interaction. The goal is to identify which combinations produce inefficient welfare outcomes, weak transaction coverage, or concentrated builder placement in equilibrium.

Setup. Each simulation instance contains five builders, five high-value sources, and five peripheral sources, sampled from their respective region pools. We sweep the high-to-low source value ratio γ from 1 to 20 and the slot duration from 10ms to 12s.

Results. Figure 3 shows that source-value asymmetry and slot duration jointly determine how close the equilibrium outcome is to the welfare-optimal outcome, and reinforces our prior results. The welfare ratio is primarily non-monotone in the slot duration. At both extremes, welfare in equilibrium is close to that in the planner benchmark. The largest welfare loss occurs at intermediate slot durations, where peripheral sources are reachable from certain regions but coverage remains sensitive to builder placement. The value ratio affects how costly this peripheral under-coverage is. When the high-to-low value ratio is small, and the peripheral sources meaningfully contribute to the total emitted value, the welfare loss becomes larger in equilibrium. As the value ratio increases, high-value sources dominate the total emitted value, so the welfare penalty from leaving peripheral sources under-covered becomes smaller.

The coverage-gap surfaces identify the mechanism. A positive gap means the corresponding source cluster is better covered in the planner benchmark than in equilibrium. When the value ratio $\gamma = 1$, the two source clusters have equal expected value, so the labels "high-value" and "peripheral" distinguish geography rather than value. In this case, planner advantage in one cluster and equilibrium advantage in the other need not create a large welfare gap.

As the value ratio increases, the high-value cluster receives a larger share of total value, and builders in equilibrium have stronger incentives to prioritize regions with access to high-value sources. At intermediate slot durations, peripheral sources are still reachable from some regions, so the planner can recover additional welfare by allocating builders to improve peripheral coverage. By contrast, peripheral coverage is under-provided in equilibrium, and builder placement remains tilted toward high-value sources, contributing to a negative coverage gap in high-value source coverage, a positive gap in peripheral source coverage, and a corresponding welfare penalty.

The HHI surfaces are consistent with this mechanism. Utility concentration becomes higher in the planner benchmark because welfare-improving allocations yield lower individual payoffs for builders placed away from high-value sources. Conversely, the utility HHI in

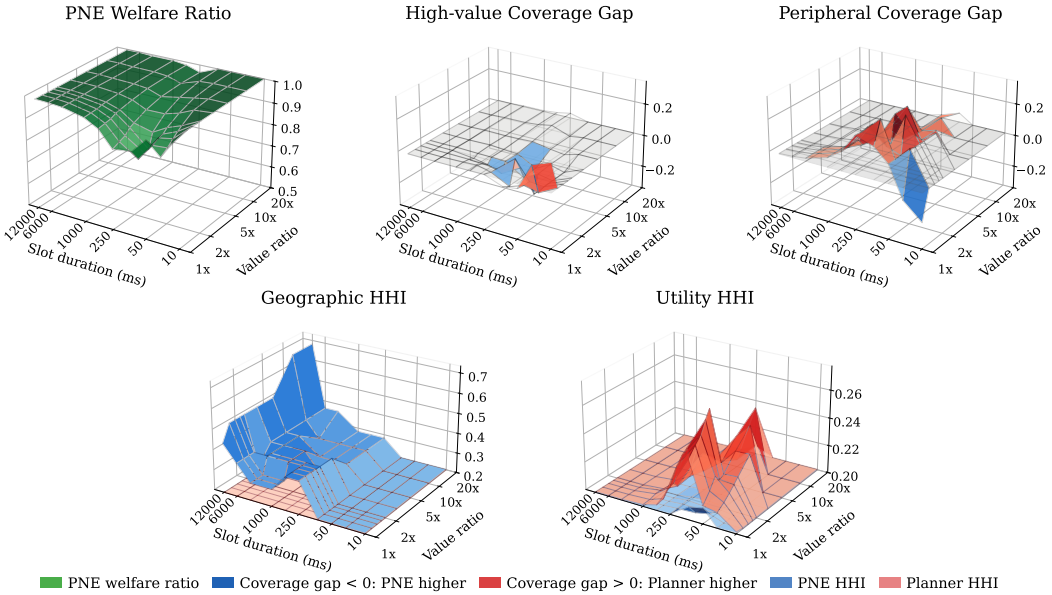


Figure 3 *Joint effect of source-value asymmetry and slot duration.* Surfaces show medians across source layouts and initializations for fixed $K = 5$. Darker colors indicate higher absolute values. Coverage gaps are defined as planner coverage minus equilibrium coverage, so positive gaps indicate that the planner covers more of the corresponding source cluster. The HHI panels compare geographic and utility concentration in equilibrium and in the planner benchmark. Welfare losses are largest at intermediate slot durations and lower-to-moderate value ratios.

equilibrium remains balanced and close to the egalitarian baseline in much of the grid, but it comes with a more geographically centralized builder placement around the high-value sources.

5.4 Experiment 4: builder participation effect

Previous experiments isolate the effects of source-value asymmetry and slot duration, as well as their interaction. We now vary the number of builders to test whether larger builder sets expand source coverage or mainly duplicate coverage around already attractive sources. Adding builders increases the system's potential coverage capacity, but when source values are asymmetric, additional builders may still prefer regions with strong access to high-value sources.

Setup. We vary the number of builders from $K = 3$ to $K = 12$. We fix the high-to-low value ratio at $\gamma = 10$ and the slot duration at $\Delta = 50$ms. For this builder-count sweep, we use the greedy welfare-maximization routine as the planner benchmark, since the cost of exact search grows rapidly with the builder count K.

Results. Figure 4 shows that increasing the number of builders does not automatically yield broader source coverage in equilibrium. In the top-left panel, the welfare ratio is non-monotone in K. When the number of builders is small, additional builders mainly improve high-value source coverage, so the welfare in equilibrium quickly moves closer to the optimal welfare. Once high-value coverage is nearly saturated, further welfare gains require better peripheral-source coverage.

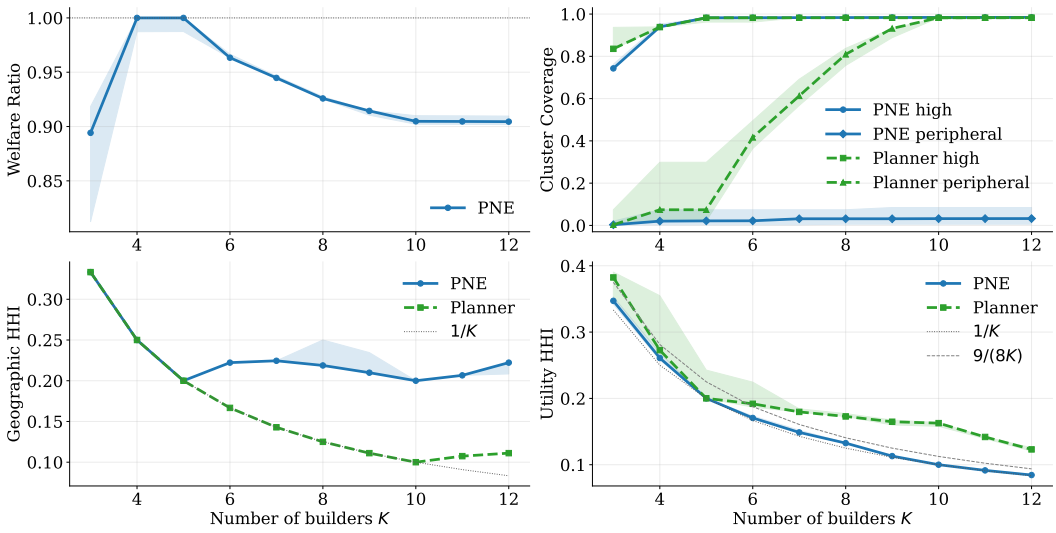


Figure 4 *Effect of builder participation.* As the number of builders K increases, high-value source coverage is improved in equilibrium, while peripheral coverage remains low, contributing to the welfare ratio first increasing then declining. Geographic HHI shows that the builder locations remain more spatially concentrated in equilibrium than the planner benchmark for larger K, while utility HHI remains comparatively close to the egalitarian baseline.

The top-right cluster-coverage panel shows that the planner uses additional builders to expand peripheral-source coverage. In equilibrium, however, they remain attracted to regions with strong access to high-value sources, even after those sources are nearly saturated. As a result, larger builder sets can generate redundant high-value coverage rather than broad source coverage, explaining why the welfare ratio first improves and then declines.

The HHI panels reveal a tradeoff between geographic decentralization and utility equality. As the number of builders increases, geographic HHI moves closer to the minimum $1/K$ in the planner benchmark, because additional builders expand coverage toward peripheral sources. Builder placement in equilibrium remains more geographically centralized: additional builders continue to locate in regions with stronger access to the high-value cluster, even after high-value coverage is nearly saturated.

The pattern is reversed for utility concentration. Utility HHI is closer to the egalitarian baseline in equilibrium, while it is higher in the planner benchmark. Improved peripheral source coverage in the planner benchmark achieves higher total welfare, but yields lower individual utility for those builders located in the peripheral regions. By contrast, in equilibrium, builders are incentivized to locate in high-value regions, where equal sharing can make payoffs more balanced. Thus, geographic decentralization and builder-utility equality need not coincide: improving source coverage and total welfare may require distributing builders into lower-payoff regions, while selfish behavior can achieve a more utility-balanced but geographically more centralized outcome in equilibrium.

We repeat this experiment with a lower value ratio of $\gamma = 2$, where peripheral sources contribute to a larger share of the total value. The qualitative patterns are similar, but the welfare gap is more pronounced. We provide the figure and discussion in the appendix of the full version [37].

6 Discussion

We now discuss the implications of our results for decentralized block-building design. The central message is that decentralization is not only a question of how many builders participate, but also of where self-interested builders choose to locate. In our model, the *price of decentralization* measures the welfare cost of decentralized, self-interested location decisions relative to a centrally coordinated placement that maximizes transaction coverage. Decentralized decisions may themselves produce a geographically concentrated outcome. We then discuss directions for future work and limitations.

6.1 Implications for decentralized block-building design

Our theory and simulations point to a structural feature of decentralized block-building or MCP designs. Multiple builders reduce the monopoly power of a single proposer, but if their location choices are uncoordinated, equilibrium placement can still duplicate coverage around privately attractive high-value sources, weakening the system's censorship-resistance and fair-access guarantees.

The simulations identify where this price of decentralization is most visible. Welfare losses are largest in intermediate regimes where peripheral sources are reachable and still contribute meaningful value, but decentralized equilibrium incentives still favor regions with strong access to high-value sources. Thus, inefficiency is not driven simply by source-value asymmetry or by latency sensitivity alone, but by their interaction: peripheral coverage must be feasible enough to matter, while still requiring builders to locate away from the most privately attractive regions.

From a protocol-design perspective, the relevant timing parameter is not slot duration in isolation, but slot duration relative to transaction propagation latencies. Longer slot durations or more relaxed (sub-)block construction deadlines can make transaction coverage less geography-dependent in our model, but geographic centralization may persist when its welfare penalty becomes small. Conversely, shorter deadlines make source–builder proximity more important, while transaction coverage is restricted to be local, limiting coverage of distant sources. The fragile regime lies between these extremes, where cross-region coverage is possible but still placement-sensitive.

Increasing builder participation expands potential coverage capacity. However, in the finite regimes our simulations examine (up to 12 builders), additional builders may not cover peripheral sources immediately. When individual incentives favor high-value source access, extra builders can duplicate coverage around already attractive regions. The results also show that geographic and utility concentration capture different objectives. Planner allocations may be geographically more dispersed but utility-wise more unequal, because the planner places builders in lower-payoff peripheral regions to improve total coverage. Equilibrium profiles may instead be more geographically concentrated while keeping builder utilities relatively balanced under equal sharing.

Since source-value asymmetry is largely an environmental primitive, while timing, builder participation, and reward-sharing rules are at least partly protocol design choices, these parameters are natural levers for shaping transaction coverage incentives. Overall, protocol designers should distinguish between maximizing welfare and coverage, improving geographic decentralization, and equalizing builder rewards, since these objectives need not move together.

6.2 Future work

Location-market modeling. Our model treats decentralized block building as a stochastic coverage game. Transaction sources are exogenous: they emit transactions, builders choose regions, and latency determines whether each transaction reaches at least one builder before the deadline. This formulation is useful for studying source-side coverage and the welfare loss from redundant coverage of the same sources.

A richer model would endogenize transaction routing. In such a model, users or applications choose which builders to send transactions to, and the builder location affects which transactions a builder can serve with sufficiently low latency. Builder rewards would then depend not only on which sources are reachable, but also on how users route order flow and how builders compete for that flow. This would make the model closer to a location-market game than a pure coverage game.

Such a formulation may be important for protocols and applications in which latency-sensitive users actively choose among builders, such as the Solana ecosystem with Constellation [22]. High-value order-flow sources, such as exchanges or latency-sensitive application servers, may emerge as strategic hubs: co-locating builders may receive higher-quality order flow or offer stronger latency guarantees. We leave the analysis of endogenous transaction routing and builder competition for future work.

Alternative reward-sharing rules. Our analysis focuses on equal splitting among builders who receive and include a transaction. Future work could compare this rule with other reward-sharing mechanisms. Winner-takes-all or unique-inclusion rules may reduce duplicate inclusion, but can strengthen incentives for exclusivity, private routing, and reward concentration. Broader committee-level sharing [15] may encourage builders to maximize aggregate source coverage, but can create free-riding incentives if rewards are paid to builders that do not contribute to coverage.

Another natural extension is contribution-weighted or Tullock-style sharing. The equal-split rule can be viewed as the uniform-score special case of proportional sharing. More general rules could assign contribution scores, for example, based on the source-builder propagation latency, and allocate value proportionally to the score. Such rules interpolate between egalitarian sharing and winner-takes-all allocation, and relate to work on the blockspace distribution and contest-style mechanisms [17, 18, 27]. Studying their effects on welfare, geographic centralization, and reward equality is an important direction for future work.

6.3 Scope and limitations

We close by clarifying the scope of the model and simulations. The goal of the paper is not to provide a full protocol-level simulator, but to isolate how latency-sensitive transaction value shapes the region choices of self-interested builders in decentralized block-building.

First, our simulations use asynchronous better-response dynamics to select a pure Nash equilibrium. We use this dynamic because it is theoretically aligned with the exact-potential structure of the game under equal sharing and gives a stable equilibrium benchmark. Simultaneous updates or learning dynamics may cycle or induce non-stationary empirical distributions; while studying such out-of-equilibrium behavior is an interesting direction, it is outside the scope of this paper. We use ABR as an equilibrium-selection procedure, not as a literal description of how all builders relocate in practice. The underlying incentive tension remains the same: builders prefer regions with stronger access to certain high-value sources, while the system may benefit from broader source coverage.

Our exact-potential argument guarantees finite convergence of strict asynchronous better-response dynamics, but it does not provide a polynomial bound on the length of improvement paths. Thus, the complexity of computing a pure Nash equilibrium in the builder region game is a natural direction for future work. For finite, explicitly represented instances with polynomial-time evaluable utilities, pure Nash equilibrium computation can be viewed as a local-search problem over the potential function, suggesting that a formal PLS analysis may be appropriate. In our simulations, however, ABR served only as an equilibrium-selection procedure, and all reported runs converged to a no-improving-deviation profile within 250 builder updates. Moreover, the smoothness guarantee in Theorem 7 extends beyond pure Nash equilibria to weaker equilibrium notions, so the factor-2 efficiency bound is not tied only to the ABR selection procedure.

Second, we abstract from many communication details. We model source-to-builder latency directly and use a common deadline for transaction reception. In practice, protocols may involve peer-to-peer propagation, intermediaries (e.g., relays), committee attestation, and leader aggregation. These features would require a richer model of consensus-layer and network communication. However, they do not remove the central mechanism studied here: location affects timely access to payoff-relevant information. Additional latency-sensitive targets may, in fact, create more location incentives [36], reinforcing the same kind of concentration pressure.

Third, our simulations use stylized source locations and value asymmetries rather than direct empirical estimates of order-flow geography. Such calibration is difficult because high-value transaction sources and their value distribution are not directly observable. We therefore use empirical observations on geographic latency competition in Ethereum block building [9] to instantiate the region graph and sweep broad ranges of value asymmetry, slot duration, and builder participation. The simulation results should be read as comparative statics over plausible environments, rather than as a point estimate for a particular network.

Finally, we omit several strategic features that are important for full protocol design. We impose no block-capacity constraint, fix the builder set, and rule out Sybil behavior and collusion. Capacity constraints would couple transaction selection decisions and may strengthen the welfare-loss channel we identify, because redundant inclusion of already-covered high-value transactions could crowd out otherwise uncovered transactions. Entry and rotation would affect which builders are active in a given round and could change equilibrium selection. Sybil behavior and collusion would require a richer model of coalitional incentives, where coordinated builders may improve welfare by avoiding redundant coverage, but they could also undermine the censorship-resistance benefits of decentralized block building by behaving as a single economic entity. These extensions are therefore important for full protocol design, but complementary to the location-choice mechanism studied here, which focuses on the inefficiency of uncoordinated self-interested placement.

7 Conclusion

We studied location choice in decentralized block building by modeling it as a stochastic coverage game. Our results show that replacing a single proposer with multiple builders is not sufficient, by itself, for broad transaction coverage or welfare-efficient placement. Under equal sharing, the builder region game admits a pure Nash equilibrium, has an asymptotically tight factor-2 PoA bound, and satisfies tight bounds on builder utility concentration. Our simulations complement these worst-case guarantees by showing that welfare losses are most pronounced in intermediate regimes where peripheral sources are reachable and valuable, but

self-interested builders still prefer regions with strong access to high-value sources. Overall, the paper highlights that the benefits of decentralized block building depend not only on the number of builders, but also on where builders locate and how protocol rules shape their incentives. Timing parameters, participation rules, and reward-sharing mechanisms can therefore affect the price of decentralization, especially in networks with asymmetric source values and latency-sensitive transaction propagation.

References

1 Authors. Decentralized Block Building Simulator, 2026. URL: https://github.com/boz1/db-sims.

2 Maryam Bahrani, Pranav Garimidi, and Tim Roughgarden. Transaction Fee Mechanism Design in a Post-MEV World. In Rainer Böhme and Lucianna Kiffer, editors, *6th Conference on Advances in Financial Technologies (AFT 2024)*, volume 316 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 29:1–29:24, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.AFT.2024.29.

3 Davide Bilò, Tobias Friedrich, Pascal Lenzner, Anna Melnichenko, and Louise Molitor. Fair tree connection games with topology-dependent edge cost. In Nitin Saxena and Sunil Simon, editors, *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2020)*, volume 182 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.FSTTCS.2020.15.

4 Simina Brânzei, Ioannis Caragiannis, Jamie Morgenstern, and Ariel D. Procaccia. How bad is selfish voting? In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, pages 138–144. AAAI Press, 2013. doi:10.1609/aaai.v27i1.8667.

5 Vitalik Buterin, Diego Hernandez, Thor Kamphefner, Khiem Pham, Zhi Qiao, Danny Ryan, Juhyeok Sin, Ying Wang, and Yan X Zhang. Combining ghost and casper. *arXiv preprint arXiv:2003.03052*, 2020. arXiv:2003.03052.

6 Cambridge Centre for Alternative Finance (CCAF). Ethereum network analytics. https://ccaf.io/cbnsi/ethereum/network_analytics, 2026. Accessed: 2026-05-09.

7 Riccardo Colini-Baldeschi, Roberto Cominetti, Panayotis Mertikopoulos, and Marco Scarsini. The asymptotic behavior of the price of anarchy. In *Web and Internet Economics: 13th International Conference, WINE 2017, Bangalore, India, December 17–20, 2017, Proceedings*, pages 133–145, Berlin, Heidelberg, 2017. Springer-Verlag. doi:10.1007/978-3-319-71924-5_10.

8 Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash Boys 2.0: Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability. In *2020 IEEE symposium on security and privacy (SP)*, pages 910–927, Los Alamitos, CA, USA, 2020. IEEE Computer Society. doi:10.1109/SP40000.2020.00040.

9 Data Always. The Geography of Block Building. The Flashbots Collective, November 2025. Flashbots. URL: https://collective.flashbots.net/t/the-geography-of-block-building/5367.

10 Fatima Elsheimy, Ioannis Kaklamanis, Sarisht Wadhwa, Charalampos Papamanthou, and Fan Zhang. Censorship Resistance vs Throughput in Multi-Proposer BFT Protocols. *Cryptology ePrint Archive*, 2026.

11 ethernodes.org. Ethernodes – countries. https://www.ethernodes.org/countries, 2026. Accessed: 2026-05-09.

12 Michal Feldman, Kevin Lai, and Li Zhang. The proportional-share allocation market for computational resources. *IEEE Transactions on Parallel and Distributed Systems*, 20(8):1075–1088, 2009. doi:10.1109/TPDS.2008.168.

13 Flashbots. decentralized building: wat do? https://writings.flashbots.net/decentralized-building-wat-do, 2026. Accessed: 2026-05-09.

14 Martin Gairing. Covering games: Approximation through non-cooperation. In Stefano Leonardi, editor, *Internet and Network Economics*, volume 5929, pages 184–195. Springer Berlin Heidelberg, 2009. Series Title: Lecture Notes in Computer Science. doi:10.1007/978-3-642-10841-9_18.

15 Pranav Garimidi, Lioba Heimbach, and Tim Roughgarden. Transaction fee mechanism design for leaderless blockchain protocols. In *Financial Cryptography and Data Security: 29th International Conference, FC 2025, Miyakojima, Japan, April 14–18, 2025, Revised Selected Papers, Part II*, pages 20–35, Berlin, Heidelberg, 2025. Springer-Verlag. doi:10.1007/978-3-032-07035-7_2.

16 Pranav Garimidi, Joachim Neu, and Max Resnick. Multiple concurrent proposers: Why and how. Cryptology ePrint Archive, Paper 2025/1772, 2025. URL: https://eprint.iacr.org/2025/1772.

17 Pranav Garimidi, Michael Neuder, and Tim Roughgarden. Tullock contests in the wild: Applications in blockchains. *ACM SIGecom Exchanges*, 23(1):24–34, July 2025. doi:10.1145/3821553.3821558.

18 Pranav Garimidi, Michael Neuder, and Tim Roughgarden. Beyond winner-take-all procurement auctions. In *Financial Cryptography and Data Security: 30th International Conference, FC 2026*, 2026. URL: https://ifca.ai/fc26/preproceedings/231.pdf.

19 Lioba Heimbach, Yann Vonlanthen, Juan Villacis, Lucianna Kiffer, and Roger Wattenhofer. Deanonymizing Ethereum Validators: The P2P Network Has a Privacy Issue. In *34th USENIX Security Symposium*, Seattle, WA, USA, 2025. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity25/presentation/heimbach.

20 Lucianna Kiffer, Lioba Heimbach, Dennis Trautwein, Yann Vonlanthen, and Oliver Gasser. Multiple Sides of 36 Coins: Measuring Peer-to-Peer Infrastructure Across Cryptocurrencies. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 9(3):1–32, 2025. doi:10.1145/3771572.

21 Seoung Kyun Kim, Zane Ma, Siddharth Murali, Joshua Mason, Andrew Miller, and Michael Bailey. Measuring Ethereum Network Peers. In *Proceedings of the Internet Measurement Conference 2018*, pages 91–104, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3278532.3278542.

22 Quentin Kniep, Max Resnick, Jakub Sliwinski, and Roger Wattenhofer. Solana constellation: Internet capital markets. White paper, Anza, 2026.

23 Robert Miller. Parallel block building. https://writings.flashbots.net/parallel-builder, 2024. Accessed: 2026-05-09.

24 Ciamac C. Moallemi, Mallesh M. Pai, and Dan Robinson. Latency advantages in common-value auctions, 2026. arXiv:2504.02077.

25 Dov Monderer and Lloyd S. Shapley. Potential Games. *Games and Economic Behavior*, 14(1):124–143, 1996. doi:10.1006/game.1996.0044.

26 Barnabé Monnot, Francisco Benita, and Georgios Piliouras. How bad is selfish routing in practice? arXiv preprint arXiv:1703.01599, 2017. doi:10.48550/arXiv.1703.01599.

27 Michael Neuder, Pranav Garimidi, and Tim Roughgarden. On block-space distribution mechanisms, 2024. Accessed: 2026-05-25. URL: https://ethresear.ch/t/on-block-space-distribution-mechanisms/19764.

28 Burak Öz, Danning Sui, Thomas Thiery, and Florian Matthes. Who Wins Ethereum Block Building Auctions and Why? In Rainer Böhme and Lucianna Kiffer, editors, *6th Conference on Advances in Financial Technologies (AFT 2024)*, volume 316 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:25, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.AFT.2024.22.

29 relayscan.io. MEV-Boost Relay & Builder Stats. https://www.relayscan.io/, 2026. Accessed: 2026-05-18.

30 Marc Roeschlin, Evangelos Markakis, Raghav Bhaskar, and Aggelos Kiayias. Incentivizing Geographic Diversity for Decentralized Systems. *Cryptology ePrint Archive*, 2026.

31 Tim Roughgarden. Intrinsic Robustness of the Price of Anarchy. *J. ACM*, 62(5), November 2015. doi:10.1145/2806883.

32 Adrian Vetta. Nash Equilibria in Competitive Societies, with Applications to Facility Location, Traffic Routing and Auctions . In *The 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 416–425, Los Alamitos, CA, USA, 2002. IEEE, IEEE Computer Society. doi:10.1109/SFCS.2002.1181966.

33 Anton Wahrstätter, Jens Ernstberger, Aviv Yaish, Liyi Zhou, Kaihua Qin, Taro Tsuchiya, Sebastian Steinhorst, Davor Svetinovic, Nicolas Christin, and Mikolaj Barczentewicz. Blockchain censorship. *arXiv preprint arXiv:2305.18545*, 2023. doi:10.48550/arXiv.2305.18545.

34 Fei Wu, Thomas Thiery, Stefanos Leonardos, and Carmine Ventre. Strategic bidding wars in on-chain auctions. In *2024 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 503–511, Dublin, Ireland, 2024. IEEE. doi:10.1109/ICBC59979.2024.10634354.

35 Sen Yang, Kartik Nayak, and Fan Zhang. Decentralization of Ethereum's Builder Market. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 1456–1474, Los Alamitos, CA, USA, 2025. IEEE Computer Society. doi:10.1109/SP61157.2025.00157.

36 Sen Yang, Burak Öz, Fei Wu, and Fan Zhang. Geographical Centralization Resilience in Ethereum's Block-Building Paradigms. *Proc. ACM Meas. Anal. Comput. Syst.*, 10(2), June 2026. doi:10.1145/3805637.

37 Burak Öz, Fei Wu, Luis Correia, Sen Yang, Bruno Mazorra, and Stefanos Leonardos. The price of decentralization in block building, 2026. doi:10.48550/arXiv.2606.01874.

Price of Censorship: Censorship Resistance and Throughput Under Rational Concurrent Proposers

Aditya Saraf
Cornell University, Ithaca, NY, USA

Ioannis Kaklamanis
Yale University, IC3, New Haven, CT, USA

Sarisht Wadhwa
Duke University, Durham, NC, USA

Fatima Elsheimy
Yale University, New Haven, CT, USA

Abstract

Censorship resistance is the defining advantage of blockchains over their centralized counterparts. Yet block proposers censor transactions for many reasons, from legal consequences to economic incentives. We study *economically-incentivized* censorship, modeled by an adversary who bribes proposers to exclude a target transaction, and define the *economic censorship resistance* (eCR) of a transaction as the adversary's expected cost of successful censorship divided by the user's expected payment for inclusion. Single-proposer systems are structurally weak by this measure: under a first-price auction the adversary need only match the user's bid, and fee burning pushes eCR to a few percent of what the user pays.

We therefore turn to multiple concurrent proposers (MCP), where block capacity is divided among n proposers and the block is the union of their sub-blocks.While MCP can substantially increase the cost of censorship by requiring the adversary to bribe many proposers, it also introduces transaction duplication, reducing throughput. The resulting trade-off depends critically on the transaction fee mechanism (TFM), which determines how fees are shared among competing proposers.

We create a game theoretic model where validators construct blocks from a shared mempool, subject to an adversary's bribery attempt. We provide an algorithm that solves for the mixed equilibrium of a given mempool, which is characterized by the probability of including each transaction. This algorithm works for a wide class of TFMs, and allows us to calculate the expected throughput and censorship resistance for any bid distribution. We then use simulations to show how the eCR and throughput vary as the number of proposers increases. We compare three TFMs, finding that the *duplication-penalizing* TFM dominates the others across many settings. We also validate our findings with empirical Ethereum data.

We find that MCP systems enjoy much higher eCR than single proposer systems. While burnt user fees can significantly lower eCR, eCR scales linearly with the number of validators, and thus strong eCR can be achieved even in the presence of significant burn. Overall, our results show that MCP systems are, with the correct TFM, significantly more robust to censorship.

2012 ACM Subject Classification Security and privacy → Economics of security and privacy

Keywords and phrases blockchains, multiple concurrent proposers, censorship resistance

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.26

Related Version *Full Version*: `https://arxiv.org/abs/2607.16995` [29]

Supplementary Material *Dataset (Source code and data)*: `https://zenodo.org/records/21831640`

Funding *Aditya Saraf*: AFOSR Grant FA9550-24-1-0267
Sarisht Wadhwa: NSF Grant CNS-2237814
Fatima Elsheimy: NSF Grant CNS-2247308, CNS-2153768

Acknowledgements Ioannis thanks IC3 for a research gift. Fatima thanks Center for Algorithms, Data, and Market Design at Yale (CADMY) and Protocol Labs.



© Aditya Saraf, Ioannis Kaklamanis, Sarisht Wadhwa, and Fatima Elsheimy;
licensed under Creative Commons License CC-BY 4.0



1 Introduction

Blockchains are decentralized across many blocks, but within a single block they are anything but: the current proposer is a monopolist with complete control over the block's contents. Decentralization is meant to give every transaction in the mempool a "fair chance" at inclusion, relative to its bid, unlike centralized systems, where the operator can simply refuse to process a payment or a post [15]. In practice, the single-slot monopoly undermines this promise. Wahrstätter et al. [38] measure that 46% of post-merge Ethereum blocks were produced by actors censoring OFAC-sanctioned transactions, and as of early 2026, live dashboards report that roughly 35% of Ethereum builders and half of all relays engage in some form of transaction censorship [11]. These measurements suggest that transaction censorship is a persistent feature of today's blockchain ecosystem.

Some of this censorship is legally motivated, and its cost is mainly latency: a transaction censored by regulated proposers waits until a validator in a different jurisdiction proposes. Economically motivated censorship is more corrosive, because the censor can profit from every slot of delay. Consider a participant in an on-chain auction, a DeFi liquidation, or an arbitrage race: excluding a competitor's transaction for even a few blocks can be worth orders of magnitude more than the fees that transaction pays. Since a bribe only needs to outbid the proposer's *fee* from including the transaction, not the value the transaction protects, censorship is profitable precisely when value at stake far exceeds fees, which is the defining regime of maximal extractable value (MEV). An attacker can thus censor a competitor indefinitely by repeatedly offering each proposer a modest bribe.

In this paper, we focus on such *economically incentivized censorship*, which we model as an adversary who publicly offers every proposer a bribe for excluding a target transaction. We quantify a system's resilience by its *economic censorship resistance* (eCR): the ratio between the adversary's expected total expenditure for successful censorship and the fee the user expects to pay for inclusion. For example, if a user expects to pay 1 ETH, but an adversary must spend 5 ETH to censor the transaction, then the eCR is 5.

The eCR of existing designs is inherently limited. With a single proposer per block and a first-price auction transaction fee mechanism (TFM), the eCR is at most 1: the adversary simply matches the user's bid. Fee burning makes matters strictly worse. On Ethereum, the proposer receives only the tip while the base fee is burned; in typical network conditions, tips are on the order of 0.5–2 gwei against base fees of 10–50 gwei, yielding $\text{eCR} \approx \frac{\text{tip}}{\text{base fee}+\text{tip}} \in [0.02, 0.1]$. An adversary can censor a transaction on today's Ethereum by paying a few percent of what the user pays.

Multiple concurrent proposers. To achieve higher censorship resistance, we must break the proposer's monopoly by having multiple concurrent proposers (MCP). MCP designs range from research protocols such as MirBFT [31], to DAG-based systems deployed in production by Sui and Aptos [24, 6, 4], to active proposals for Ethereum and Solana [21].

We propose a simple MCP model that captures many of these systems: the total block capacity is divided among n concurrent proposers, and the final block is the union of their sub-blocks. Some transactions may then be included by multiple proposers, which yields two complications. First, the TFM must dictate how a transaction's fee is split among the proposers. [20] analyzes a TFM that pools all tips and divides them equally among proposers, regardless of who includes which transactions. [17] solicits a large and a small tip from each user: a transaction's sole includer receives the large tip, while multiple includers each receive the small one. As the Ethereum calculation above suggests, this choice heavily affects

censorship resistance. We mostly analyze an extreme version of the TFM of [17], which we term the *duplication-penalizing* TFM: it solicits a single tip and pays it to the includer only if the transaction is included exactly once. Our equilibrium analysis, however, holds for a large family of TFMs including all these examples.

The second complication is that multiple inclusions cause a loss of *throughput*, which we define as the percentage of *unique* transactions in a block. There is a natural tension between censorship resistance and throughput: doubling the number of proposers increases the amount the adversary must spend, but also creates more opportunities for duplication. The TFM again matters, as different TFMs discourage duplication to different degrees.

MCP systems are adopted precisely for these two advertised benefits: higher throughput, from parallel block production, and censorship resistance, from the fact that no single proposer can unilaterally exclude a transaction. But a design cannot simply maximize both. The redundancy that protects transactions manifests, in equilibrium, as duplication that consumes the very capacity parallelism creates, and where a system lands on this frontier depends directly on its transaction fee mechanism. This dependence frames the central question of our work:

> *Which transaction fee mechanism best navigates the trade-off between censorship resistance and throughput, and how do both scale with the number of concurrent proposers?*

Main contributions

To answer this question, we perform a game-theoretic analysis: due to the possibility of transaction duplication, a proposer's utility from their block depends on the other proposers. Our threat model is deliberately simple and stated up front: the adversary announces a public, unconditional bribe paid to any proposer that excludes the target transaction; proposers are rational, do not collude (collusion is typically equivalent to a single proposer), and build their blocks simultaneously from a shared mempool; and user tips are exogenous. Richer adversaries (e.g., bribes conditional on successful censorship) and endogenous user bidding are natural extensions that we discuss but do not model.

We solve this game algorithmically for its unique symmetric mixed Nash equilibrium, which specifies the probability of including each transaction as a function of the bids. Our algorithm is a one-dimensional bisection, is efficient, and applies to a wide family of TFMs. A "cut-and-paste" argument then yields closed-form expressions for the bribe required to censor any target transaction with any target probability.

With this machinery, we compute the expected throughput, user payments, and cost of bribery for any bid distribution. On synthetic (Pareto) bids, we vary the number of proposers, the block congestion, the adversary's target censorship probability, and the amount of burn, comparing three TFMs: duplication-penalizing, even-split, and collective. The duplication-penalizing TFM consistently dominates the other two on both eCR and throughput. We then validate these findings on the empirical tip distribution of 10,000 recent Ethereum blocks.

Quantitatively, we find that eCR scales linearly with the number of proposers, while throughput quickly plateaus near the optimistic bound achieved by uncoordinated uniform sampling; thus, beyond the one-time duplication cost of moving away from a single proposer, adding proposers buys censorship resistance at essentially no additional throughput cost. This scaling survives fee burning: at Ethereum's early-2026 burn levels (base fee $\approx 8.7\times$ the average tip), a single proposer yields eCR ≈ 0.09, whereas 49 concurrent proposers yield eCR ≈ 2.5; the adversary must outspend the user rather than pay a few percent of their fee. Overall, our results show that MCP, equipped with the right TFM, is a promising structural remedy to economic censorship.

2 Related Work

Protocol-level censorship resistance. Classical analyses of blockchain protocols establish *liveness*: every valid transaction is eventually included [19, 43, 26]; see [2] for a survey of the censorship-resistance guarantees of widely-used consensus protocols. Eventual inclusion, however, is too weak for financial applications, motivating *short-term* censorship resistance in the Byzantine model. BigDipper [41] adds short-term censorship resistance to leader-based BFT, and Abraham, Efron, and Ren [1] prove inherent latency costs of censorship resistance. Closest to our setting, concurrent work by Elsheimy et al. [16] formalizes the censorship-resistance/throughput trade-off in multi-proposer BFT protocols with up to f Byzantine parties, and designs transaction-assignment rules that navigate this trade-off. This line of work bounds the *worst-case delay* that faulty parties can impose; in contrast, we assume all proposers are rational and instead *price* censorship under bribery. The two perspectives are complementary: Byzantine analyses cover arbitrary deviations by a bounded fraction of parties, while ours covers economically-motivated deviations by all parties.

Transaction fee mechanisms. Transaction fee mechanisms (TFMs) in blockchain systems have evolved from first-price auctions (as in Bitcoin) to hybrid models like Ethereum's EIP-1559. Roughgarden's [28] analysis of EIP-1559 showed that it steers the system toward market-clearing prices under demand, but also that no mechanism can be both fully strategy-proof and resistant to collusion or bribery. Chung and Shi [12] sharpened this into a formal impossibility: no non-trivial TFM can simultaneously be incentive-compatible for users and for the block producer while resisting producer–user collusion. These results define a benchmark for later designs: TFMs must trade off between efficiency, incentive-compatibility, and resistance to strategic censorship or bribery.

Recent research has begun to extend TFM design to multi-proposer settings. One line of work, such as FOCIL (Fork-Choice Enforced Inclusion Lists), uses committees of validators to construct transaction inclusion lists before block building [36]. However, how to fairly allocate fees among multiple proposers remained an open question. Stouka et al. [32] introduce a game-theoretic model for multi-proposer blockchains (motivated by Ethereum's FOCIL design). They extend Roughgarden's canonical model by adding stages where several proposers each choose which transactions to include and by introducing a bribing adversary that can pay proposers to censor a transaction. Using this framework, they design and evaluate concrete multi-proposer TFMs (e.g. "Double" and "Single" TFMs) that maintain the Bayesian incentive-compatibility guarantees of EIP-1559 while greatly raising the cost of censorship. In effect, Stouka et al. show that appropriate fee-splitting rules can preserve honest behavior and enforce inclusion even under bribery. Notably, their adversary can condition bribes on successful censorship, which is generally cheaper for the adversary than the unconditional bribes we analyze; extending our equilibrium computation to conditional bribes is a natural direction for future work (see Section 7).

Fox et al. [17] and others quantify the "cost of censorship" in economic terms and demonstrate that multi-proposer TFMs can raise this cost by distributing responsibility for inclusion. Some mechanisms reward proposers based on the number of independent inclusions of a transaction, effectively discouraging selective omission [17, 32]. At the application layer, Alpos et al. [3] design censorship-resistant sealed-bid auctions, complementing the auction-driven motivation of [17].

Bribery attacks and censorship in practice. Bribery attacks on consensus have a long history: Bonneau [8] observed that adversaries can *rent* rather than buy consensus power, McCorry et al. [25] constructed smart contracts that trustlessly bribe miners, Winzer et al. [40] analyzed temporary censorship attacks against rational miners, Sun et al. [33] give a practical quantification of bribery attacks, and Karakostas et al. [23] study the efficacy of counterincentives. Closest to our notion of economic censorship, Berger et al. [7] study bribery attacks against optimistic-rollup fraud proofs. Their setting considers repeated censorship over many sequential blocks, whereas we study the cost of censoring a transaction within a single block under rational proposer behavior. The two works are complementary: their analysis assumes a per-block censorship cost, while our eCR metric quantifies exactly that cost. Nor is censorship hypothetical: beyond the measurements of Wahrstätter et al. [38], empirical studies of Ethereum's proposer–builder separation document widespread filtering by builders and relays [22], heavy concentration of the builder market [42], and the centralizing role of private order flow [39] – the latter directly motivating the private-mempool extension we discuss in Section 7.

In parallel, new multi-leader consensus protocols have been proposed to boost throughput and resilience. Building on DAG mempools and consensus [14, 30, 18], DAG-based or "leaderless" designs (such as those used in Avalanche [5], Sui's Mysticeti [6], or Aptos [4]) allow every validator to propose blocks simultaneously, vastly improving throughput compared to single-leader chains. Garimidi, Neu, and Resnick [21] make the general case for multiple concurrent proposers and design a protocol achieving selective-censorship resistance together with transaction hiding. Garimidi et al. [20] formalize fee mechanisms for such leaderless protocols: they define a "strongly BPIC" equilibrium concept and propose the FPA-EQ mechanism, which splits first-price auction revenue equally among all contributing blocks. They prove FPA-EQ is strongly incentive-compatible and guarantees at least a 63.2% fraction of the optimal welfare at equilibrium. These works (and others on Avalanche and Narwhal) highlight that multiple concurrent proposals require fundamentally different TFM designs, but can achieve higher performance and improved censorship resistance if fees are shared appropriately.

Collectively, these works highlight a growing interest in mechanisms that jointly optimize censorship resistance, incentive compatibility, and throughput in decentralized settings. Our work contributes to this space by modeling bribery and duplication in MCP systems and analyzing the equilibrium effects of different TFMs.

Comparison to inclusion lists. Inclusion lists are a powerful tool for censorship resistance. They originate from a line of Ethereum research on censorship under proposer–builder separation [9], refined through forward inclusion lists [13], EIP-7547 [27], the "no free lunch" design [10], and committee-enforced inclusion sets [35], culminating in FOCIL [36], now drafted as EIP-7805 [34]. Their main strength lies in their strictness; proposers sacrifice all of their reward for censoring transactions on the inclusion list. This is also a weakness. Proposers in certain jurisdictions may be legally required to censor certain transactions. By preventing this, inclusion lists may result in more geographic centralization.

A related recent proposal is AUCIL (Auction-based Inclusion Lists) [37], which is designed for committee-based inclusion. In AUCIL, a committee of proposers each compiles an input list of transactions where the number of overlaps is based on the fee paid by the transaction, and then an on-chain auction algorithm merges these lists into a final committed inclusion set. The key idea is that by having each committee member assigned certain transactions (using a correlated equilibrium based algorithm), the protocol ensures that high-paying

transactions appear on at least one list. An adversary wishing to censor a transaction must therefore bribe all proposers who listed it, which is prohibitively expensive. In practice, AUCIL's auction step selects as many of the committee input lists as possible to form the block, thereby "authentically" binding each proposer to its list and making censorship very costly. This mechanism complements the multi-proposer TFMs above by enforcing that committed transactions are actually included, further strengthening censorship resistance in multi-proposer or proposer-builder systems. As a point of comparison to our design, AUCIL employs the even-split TFM, one of the admissible TFMs analyzed in this work.

In contrast, MCP presents a weaker, but more flexible, tool for censorship resistance. Proposers can still censor transactions if legally necessary, but that makes such transactions even more lucrative for proposers who can include it, helping to avoid geographic centralization. Also, MCP is desired for other reasons; namely, it can reduce MEV by limiting the ability of proposers to control the exact ordering of the final block. In that sense, our work analyzes the "free" censorship resistance that comes along with MCP; beyond the consensus protocol and TFM, no additional mechanism is required.

3 Model and Definitions

We now explain our model. There are n proposers who collectively build a block of size B. Each proposer submits a sub-block of at most $\ell := B/n$ transactions (we assume for simplicity that n divides B). These transactions are chosen from a *shared mempool*. We denote this by the sorted list $\{t_1, t_2, \ldots, t_m\}$, where t_i refers to the tip of transaction i, and $t_1 \geq t_2 \geq \cdots \geq t_m$. We sometimes abuse notation to use t_i to refer to the tip or the transaction itself, when it is clear from the context.

The union of the proposers' sub-blocks form the final list of executed transactions. The TFM then portions out the proposers' payments as a function of the full list of sub-blocks. We consider the following three TFMs in this paper:

1. the *duplication-penalizing* TFM. If a transaction t_i is included by only one proposer, then the proposer who included it is paid the full t_i. If a transaction is included by more than one proposer, then the transaction is not charged, and no tip is given to the proposers.
2. the *even-split* TFM. If a transaction t_i is included by k different proposers, then it is charged t_i, and every proposer that includes t_i is paid t_i/k.
3. the *collective* TFM. All transactions are charged their tips, which are pooled and shared equally among *all* proposers.

Further, we assume that the TFM can specify a *base fee* β that is burnt. So, each included transaction pays a flat fee β as well as its payment from the auction.

▶ Remark 1 (User payments and revenue). Our analysis takes tips as exogenous and studies proposer incentives; we do not model users' bidding behavior or user–proposer collusion. Two features of the duplication-penalizing TFM deserve emphasis in this light. First, duplicated transactions are executed free of charge, so this TFM forgoes fee revenue relative to the even-split TFM, which charges every included transaction; our eCR metric reflects this through lower expected user payments. Second, a user able to coordinate with even a single proposer could deliberately trigger duplication to avoid paying. The two-tier tips of Fox et al. [17], of which our duplication-penalizing TFM is the extreme case, mitigate both effects by charging a small tip upon duplicated inclusion; such intermediate TFMs are admissible in the sense of Definition 8, so our equilibrium framework applies to them verbatim. We analyze the extreme case for clarity, and leave a user-side equilibrium analysis to future work.

We assume the presence of an adversary who tries to censor a single transaction, t_c. The adversary offers an unconditional bribe of b to all proposers who do not include t_c. This bribe is common knowledge; all proposers know that all other proposers have been offered the same bribe.

The utility of a proposer is the sum of their payments, both from transactions and possibly from the bribe. Each proposer's strategy space is all subsets of size B/n from the mempool. Our solution concept is a symmetric, mixed Nash equilibrium, which specifies the probability of sampling *each subset* of transactions. But for our purposes, the only relevant feature of the equilibrium is its *induced probabilities* over the transactions, rather than the full distribution over all subsets. We thus represent the equilibrium as $\mathbf{p} = (p_1, \ldots, p_m)$, where p_i is the probability that the proposer includes transaction i in equilibrium.

For the following definitions, we fix the number of proposers, n, the total block size, B, the mempool, $\{tx_1, \ldots, tx_m\}$, and the TFM, $\mathbf{T}$, in which each proposer that includes a transaction tx_i receives $\mathbf{T}(t_i, n_i)$, where t_i is the transaction's tip and n_i is the number of *other* proposers that also include it (so a sole includer receives $\mathbf{T}(t_i, 0)$). In certain cases, even without including the transaction, the proposer can gain a certain value. This induces a unique equilibrium profile, $\mathbf{p}$, as we will show later.

▶ **Definition 2** (fixed-mempool throughput). *The throughput of the current mempool is the expected number of unique transactions in the final block (given that all proposers play* $\mathbf{p}$*), divided by* $\min(B, m)$.

So, the maximum throughput occurs when all transactions are included, or, if that is not possible, when the block is full with no duplicate transactions.

Finally, we turn to economic censorship resistance, defining it first on a per-transaction basis, assuming a fixed mempool state.

▶ **Definition 3** (transaction cost). *The transaction cost of tx_i is its expected payment* conditional on inclusion *in the final block: the flat base fee β plus the expected auction payment of tx_i, in the equilibrium induced by the mempool state and the TFM, conditioned on the event that tx_i appears in the final block.*

This is the quantity that a user who ends up included actually pays; transactions that are excluded pay nothing, and normalizing by the conditional payment makes the eCR of transactions with different inclusion probabilities comparable. To define the cost of bribery, we first assume that the adversary's goal is for t_c to appear in the final block with probability at most $1 - s$. Thus, s is the adversary's desired rejection probability. For ease of exposition, we typically assume that $s = 0.9$, but we later show how our simulation results vary with s. We also rely on the fact that, for any bribe b, there is a unique equilibrium profile $\mathbf{p}^b$, which we show later.

▶ **Definition 4** (bribery cost). *Suppose an adversary wishes for t_c to be censored with probability s, and let b^* be the minimum per-proposer bribe such that, under the induced equilibrium profile $\mathbf{p}^{b^*}$, the probability that t_c is included in the final block is at most $1 - s$. The bribery cost of transaction c is the adversary's* expected total expenditure, $n\,(1 - p_c^{b^*})\,b^*$*: the bribe is offered to each of the n proposers, and each collects it (by excluding t_c) with probability $1 - p_c^{b^*}$.*

▶ **Definition 5** (per-tx eCR). *The economic censorship resistance of transaction t_i is its bribery cost divided by its transaction cost.*

The intuition behind our definition is that transactions which cost more are likely from users with higher valuations, and should thus be harder to censor. Put differently, a bribery cost of 10 and a transaction cost of 1 represents strong censorship resistance; the adversary must spend 10 times the user. If the transaction cost is itself 10, this is no longer as resistant to censorship.

Since all of these definitions depend on the state of the mempool, we add a distributional assumption. So suppose all t_i's are drawn iid from distribution F.

▶ **Definition 6** (throughput). *The expected throughput under F is the expectation, over mempools $\mathbf{t} \sim F^m$, of the throughput of $\mathbf{t}$.*

▶ **Definition 7** (eCR). *The expected censorship resistance under F is the expectation, over mempools $\mathbf{t} \sim F^m$, of the average economic censorship resistance of a transaction $t_i \in \mathbf{t}$.*

4 A Generalized Equilibrium Algorithm

In this section, we do not consider a fixed TFM or bribe. Instead, we provide an algorithm that solves the symmetric mixed Nash equilibrium, given *payoff functions*, that describe how the utility of a proposer changes as they include a transaction, given the other proposers' probability of including that transaction. Our algorithm works for a range of admissible payoff functions, that include all three TFMs we consider, in the absence or presence of any bribe. Intuitively, an admissible payoff function is one where the utility of including transaction i decreases as the probability that others include that transaction increases. In the simplest case of the duplication-penalizing TFM, this holds as the utility of including a transaction is only nonzero if no one else includes it.

4.1 Symmetric Equilibrium via a Linear Program

We characterize symmetric mixed-strategy equilibria of the proposer game as optima of a linear program, and use linear-programming duality to derive a relation between the fee paid by transactions and the probability of its selection. This relation, that all transactions assigned interior probability share a common marginal payoff λ^* is then used to prove a symmetric ($\epsilon-$approximate) equilibrium in Algorithm 1.

Setup. Consider a set of transactions $tx_1, \ldots, tx_m$, each with a single tip value t_i. We seek a symmetric mixed strategy: a probability vector $\vec{p} = (p_1, \ldots, p_m) \in [0,1]^m$ used by every proposer, where p_i is the probability of including tx_i. If, in addition to the proposer under consideration, n_i *other* proposers include tx_i, the proposer's payment for it is $\mathbf{T}(t_i, n_i)$, following the convention of Section 3. We assume throughout that $m > \ell$; otherwise the block trivially fits the entire mempool.

For each transaction tx_i, let $\overline{\lambda}_i$ denote the maximum possible payoff to a proposer (usually when it is the sole includer of tx_i), and $\underline{\lambda}_i$ the minimum possible payoff (usually when all proposers include tx_i). Fix a profile of beliefs $\vec{p}^* \in [0,1]^m$ representing what every *other* proposer will do.

From a proposer's perspective, the expected marginal payoff of including tx_i is a function of the probability that other proposers select this transaction, and t_i. Let u_i be the marginal utility for the proposer (utility for including the transaction minus utility for not including the transaction) when including the transaction tx_i. If every *other* proposer includes tx_i independently with probability p_i, then

$$u_i = \sum_{n_i=0}^{n-1} \binom{n-1}{n_i} p_i^{n_i} (1-p_i)^{n-1-n_i} \mathbf{T}(t_i, n_i) \tag{1}$$

where, with slight abuse of notation, $\mathbf{T}(t_i, n_i)$ in (1) denotes the *marginal* payoff – the difference in a proposer's payment between including and not including tx_i – given that n_i others include it. For the duplication-penalizing and even-split TFMs this coincides with the payment itself; for the collective TFM it is the change in the proposer's pool share (Example 9).

Best response as a linear program. *Treating $\vec{p}^*$ (and hence each u_i) as fixed*, a deviator's best response maximizes total expected payoff subject to a block-size budget ℓ:

$$\begin{aligned} \max_{\vec{p}} \quad & \sum_{i=1}^{m} u_i\, p_i && (2)\\ \text{s.t.} \quad & \sum_{i=1}^{m} p_i = \ell, \\ & 0 \le p_i \le 1 && \forall\, i \in \{1, \dots, m\}. \end{aligned}$$

A symmetric equilibrium then corresponds to a fixed point at which $\vec{p} = \vec{p}^*$ is itself optimal for (2) with the marginal payoffs u_i induced by $\vec{p}^*$ via (1). (We impose the block-size budget with equality; since payments are non-negative, only a bribed transaction can have negative marginal utility, and as $m > \ell$ there are always at least ℓ transactions with non-negative marginal utility to fill the sub-block, so this is without loss of generality.)

▶ **Property 1** (Equilibrium structure). *There exists an optimal solution $\vec{p}^*$ to (2) and a scalar $\lambda^* \in \mathbb{R}$ such that, for every $i \in \{1, \dots, m\}$:*
1. *if $u_i > \lambda^*$, then $p_i^* = 1$;*
2. *if $u_i < \lambda^*$, then $p_i^* = 0$;*
3. *if $0 < p_i^* < 1$, then $u_i = \lambda^*$.*

Conversely, any feasible $\vec{p}$ for which some λ satisfies conditions 1–3 is optimal for (2).

Proof sketch. The claim follows from linear-programming duality: introducing a dual variable $a_i \ge 0$ for each capacity constraint $p_i \le 1$ and λ for the clearing constraint, complementary slackness forces $p_i^* = 1$ when $u_i > \lambda^*$, $p_i^* = 0$ when $u_i < \lambda^*$, and $u_i = \lambda^*$ on the interior; the converse direction verifies that any profile satisfying conditions 1–3 admits a dual-feasible certificate. The full proof appears in the full version [29]. ◀

Property 1 reduces the problem of finding a symmetric equilibrium to identifying the threshold λ^* and the corresponding inclusion probabilities. The next subsection presents an efficient binary-search algorithm for this task.

4.2 A Binary-Search Algorithm for Symmetric Equilibria

Building on Property 1, we now give an efficient algorithm to compute symmetric equilibria. The idea is to invert the marginal-payoff relationship (1): at a symmetric fixed point $\vec{p} = \vec{p}^*$, equation (1) couples each p_i to u_i, and Property 1 couples each interior p_i to a common threshold λ^*. Composing these, the equilibrium probability of each transaction is determined by λ^* alone, reducing the search to a one-dimensional bisection over λ.

Setup. For each transaction tx_i, let $f_i(\lambda)$ denote the symmetric inclusion probability that makes a proposer indifferent at marginal payoff λ: the value of p_i for which (1) yields $u_i = \lambda$ when every other proposer also plays p_i. Concretely, for the marginal-payoff form

of Example 9, f_i admits a closed form; in general it can be computed by inverting (1) numerically. By construction, f_i takes the value 1 at $\lambda = \underline{\lambda}_i$ (full competition) and 0 at $\lambda = \overline{\lambda}_i$ (monopoly inclusion), matching the bounds defined in Section 4.1.

▶ **Definition 8** (Admissible payoff function). *A function $f_i\colon \mathbb{R} \to \mathbb{R}$ is* admissible *with parameters $(\underline{\lambda}_i, \overline{\lambda}_i)$ if:*

- ***(A1) Boundary values:*** *$f_i(\underline{\lambda}_i) = 1$ and $f_i(\overline{\lambda}_i) = 0$.*
- ***(A2) Continuity and strict monotonicity:*** *f_i is continuous and strictly decreasing on $[\underline{\lambda}_i, \overline{\lambda}_i]$.*
- ***(A3) Range:*** *$f_i(\lambda) > 1$ for $\lambda < \underline{\lambda}_i$ and $f_i(\lambda) < 0$ for $\lambda > \overline{\lambda}_i$.*

▶ **Example 9** (Collective TFM). Suppose all fees from included transactions are pooled and divided equally among the n proposers, regardless of who included them. Proposer j's total payoff is

$$U_j \;=\; \frac{1}{n}\sum_{i=1}^{m} T_i \cdot \mathbf{1}\big[tx_i \text{ included by at least one proposer}\big].$$

Because j receives the same share of T_i whether or not she personally includes tx_i (provided someone does), the *marginal* value of inclusion equals the expected pool contribution conditional on every other proposer omitting tx_i:

$$\Delta u_i(p_i) \;=\; \frac{T_i}{n}\,(1-p_i)^{n-1}.$$

Setting $\Delta u_i(p_i) = \lambda$ and inverting yields

$$f_i(\lambda) \;=\; 1 - \left(\frac{n\lambda}{T_i}\right)^{\frac{1}{n-1}}, \qquad \underline{\lambda}_i = 0, \qquad \overline{\lambda}_i = \frac{T_i}{n}.$$

Verification of admissibility. **(A1)** $f_i(0) = 1$ and $f_i(T_i/n) = 0$. **(A2)** The map $\lambda \mapsto (n\lambda/T_i)^{1/(n-1)}$ is strictly increasing for $\lambda \geq 0$, so f_i is strictly decreasing on $[0, T_i/n]$. **(A3)** For $\lambda > T_i/n$ the inner term exceeds 1, so $f_i(\lambda) < 0$; the natural extension to $\lambda < 0$ gives $f_i(\lambda) > 1$.

▶ **Example 10** (Duplication-penalizing TFM). Under this TFM a transaction pays its tip T_i to its includer only when it is included exactly once, so the marginal value of including tx_i is nonzero only in the event that no other proposer includes it:

$$u_i(p_i) \;=\; (1-p_i)^{n-1}\,T_i, \qquad f_i(\lambda) \;=\; 1 - \left(\frac{\lambda}{T_i}\right)^{\frac{1}{n-1}}, \qquad \underline{\lambda}_i = 0, \qquad \overline{\lambda}_i = T_i.$$

This is the same functional form as Example 9 with T_i in place of T_i/n, and admissibility is verified identically.

▶ **Example 11** (Even-split TFM). Suppose a transaction included by k proposers is charged its full tip T_i, which is split evenly, so that each includer receives T_i/k. If every other proposer includes tx_i independently with probability p_i, the number of *other* includers is $K \sim \mathrm{Bin}(n-1, p_i)$, and the marginal value of inclusion is

$$u_i(p_i) \;=\; T_i \cdot \mathbb{E}\left[\frac{1}{1+K}\right] \;=\; T_i \sum_{k=0}^{n-1} \binom{n-1}{k} \frac{p_i^k (1-p_i)^{n-1-k}}{k+1} \;=\; T_i\,\frac{1-(1-p_i)^n}{n\,p_i},$$

where the last equality uses $\binom{n-1}{k}\frac{1}{k+1} = \frac{1}{n}\binom{n}{k+1}$ and the binomial theorem, and $u_i(0) := T_i$ by continuity.

u_i is strictly decreasing with $u_i(0) = T_i$ and $u_i(1) = T_i/n$, so its inverse f_i is admissible with parameters $(\underline{\lambda}_i, \overline{\lambda}_i) = (T_i/n, T_i)$; the verification of (A1)–(A3) appears in the full version [29].

Unlike the collective TFM, u_i here admits no closed-form inverse. This poses no obstacle: Algorithm 1 only requires *pointwise evaluation* of f_i, and since u_i is continuous and strictly decreasing, $f_i(\lambda)$ can be evaluated to any accuracy ε' by an inner bisection on $[0, 1]$, adding a factor $O(\log(1/\varepsilon'))$ to the per-evaluation cost in Theorem 14.

A tractable approximation. Replacing $\mathbb{E}[1/(1+K)]$ by $1/(1+\mathbb{E}[K])$ yields a closed-form admissible approximation of u_i that understates the value of inclusion (by Jensen's inequality) and shares its boundary values; we use it only for intuition, and all reported numerical results are computed with the exact u_i. Details appear in the full version [29].

The clamped aggregate. For any candidate $\lambda \in \mathbb{R}$, define the *clamped total inclusion mass*

$$G(\lambda) = \sum_{i=1}^{m} \min\big(\max(f_i(\lambda),\, 0),\, 1\big).$$

By admissibility, the ith term equals 1 for $\lambda \le \underline{\lambda}_i$, equals 0 for $\lambda \ge \overline{\lambda}_i$, and is strictly decreasing in between. Hence G is non-increasing on $\mathbb{R}$, with $G(\min_i \underline{\lambda}_i) = m$ and $G(\max_i \overline{\lambda}_i) = 0$. Combining Property 1 with the definition of f_i, finding a symmetric equilibrium reduces to locating λ^* with $G(\lambda^*) = \ell$, where ℓ is the target block size. Since G is monotone, a one-dimensional bisection suffices.

▶ Remark 12 (Marginals suffice). Two observations justify working with inclusion probabilities rather than full distributions over sub-blocks. First, since proposers randomize independently and a proposer's payoff is additive over transactions, with the term for tx_i depending on the others only through the number of them that include tx_i (distributed $\mathrm{Bin}(n-1, p_i)$), opponents' strategies affect a proposer's payoff only through their marginals; hence the best-response problem over distributions on ℓ-subsets reduces to the linear program (2) over marginals. Second, every feasible point of (2) is realizable: for any $\vec{p} \in [0,1]^m$ with $\sum_i p_i = \ell \in \mathbb{N}$ there is a random ℓ-subset S with $\Pr[i \in S] = p_i$. (This follows from a standard systematic-sampling construction; see the full version [29].) Together, these imply that optimal marginal vectors correspond exactly to best responses in the original strategy space.

▶ **Theorem 13** (Existence and uniqueness). *For any set of transactions $tx_1, \ldots, tx_m$ with admissible payoff functions $f_1, \ldots, f_m$ and any target block size $\ell \in (0, m)$, there exists λ^* with $G(\lambda^*) = \ell$, and $\mathbf{p}^* = (\hat{f}_1(\lambda^*), \ldots, \hat{f}_m(\lambda^*))$ is a symmetric mixed Nash equilibrium. Moreover, the equilibrium profile is unique: any two thresholds with clamped inclusion mass ℓ induce the same probability vector $\mathbf{p}^*$.*

Proof sketch. *Existence:* G is continuous and non-increasing with $G(\min_i \underline{\lambda}_i) = m$ and $G(\max_i \overline{\lambda}_i) = 0$, so the intermediate value theorem yields λ^* with $G(\lambda^*) = \ell$, and the induced profile satisfies the conditions of Property 1 (realizable over ℓ-subsets by Remark 12). *Uniqueness:* any two thresholds with clamped mass ℓ induce identical probability vectors, since each $\hat{f}_i$ is non-increasing and the sums agree. The full proof appears in the full version [29]. ◀

▶ **Theorem 14** (Termination and runtime). *Let ω_G be a modulus of continuity of G on $[\min_i \underline{\lambda}_i, \max_i \overline{\lambda}_i]$ (one exists, since G is continuous on this compact interval), and fix any $\delta_\varepsilon > 0$ with $\omega_G(\delta_\varepsilon) \le \varepsilon$. Then Algorithm 1 terminates after at most $\lceil \log_2(\Delta/\delta_\varepsilon) \rceil$ iterations, where $\Delta = \max_i \overline{\lambda}_i - \min_i \underline{\lambda}_i$. Each iteration evaluates G once at cost of $O(m)$ evaluations of the f_i, giving total runtime $O\big(m \log(\Delta/\delta_\varepsilon)\big)$.*

Algorithm 1 Symmetric-equilibrium binary search.

Require: Admissible functions $f_1, \dots, f_m$ with parameters $(\underline{\lambda}_i, \overline{\lambda}_i)$; block size ℓ; tolerance $\varepsilon > 0$.

Ensure: Probabilities $(p_1^*, \dots, p_m^*)$ forming an ε-approximate symmetric equilibrium.

$G(\lambda)$ denotes the clamped total inclusion mass defined above.

function BinarySearch($f_1, \dots, f_m,\ \ell,\ \varepsilon$)
  if $m \le \ell$ **then return** $(1, 1, \dots, 1)$
  if $\ell = 0$ **then return** $(0, 0, \dots, 0)$
  $\lambda_{\min} \leftarrow \min_i \underline{\lambda}_i$; $\lambda_{\max} \leftarrow \max_i \overline{\lambda}_i$ ▷ $G(\lambda_{\min}) = m \ge \ell$, $G(\lambda_{\max}) = 0 \le \ell$
  loop
    $\lambda_{\text{mid}} \leftarrow (\lambda_{\min} + \lambda_{\max})/2$
    if $|G(\lambda_{\text{mid}}) - \ell| \le \varepsilon$ **then**
      break ▷ *inclusion mass is within tolerance of ℓ*
    else if $G(\lambda_{\text{mid}}) > \ell$ **then**
      $\lambda_{\min} \leftarrow \lambda_{\text{mid}}$ ▷ *too many inclusions; raise λ*
    else
      $\lambda_{\max} \leftarrow \lambda_{\text{mid}}$ ▷ *too few inclusions; lower λ*
  $\lambda^* \leftarrow \lambda_{\text{mid}}$
  for $i = 1, \dots, m$ **do**
    $p_i^* \leftarrow \min\big(\max(f_i(\lambda^*), 0), 1\big)$
  return $(p_1^*, \dots, p_m^*)$

Proof. By Theorem 13 there exists λ^* with $G(\lambda^*) = \ell$, and the update rule maintains the invariant $\lambda_{\min} \le \lambda^* \le \lambda_{\max}$ (verified in the proof of Theorem 16). The bracket width halves each iteration: after k iterations it equals $\Delta/2^k$. Once $\Delta/2^k \le \delta_\varepsilon$, the midpoint satisfies $|\lambda_{\text{mid}} - \lambda^*| \le \delta_\varepsilon$, hence $|G(\lambda_{\text{mid}}) - \ell| = |G(\lambda_{\text{mid}}) - G(\lambda^*)| \le \omega_G(\delta_\varepsilon) \le \varepsilon$, and the termination condition is met. Each iteration computes G once, inspecting all m transactions; for TFMs whose f_i admits no closed form (e.g., the even-split TFM of Example 11), each evaluation additionally costs an inner bisection, multiplying the runtime by $O(\log(1/\varepsilon'))$ for inner accuracy ε'. ◀

With existence and uniqueness established, we can now prove that Algorithm 1 outputs an approximate equilibrium.

▶ **Definition 15** (ε-approximate equilibrium). *A probability vector $(p_1, \dots, p_m) \in [0,1]^m$ is an* ε-approximate symmetric mixed Nash equilibrium *if there exists $\hat{\lambda} \in \mathbb{R}$ such that*

1. *the equilibrium conditions of Property 1 hold at $\hat{\lambda}$ with probabilities p_i, and*
2. $\left|\sum_{i=1}^m p_i - \ell\right| \le \varepsilon$.

▶ **Theorem 16** (Correctness). *The output $(p_1^*, \dots, p_m^*)$ of Algorithm 1 is an ε-approximate symmetric mixed Nash equilibrium, where ε is the tolerance supplied to the algorithm.*

Proof sketch. By Theorem 13 there is a λ^* with $G(\lambda^*) = \ell$. The bisection invariant brackets λ^*, and after the stated number of iterations the interval is short enough that, by the modulus of continuity of G and monotonicity of each $\hat{f}_i$, the returned probabilities are ε-approximate in the stated sense. The full proof appears in the full version [29]. ◀

5 Deriving Economic Censorship Resistance

In this section, we describe a general "cut-and-paste" method to derive the bribe required to censor a target transaction with a given probability. Our method only requires computing the equilibrium probabilities for a *single* game. Using these probabilities, we can derive a closed-form expression for the bribe required, and by extension, for the economic censorship resistance (per Definition 5) of the target transaction. We then apply the cut-and-paste method to all three TFMs we study in this paper.

5.1 Cut-and-Paste Method

Setup. We recall some notation from earlier. Let $u_i(p_i)$ be the expected (increase in) utility of including tx_i assuming everyone else includes it with probability p_i. Given transaction tips $\mathbf{t} = (t_i)_{i \in m}$ and target probability mass ℓ, let $(\mathbf{p} = (p_i)_{i \in [m]}, \lambda) = \textsc{equilibrium}(\mathbf{t}, \ell)$ be the equilibrium probabilities and common utility level, respectively. This means that $u_i(p_i) = \lambda$ for all $i \in [m]$. Also let $f_i(x)$ be the admissible function (per Definition 8) such that $p_i = f_i(\lambda)$.

Suppose the adversary $\mathcal{A}$ wishes to censor a target transaction tx_c (with tip t_c) for some $c \in [m]$. We are interested in determining the bribe amount b^* that $\mathcal{A}$ needs to offer to all parties, so that tx_c is censored with probability $s \in (0, 1]$. In order for $\mathcal{A}$ to achieve the target censorship probability s, each party must include tx_c with probability p_c^* such that $s = (1 - p_c^*)^n$, or equivalently, $p_c^* = 1 - s^{1/n}$. This holds regardless of the underlying TFM.

Cut-and-Paste Method. The method proceeds in two steps.

First, the *cut* step: solve for the equilibrium of the game with tx_c removed, reserving exactly p_c^* units of capacity to be filled in later. Let $(\mathbf{p}_{-c}, \lambda_{-c}) := \textsc{equilibrium}(\mathbf{t}_{-c}, \ell - p_c^*)$ denote this equilibrium.

Second, the *paste* step: add tx_c back, with the adversary's bribe b^* chosen so that, in the resulting equilibrium of the full game $G(\mathbf{t}, \ell)$, (1) tx_c is included with the target probability p_c^*; and (2) every $tx_j \neq tx_c$ retains the probability p_j it had in $\mathbf{p}_{-c}$. We carry out this analysis in Section 5.2 and obtain the closed-form bribe $b^* = u_c(p_c^*) - \lambda_{-c}$.

5.2 Correctness of the Cut-and-Paste Method

We prove that the cut-and-paste method produces an equilibrium of $G(\mathbf{t}, \ell)$ under bribe b^* in which tx_c attains probability p_c^* and every $tx_j \neq tx_c$ retains its probability from the post-cut equilibrium $\mathbf{p}_{-c}$.

The argument relies on a structural property of equilibria captured by the following lemma: a new transaction can be added to the game without disturbing the existing inclusion probabilities or utility level, provided capacity is expanded by exactly the amount the new transaction consumes at level λ. The paste step will exploit this to add tx_c back into the post-cut equilibrium without re-solving the full system.

▶ **Lemma 17** (Marginal extension of equilibrium). *Let $\mathbf{p} = \{p_1, \ldots, p_m\}$ be a symmetric Nash equilibrium of the game $G(\mathbf{t}, \ell)$ at utility level λ, where $\mathbf{t} = \{tx_i\}_{i \in [m]}$. Let tx_{m+1} be a new transaction with admissible function f_{m+1} of parameters $\underline{\lambda}_{m+1}, \overline{\lambda}_{m+1}$, and let $\hat{f}_{m+1}$ be its associated clamped function. Define $\mathbf{t}' := \mathbf{t} \cup \{tx_{m+1}\}$ and $\ell' := \ell + \hat{f}_{m+1}(\lambda)$. Then $\mathbf{p}' := \mathbf{p} \cup \{\hat{f}_{m+1}(\lambda)\}$ is the unique symmetric mixed Nash equilibrium of $G(\mathbf{t}', \ell')$, supported at the same utility level λ. In particular, p_i is unchanged for every $i \in [m]$, and tx_{m+1} attains $p_{m+1} = \hat{f}_{m+1}(\lambda)$.*

Proof sketch. One verifies that appending $\hat{f}_{m+1}(\lambda)$ to the original profile satisfies all equilibrium conditions of $G(\mathbf{t}', \ell')$ at the same level λ: the per-transaction identities are local and unaffected by enlarging the transaction set, and the clearing condition holds by construction of the enlarged budget. Details appear in the full version [29]. ◀

▶ **Theorem 18** (Correctness of cut-and-paste)**.** *Fix a target probability $p_c^* \in (0,1)$ and let $(\mathbf{p}_{-c}, \lambda_{-c}) = \textsc{equilibrium}(\mathbf{t}_{-c},\, \ell - p_c^*)$ be the post-cut equilibrium. Under the per-proposer bribe*

$$\boxed{b^* \;=\; u_c(p_c^*) \;-\; \lambda_{-c}}$$

the game $G(\mathbf{t}, \ell)$ has a symmetric equilibrium in which tx_c is included with probability p_c^* and every $tx_j \neq tx_c$ retains its probability from $\mathbf{p}_{-c}$. The boundary case $p_c^* = 0$ (full censorship, $s = 1$) is handled by continuity, giving $b^* = \overline{\lambda}_c - \lambda_{-c}$.

Proof sketch. A bribe b translates the demand of tx_c: the shifted inverse satisfies the *bribe-shift identity* $f_c'(\lambda, b) = f_c(\lambda + b)$, and horizontal translation preserves admissibility. Applying Lemma 17 to the post-cut equilibrium with tx_c (equipped with the shifted demand) as the new transaction, the augmented equilibrium coincides with that of $G(\mathbf{t}, \ell)$ at target probability p_c^* precisely when $\hat{f}_c'(\lambda_{-c}, b) = p_c^*$; by the bribe-shift identity this reads $\hat{f}_c(\lambda_{-c} + b) = p_c^*$, whose solution is $b^* = u_c(p_c^*) - \lambda_{-c}$. The full derivation appears in the full version [29]. ◀

Finally, recall from Definition 4 that the bribery cost is the adversary's *expected total expenditure*: the per-proposer bribe b^* is collected by each proposer with probability $1 - p_c^* = s^{1/n}$, so the bribery cost equals $n\, s^{1/n}\, b^*$. All eCR values we report (Section 6) use this total.

5.3 Applying Cut-and-Paste to Concrete TFMs

We now apply the cut-and-paste recipe to all three TFMs.

Duplication-Penalizing TFM. For this TFM, recall from Example 10 that

$$u_i(p_i) = (1 - p_i)^{n-1} \cdot t_i \qquad f_i(\lambda) = 1 - \left(\frac{\lambda}{t_i}\right)^{\frac{1}{n-1}}.$$

Since f_i is admissible, substituting into $b = u_i(p_i) - \lambda$ gives $b = s^{\frac{n-1}{n}} \cdot t_i - \lambda$.

Even-Split TFM. For this TFM, recall from Example 11 that

$$u_i(p_i) = t_i \cdot \frac{1 - (1 - p_i)^n}{n\, p_i},$$

whose inverse f_i is admissible with parameters $(\underline{\lambda}_i, \overline{\lambda}_i) = (t_i/n,\, t_i)$, though it has no closed form. The bribe, however, does: substituting $p_c^* = 1 - s^{1/n}$ into $b = u_c(p_c^*) - \lambda_{-c}$ and using $(1 - p_c^*)^n = s$ gives $b = \frac{t_c\,(1-s)}{n\,(1-s^{1/n})} - \lambda_{-c}$. As a sanity check, as $n \to \infty$ we have $n(1 - s^{1/n}) \to \ln(1/s)$, so the per-proposer bribe approaches $t_c(1-s)/\ln(1/s) - \lambda_{-c}$: even with many proposers, each proposer's marginal stake in tx_c remains a constant fraction of t_c, unlike the collective TFM below where it vanishes as $1/n$.

Collective TFM. For this TFM, recall from Example 9 that

$$u_i(p_i) = (1-p_i)^{n-1} \cdot \frac{t_i}{n} \quad \text{and} \quad f_i(\lambda) = 1 - \left(\frac{\lambda}{\frac{t_i}{n}}\right)^{\frac{1}{n-1}}.$$

Since f_i is admissible, substituting into $b = u_i(p_i) - \lambda$ gives $b = s^{\frac{n-1}{n}} \cdot \frac{t_i}{n} - \lambda$.

Note that this reduces to the duplication-penalizing TFM, with just a $1/n$ factor applied to all tips. This reflects the fact that the mechanisms are game-theoretically quite similar. For both mechanisms, the only factor relevant to the validator's utility is whether anyone else has included the transaction. If so, including it in their sub-block doesn't increase their utility; if not, their utility increases by t_i or t_i/n.

6 Comparing TFMs on Idealized and Real-World Data

In this section, we run Monte-Carlo simulations using Algorithm 1 and the cut-and-paste recipe to plot the approximate throughput and eCR as the number of proposers increases. We show how these values vary with congestion, burn, and the adversary's desired censorship probability. We compare across the three TFMs, finding that the duplication-penalizing TFM performs best across the board.

6.1 Comparing TFMs

We first describe our baseline parameters and simulation method. We simulate a mempool of 250 transactions for a block that can hold any 200 of these. The bids of the transactions are drawn iid from $\textsc{Pareto}(1, 2.5)$, to simulate the standard environment of many low bidders and a few high bidders. We perform 1000 trials, each with different mempools, and for each trial, we use our "cut-and-paste" method to compute its economic censorship resistance and throughput, assuming that an adversary wants to achieve a 90% censorship probability. We graph the average of the trials as the number of validators, n, varies.

Figure 1 compares the three TFMs using the baseline parameters, showing that the duplication-penalizing TFM achieves the highest eCR, and ties the collective TFM for the highest throughput. Recall from Section 5 that the *collective* TFM reduces to the duplication-penalizing TFM with all tips scaled by $1/n$; since a uniform rescaling of the tips merely rescales λ^*, the two TFMs induce identical equilibrium probabilities, and thus identical throughput. They differ starkly in eCR, however: under the collective TFM each proposer's marginal stake in any single transaction is only t_i/n, so the adversary's bribes shrink as n grows, while users continue to pay their full tips. This is why the collective TFM's eCR remains near zero regardless of n.

The even-split TFM has lower throughput than the other two: including an already-included transaction still pays t_i/k, so duplication is penalized only weakly, and the equilibrium concentrates more probability mass on high-tip transactions. Its eCR sits between the other two TFMs. Compared to the duplication-penalizing TFM, the gap is driven by the price to users: the even-split TFM always charges users their tip, while the duplication-penalizing TFM only does so if their transaction is singly-included. As a result, the duplication-penalizing TFM has significantly higher eCR, as the bribes required are much larger relative to the expected user payment.

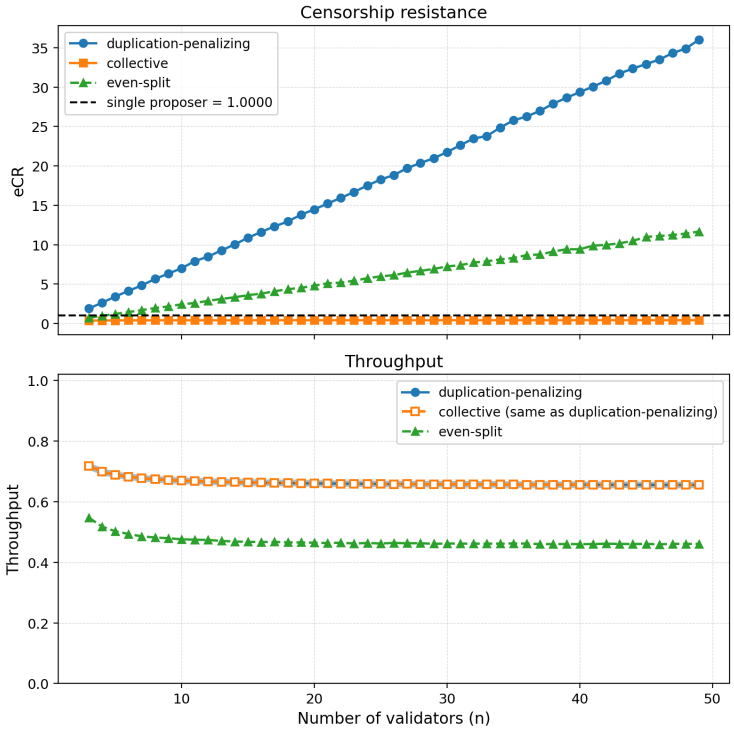


Figure 1 Comparison of three TFMs, with the baseline parameters of $m = 250$, $B = 200$, $s = 0.9$, and bids drawn iid from $\textsc{Pareto}(1, 2.5)$.

Throughput. Note that the throughput quickly converges in both cases. For context, a single proposer trivially achieves throughput 1 by deduplicating its own block; moving to MCP thus incurs a one-time duplication cost (roughly 0.65 at our baseline under the duplication-penalizing TFM). The key observation is that this cost does not compound: it stabilizes almost immediately, so increasing n further buys censorship resistance at essentially no additional throughput cost. To contextualize the throughput, we compare these algorithms to an optimistic baseline. Suppose that the validators had the explicit goal of maximizing throughput, but were not able to coordinate. It's easy to see that the optimal strategy would be for all validators to choose each transaction uniformly at random. Now, assuming that $m \geq B$, i.e., there are more transactions than block space (so that $\min(B, m) = B$ in Definition 2), the throughput can be written as:

$$\frac{1}{B}\sum_{i \leq m} 1 - (1 - p_i)^n \approx \frac{1}{B}\sum_{i \leq m} 1 - e^{-np_i}$$

The approximation holds well as long as p_i is small, which happens when n is moderate, as that decreases the sub-block size. Let $y_i = np_i$, the expected number of copies of transaction §i. Under the uniform distribution, each transaction is chosen with $p_i = \ell/m$, where $\ell = B/n$ is the sub-block size. Thus, $y_i = np_i = B/m$, and the throughput of the uniform distribution is approximately $\frac{m}{B}(1 - e^{-B/m})$. This represents an *upper bound* on throughput – profit incentives will decrease the actual throughput, as validators are more likely to include more valuable transactions.

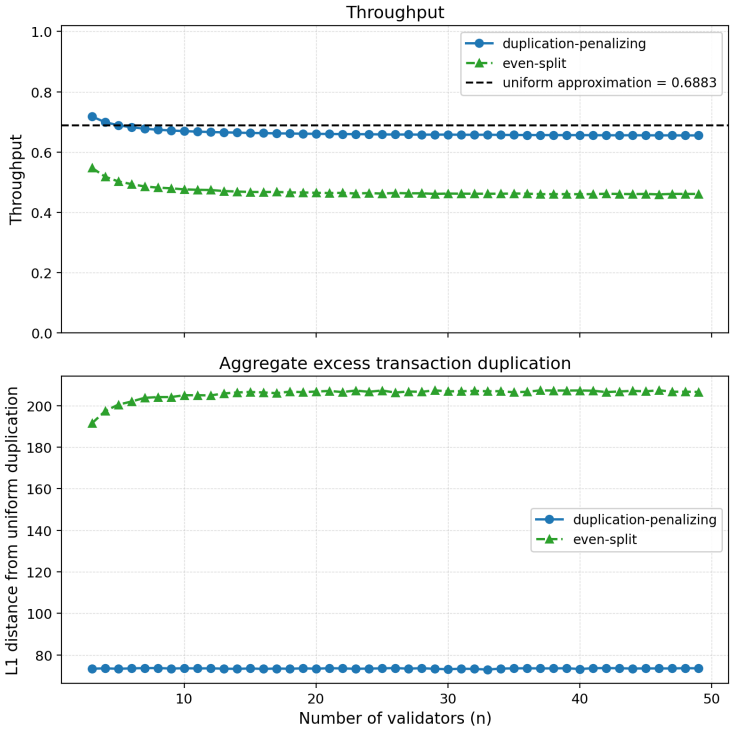


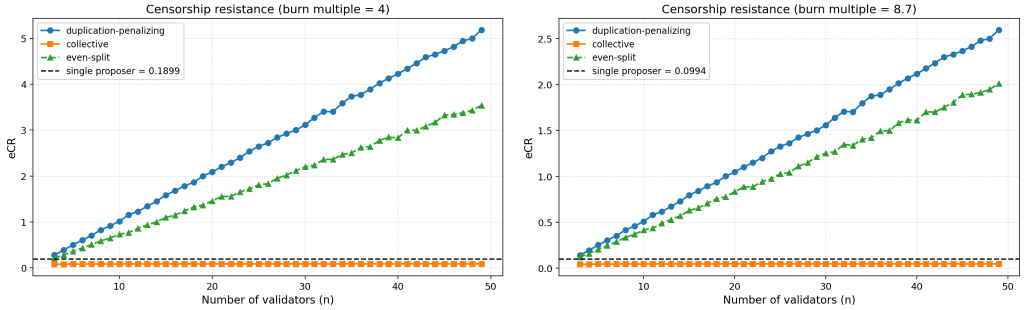
Figure 2 Comparing throughput to the optimistic upper bound of a uniform distribution. We use the baseline parameters of $m = 250$, $B = 200$, $s = 0.9$, and bids drawn iid from $\textsc{Pareto}(1, 2.5)$.

In Figure 2, we plot the throughput of the duplication-penalizing and even-split TFMs, as well as their "gap from uniform," in terms of their $\sum_{i \leq m} |y_i - B/m|$. This represents their aggregate excess duplication, where "excess" is defined relative to the uniform distribution. The main takeaway is that the duplication-penalizing TFM maintains a throughput very close to the optimistic upper-bound of a uniform distribution.

Burn. So far, we have been assuming no burn; users pay only the payment from the TFM. Burn hurts eCR significantly, since higher burn means that adversaries can bribe proposers for much less than the users pay. To determine a reasonable amount of burn, we use data from Ethereum in January 2026, and find that the base fee was 8.7x the average tip. Figure 3 plots the eCR of the three TFMs assuming a burn multiple of 4 and 8.7. All users are charged the same amount of burn.

Though burn significantly hampers the eCR, we see that eCR still increases roughly linearly with n, and so increasing the number of validators can still result in high eCR. For example, with a burn multiple of 8.7 and 49 validators, the eCR is approximately 2.5, so an adversary would need to pay 2.5x more than the user to censor the average transaction. The figure also plots the single-proposer baseline, which suffers greatly as burn increases. At a burn multiple of 8.7, the single-proposer eCR is approximately 0.09, so an adversary can censor the average transaction by paying only 9% of what the user pays.

Further, note that increasing the burn brings the duplication-penalizing and even-split TFMs closer together. With no burn and 49 validators, the eCR was approximately 35 and 10, for the duplication-penalizing and even-split TFMs respectively. At a burn multiple of 8.7, the eCR was approximately 2.5 and 1.8. This is because the burn is a flat fee charged to all transactions, and so the differences in user payments under the two TFMs are diluted.



(a) A burn multiple of 4. **(b)** A burn multiple of 8.7.

Figure 3 Comparing eCR assuming that the burn is either 4x or 8.7x the average tip. We use the baseline parameters of $m = 250, B = 200, s = 0.9$, and the bids are drawn iid from $\text{PARETO}(1, 2.5)$.

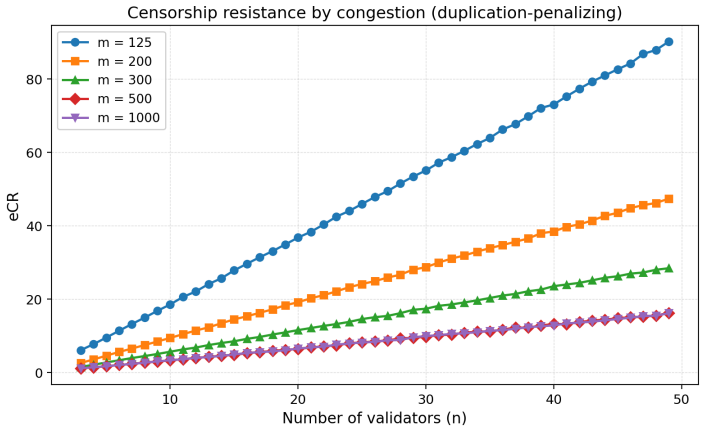


Figure 4 Comparing eCR as congestion varies, using the baseline parameters of $B = 200, s = 0.9$, and bids drawn iid from $\text{PARETO}(1, 2.5)$. The number of transactions m varies in the different curves.

6.2 Congestion and Adversary Incentives

Here, we evaluate the impact of congestion and the adversary's goal on the censorship resistance. We only consider the duplication-penalizing TFM, as it beats the other two in both eCR and throughput.

For congestion, with the baseline parameters, we have 250 transactions with a block size of 200. In Figure 4, we plot the eCR for both more congested and less congested cases. We fix the block size at 200, and vary the number of transactions. As congestion increases, the eCR goes down, because the opportunity cost of censoring a transaction decreases (it's more likely that there is another good transaction to take its place).

In Figure 5, we plot the eCR for a range of adversary censorship probabilities. Our baseline assumes that the adversary wants to censor a transaction with 90% probability; we plot for both more lenient and more strict adversaries. As the adversary's desired censorship probability decreases, the eCR also decreases, as smaller bribes are required.

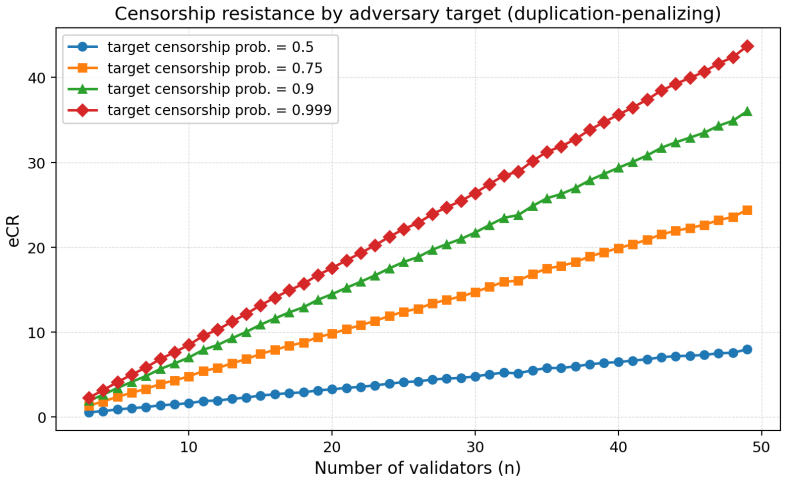


Figure 5 Comparing eCR as the adversary's censorship probability varies, using the baseline parameters of $B = 200, m = 250$, and bids drawn iid from $\textsc{Pareto}(1, 2.5)$. The adversary's censorship probability varies in the different curves.

6.3 Analysis on Empirical Data

We next perform the comparison on real Ethereum transaction data. We use a window of 10,000 recent Ethereum blocks, covering block numbers 24,341,338 through 24,351,341. For each block $\mathcal{B}$, we compute each transaction's tip as `gasPrice` − `baseFeePerGas` and keep only transactions with strictly positive tip, since these are the transactions that are afforded any eCR. This yields a block-specific set of valid transactions, whose size we denote by $m_{\mathcal{B}}$.

For each block $\mathcal{B}$, we fix the target total block size at $B_{\mathcal{B}} = 0.8\, m_{\mathcal{B}}$, so that the protocol aims to include roughly 80% of the positive-tip transactions in that block. We then vary the number of validators n from 2 to 50, and for each value of n set the per-validator probability-mass budget to $\ell_{\mathcal{B}} = B_{\mathcal{B}}/n$, so that total capacity $n\ell_{\mathcal{B}}$ remains fixed as n changes. We run this procedure for the three TFMs in our study: duplication-penalizing, collective, and even-split.

Given the positive-tip transactions of a block $\mathcal{B}$ and a choice of n, we first solve for the baseline single-proposer equilibrium probabilities $(p_i)_i$ under the relevant TFM. From these probabilities we compute throughput by taking the expected number of distinct transactions included by at least one of the n validators and normalizing by the target block size. To measure economic censorship resistance, we take a sample of 20 target transactions per block. For each sampled target transaction tx_i, we use the cut-and-paste method to compute the per-validator bribe needed to achieve a target *total* censorship probability of 90%. This gives a post-bribe target inclusion probability p_i' and a per-censor bribe b_i, from which we compute

$$\mathsf{eCR}_i = \frac{n(1 - p_i')\, b_i}{P_i},$$

where P_i is the baseline expected user payment conditional on inclusion, in accordance with Definitions 4 and 5.

Finally, for each block $\mathcal{B}$ and each n, we average eCR_i over the sampled target transactions in that block, and separately record the corresponding throughput. We then average these block-level quantities over the full 10,000-block dataset to obtain the curves shown in Figure 6.

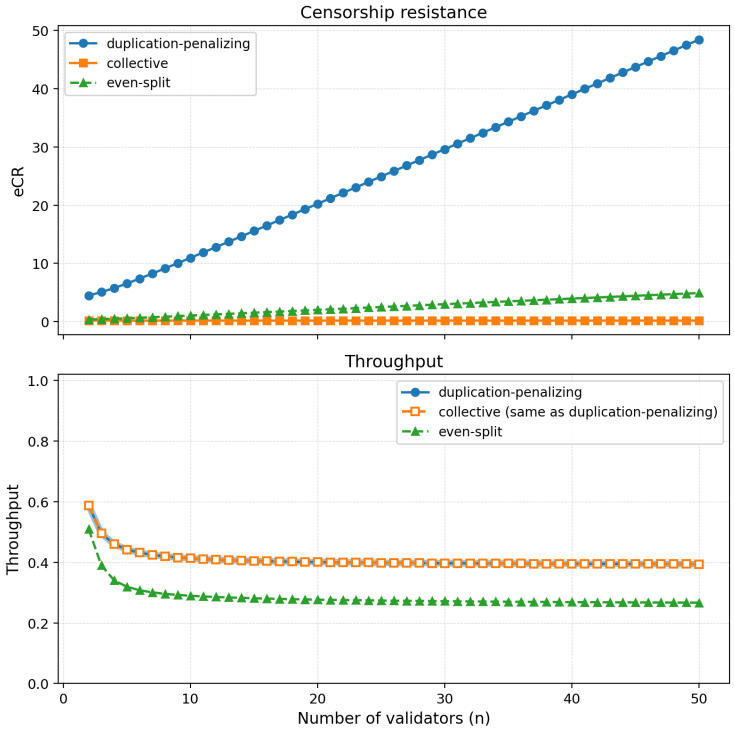


Figure 6 Comparison of three TFMs on recent Ethereum block data, with $B = 0.8m$, $s = 0.9$, and bids given by observed positive transaction tips.

One caveat is in order: this dataset conditions on inclusion. Transactions that were successfully censored never appear on-chain, so the empirical tip distribution may understate the mass of low-tip or contested transactions. We therefore view these experiments as evaluating the TFMs on realistic bid shapes, rather than as a measurement of censorship itself.

Takeaways from empirical analysis. Similar to the analysis on the synthetic data, the analysis on the empirical data shows that the duplication-penalizing TFM dominates the other two in both throughput and economic censorship resistance. The main difference between the empirical data and our synthetic data is that the empirical tip distribution is more skewed; a larger percentage of the tips are closer to 0 in reality. This reduces the throughput, because the probability mass concentrates on the few high value tips. Lower throughput means more duplications, and so the duplication-penalizing TFM has much higher censorship resistance, since the average user payment is low. The link to reproduce any empirical result can be found at zenodo record number 21831640.

7 Conclusion

We proposed a novel definition of economic censorship resistance (eCR), and modeled the incentives of rational proposers in the presence of a censoring adversary. We found an efficient algorithm to determine the equilibrium of our game, and our algorithm works for a variety of TFMs. We used this algorithm to compare the eCR and throughput of three TFMs on real and synthetic data. Overall, we found that:

1. MCP systems enjoy much higher eCR than single proposer systems
2. Burnt user fees can significantly lower eCR
3. eCR scales linearly with the number of validators, and thus strong eCR can be achieved even in the presence of significant burn
4. Of the TFMs we study, the duplication-penalizing TFM dominates in both throughput and eCR, across a wide range of parameters

We suggest three avenues for future work. The first is to analyze a wider range of TFMs. In particular, one may be able to prove that the duplication-penalizing TFM is optimal among a family of TFMs, or to prove some theoretical upper bounds on censorship, as we informally did for throughput. A user-side equilibrium analysis – endogenous tips, and robustness to user–proposer collusion under the duplication-penalizing TFM (Remark 1) – is another natural target. This would complement our empirical analysis, and provide guidance on how to best implement an MCP system.

Second, one could analyze different bribery attacks. We consider a simple setting, where the censoring-adversary must declare a transaction to be censored, along with a bribe for any proposer who doesn't include that transaction. But censoring behavior may take other forms. For example, the bribe could be paid out only if *no* proposer includes that transaction.

Finally, we assume that all proposers have access to a shared mempool. But some proposers may have private access to some transactions. The simplest example of this is private order flow, but this may also occur for geographical reasons. For example, proposers located far from the rest of the network may receive transactions from their region before the rest of the network, and thus have semi-private access in some timing windows. Our model, and the duplication-penalizing TFM in particular, heavily rewards proposers with private transactions. Thus, MCP systems may serve as a counterbalance to the latency pressures that encourage geographical co-location.

References

1 Ittai Abraham, Yuval Efron, and Ling Ren. The latency cost of censorship resistance. Cryptology ePrint Archive, Paper 2025/2136, 2025. URL: https://eprint.iacr.org/2025/2136.

2 Orestis Alpos, Bernardo David, Nikolas Kamarinakis, and Dionysis Zindros. An overview of efficiency and censorship resistance in consensus protocols. *CoRR*, abs/2504.03588, 2025. arXiv:2504.03588.

3 Orestis Alpos, Lioba Heimbach, Kartik Nayak, and Sarisht Wadhwa. Censorship-resistant sealed-bid auctions on blockchains. Cryptology ePrint Archive, Paper 2025/2127, 2025. URL: https://eprint.iacr.org/2025/2127.

4 Aptos Foundation. Aptos White Paper. https://aptos.dev/network/blockchain/aptos-white-paper, 2025. Accessed 2025-11.

5 Ava Labs. Avalanche consensus protocol. https://docs.avax.network/overview/avalanche-consensus, 2020.

6 Kushal Babel, Andrey Chursin, George Danezis, Lefteris Kokoris-Kogias, and Alberto Sonnino. Mysticeti: Low-Latency DAG Consensus with Fast Commit Path, February 2024. arXiv:2310.14821 [cs]. doi:10.48550/arXiv.2310.14821.

7 Ben Berger, Edward W. Felten, Akaki Mamageishvili, and Benny Sudakov. Economic censorship games in fraud proofs. In *Proceedings of the 26th ACM Conference on Economics and Computation*, EC '25, pages 670–687, New York, NY, USA, 2025. Association for Computing Machinery. doi:10.1145/3736252.3742611.

8 Joseph Bonneau. Why buy when you can rent? Bribery attacks on Bitcoin-style consensus. In *Financial Cryptography and Data Security – FC 2016 International Workshops (BITCOIN)*, volume 9604 of *Lecture Notes in Computer Science*, pages 19–26. Springer, 2016. doi:10.1007/978-3-662-53357-4_2.

9 Vitalik Buterin. State of research: Increasing censorship resistance of transactions under proposer/builder separation (PBS). https://notes.ethereum.org/s3JToeApTx6CKLJt8AbhFQ, 2021. Ethereum research notes.

10 Vitalik Buterin and Mike Neuder. No free lunch – a new inclusion list design. https://ethresear.ch/t/no-free-lunch-a-new-inclusion-list-design/16389, 2023. Ethereum Research post.

11 Censorship.pics. https://censorship.pics, 2025. Accessed 2025.

12 Hao Chung and Elaine Shi. Foundations of transaction fee mechanism design. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3856–3899. SIAM, 2023. doi:10.1137/1.9781611977554.ch150.

13 Francesco D'Amato. Forward inclusion lists. https://notes.ethereum.org/@fradamt/forward-inclusion-lists, 2022. Ethereum research notes.

14 George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. Narwhal and Tusk: A DAG-based mempool and efficient BFT consensus. In *Proceedings of the Seventeenth European Conference on Computer Systems (EuroSys '22)*, pages 34–50. ACM, 2022. doi:10.1145/3492321.3519594.

15 Electronic Frontier Foundation. Financial censorship. https://www.eff.org/issues/financial-censorship, 2024.

16 Fatima Elsheimy, Ioannis Kaklamanis, Sarisht Wadhwa, Charalampos Papamanthou, and Fan Zhang. Censorship resistance vs throughput in multi-proposer BFT protocols. Cryptology ePrint Archive, Paper 2026/126, 2026. URL: https://eprint.iacr.org/2026/126.

17 Elijah Fox, Mallesh M. Pai, and Max Resnick. Censorship Resistance in On-Chain Auctions. In Joseph Bonneau and S. Matthew Weinberg, editors, *5th Conference on Advances in Financial Technologies (AFT 2023)*, volume 282 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 19:1–19:20, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.AFT.2023.19.

18 Fangyu Gai, Jianyu Niu, Ivan Beschastnikh, Chen Feng, and Sheng Wang. Scaling blockchain consensus via a robust shared mempool (stratus). In *Proc. of IEEE ICDE 2023*, pages 530–543. IEEE, 2023. doi:10.1109/ICDE55515.2023.00047.

19 Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology – EUROCRYPT 2015*, volume 9057 of *Lecture Notes in Computer Science*, pages 281–310. Springer, 2015. doi:10.1007/978-3-662-46803-6_10.

20 Pranav Garimidi, Lioba Heimbach, and Tim Roughgarden. Transaction fee mechanism design for leaderless blockchain protocols, 2025. doi:10.48550/arXiv.2505.17885.

21 Pranav Garimidi, Joachim Neu, and Max Resnick. Multiple concurrent proposers: Why and how. Cryptology ePrint Archive, Paper 2025/1772, 2025. URL: https://eprint.iacr.org/2025/1772.

22 Lioba Heimbach, Lucianna Kiffer, Christof Ferreira Torres, and Roger Wattenhofer. Ethereum's proposer-builder separation: Promises and realities. In *Proceedings of the 2023 ACM Internet Measurement Conference (IMC '23)*, pages 406–420. Association for Computing Machinery, 2023. doi:10.1145/3618257.3624824.

23 Dimitris Karakostas, Aggelos Kiayias, and Thomas Zacharias. Blockchain bribing attacks and the efficacy of counterincentives. In *Proc. of CCS 2024*, pages 1031–1045, 2024. doi:10.1145/3658644.3670330.

24 Mysten Labs. Sui: A platform for high-performance smart contracts. https://github.com/MystenLabs/sui/blob/main/doc/paper/sui.pdf, 2022. Accessed 2025-11.

25 Patrick McCorry, Alex Hicks, and Sarah Meiklejohn. Smart contracts for bribing miners. In Aviv Zohar, Yael Kalai, and David McGrew, editors, *Financial Cryptography 2018 Workshops*, volume 10958 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 2019. doi:10.1007/978-3-662-58820-8_1.

26 Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of bft protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 31–42, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2976749.2978399.

27 Mike Neuder, Vitalik Buterin, Francesco D'Amato, Terence Tsao, Potuz, and Manav. EIP-7547: Inclusion lists. https://eips.ethereum.org/EIPS/eip-7547, 2023. Ethereum Improvement Proposal (draft).

28 Tim Roughgarden. Transaction fee mechanism design. *CoRR*, abs/2106.01340, 2021. arXiv:2106.01340.

29 Aditya Saraf, Ioannis Kaklamanis, Sarisht Wadhwa, and Fatima Elsheimy. Censorship resistance and throughput with multiple concurrent proposers, 2026. doi:10.48550/arXiv.2607.16995.

30 Alexander Spiegelman, Neil Giridharan, Alberto Sonnino, and Lefteris Kokoris-Kogias. Bullshark: DAG BFT Protocols Made Practical. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, pages 2705–2718, New York, NY, USA, November 2022. Association for Computing Machinery. doi:10.1145/3548606.3559361.

31 Chrysoula Stathakopoulou, Tudor David, Matej Pavlovic, and Marko Vukolić. Mir-bft: High-throughput robust bft for decentralized networks, 2021. arXiv:1906.05552.

32 Aikaterini-Panagiota Stouka, Julian Ma, and Thomas Thiery. Multiple proposer transaction fee mechanism design: Robust incentives against censorship and bribery, 2025. arXiv:2505.13751v2 [cs.GT]. doi:10.48550/arXiv.2505.13751.

33 Hanyi Sun, Na Ruan, and Chunhua Su. How to model the bribery attack: A practical quantification method in blockchain. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *Computer Security – ESORICS 2020, Proceedings, Part II*, volume 12309 of *Lecture Notes in Computer Science*, pages 569–589. Springer, 2020. doi:10.1007/978-3-030-59013-0_28.

34 Thomas Thiery, Francesco D'Amato, et al. EIP-7805: Fork-choice enforced inclusion lists (FOCIL). https://eips.ethereum.org/EIPS/eip-7805, 2024. Ethereum Improvement Proposal (draft).

35 Thomas Thiery, Francesco D'Amato, and Barnabé Monnot. The more, the less censored: Committee-enforced inclusion sets (COMIS) on Ethereum. https://ethresear.ch/t/18835, 2024. Ethereum Research post.

36 Thomas Thiery, Barnabe Monnot, Luca Zanolini, and Julian Ma. Fork-choice enforced inclusion lists (focil). https://ethresear.ch/t/19870, 2024. Ethereum Research post.

37 Sarisht Wadhwa, Julian Ma, Thomas Thiery, Barnabe Monnot, Luca Zanolini, Fan Zhang, and Kartik Nayak. Aucil: An inclusion list design for rational parties. *Cryptology ePrint Archive*, 2025.

38 Anton Wahrstätter, Jens Ernstberger, Aviv Yaish, Liyi Zhou, Kaihua Qin, Taro Tsuchiya, Sebastian Steinhorst, Davor Svetinovic, Nicolas Christin, Mikolaj Barczentewicz, and Arthur Gervais. Blockchain censorship. In *Proceedings of the 33rd ACM Web Conference (WWW '24)*, pages 1632–1643, Singapore, 2024. Association for Computing Machinery. doi:10.1145/3589334.3645431.

39 Shuzheng Wang, Yue Huang, Wenqin Zhang, Yuming Huang, Xuechao Wang, and Jing Tang. Private order flows and builder bidding dynamics: The road to monopoly in Ethereum's block building market. In *Proceedings of the ACM Web Conference 2025 (WWW '25)*. Association for Computing Machinery, 2025. doi:10.1145/3696410.3714754.

40 Fredrik Winzer, Benjamin Herd, and Sebastian Faust. Temporary censorship attacks in the presence of rational miners. In *IEEE Europe. Symposium on Security and Privacy Workshops (EuroSP Workshops) 2019*, pages 357–366. IEEE, 2019. doi:10.1109/EuroSPW.2019.00046.

41 Bowen Xue, Soubhik Deb, and Sreeram Kannan. Bigdipper: A hyperscale bft system with short term censorship resistance, 2023. doi:10.48550/arXiv.2307.10185.

42 Sen Yang, Kartik Nayak, and Fan Zhang. Decentralization of Ethereum's builder market. In *Proceedings of the 46th IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2025. doi:10.48550/arXiv.2405.01329.

43 Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan Gueta, and Ittai Abraham. Hotstuff: Bft consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (PODC '19)*, PODC '19, pages 347–356, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3293611.3331591.

The Execution Dilemma in Pessimistic Blockchains: Profitability XOR Fair Ordering

Artjom Pugatsov
Delft University of Technology, The Netherlands

Can Umut Ileri
IOTA Foundation, Berlin, Germany
Delft University of Technology, The Netherlands

Jérémie Decouchant
Delft University of Technology, The Netherlands

Abstract

The successive generations of consensus algorithms progressively displaced the performance bottleneck of blockchains to the execution layer. Recent works address the execution performance bottleneck by parallelizing the execution of non-conflicting transactions. Historically, execution closely followed the consensus-level transaction ordering determined by validators, a practice highly susceptible to Maximal Extractable Value (MEV) exploitation. Conversely, recent academic proposals introduce rigid fair-ordering protocols that are bound to severely restrict transaction reordering at the execution layer. Parallel execution frameworks optimize the sequencing layer, which lies between consensus and execution and assembles transaction batches from committed consensus blocks and transmits them to execution workers to maximize both execution parallelism and realized transaction fees. To achieve this optimization, current sequencing implementations may defer transactions. Importantly, these implementations do not maintain order-fairness properties. To the best of our knowledge, preserving these properties currently requires sequential execution, which would drastically reduce both profitability and performance.

In this work, we address the tension between validator profit and order fairness using a dynamic optimization framework. We introduce a blockchain-agnostic model for transaction sequencing in a continuous setting where block sequencing and execution run concurrently. Consequently, when sequencing cannot be completed within the available time window, our framework dynamically returns its best intermediate result. Within this framework, we propose an anytime genetic algorithm that utilizes gas prices, object sets, and predicted execution times to optimize schedules. We also augment this algorithm to optionally maintain fair-ordering. We evaluate our approach with real-world datasets from Sui and Ethereum, and demonstrate that it increases validator profit by approximately 15% and accelerates congestion relief speed by up to 58%. Furthermore, we quantify the impact of fair-ordering constraints, showing that they can reduce validator profit by 50% to 60% during periods of high congestion. We provide the first evidence that enforcing strict fair ordering might effectively nullify the advantages of advanced sequencing.

2012 ACM Subject Classification Theory of computation → Online algorithms; Applied computing → Electronic commerce

Keywords and phrases Blockchain, Execution layer, Sequencing, Byzantine Fault Tolerance

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.27

Related Version *Full Version*: `https://arxiv.org/abs/2604.23266` [50]

Supplementary Material
Software (Source Code): `https://github.com/Artjom-Pugatsov/genetic-sequencing`



© Artjom Pugatsov, Can Umut Ileri, and Jérémie Decouchant;
licensed under Creative Commons License CC-BY 4.0



1 Introduction

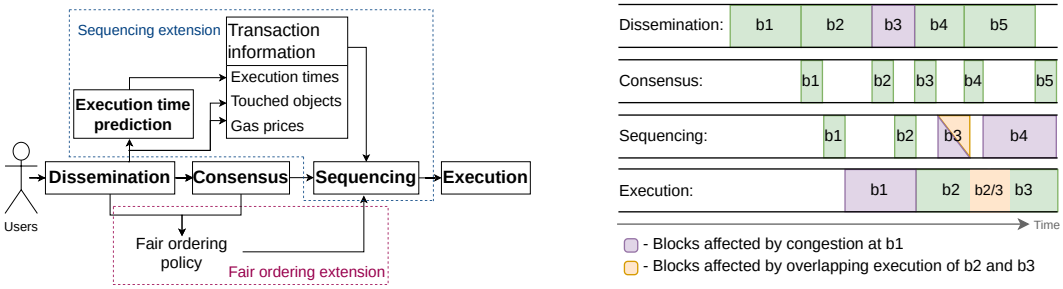
Historically, blockchain systems have faced significant performance limitations. For example, Bitcoin and Ethereum support only about 7 and 40 transactions per second (TPS), respectively [40, 12]. As blockchain applications have expanded beyond payments to domains such as decentralized finance and the Internet of Things, ensuring substantially higher throughput has become critical. Early blockchains used monolithic architectures that tightly coupled transaction dissemination, consensus, verification, and execution, making it difficult to optimize individual components. This limitation led to lazy blockchain architectures [14, 25, 56], which separate these functions into distinct stages that can be optimized independently and executed concurrently. As a result, system performance is bounded by the slowest stage. While research has mostly focused on consensus [29], recent approaches, e.g., high-throughput Directed Acyclic Graph (DAG)-based protocols [35, 14, 5, 48], have shifted the main performance bottleneck to the execution layer.

Execution performance can be optimized in two ways. Vertical optimization improves the execution of individual transactions, often by analyzing smart contract languages [26, 2], but its effectiveness depends on transaction type. We focus on horizontal optimization, which increases throughput by executing independent transactions in parallel across multiple executors [39, 3, 52, 8, 18, 15, 27]. Using horizontal optimization, state-of-the-art execution engines reportedly achieve up to 170 kTPS [27]. The key property that enables efficient horizontal parallelization is transaction independence. Independent transactions can be executed concurrently across distributed workers, necessitating only a final state merge. In the object model [20, 6, 22], transactions consist of read and write operations on specific state objects and two transactions are therefore considered independent if their operations on common objects are read-only.

Traditional blockchains, such as Bitcoin and Ethereum, allow validators to choose transaction order within a block. This acts as a form of congestion control, as validators prioritize transactions with higher gas fees to mitigate spam and balance throughput with available execution resources [7]. However, this opaque discretion also enables Maximal Extractable Value (MEV), where validators reorder transactions for profit [13]. In response, fair ordering protocols have been proposed for the dissemination and consensus layers [44, 37, 60], enforcing transaction order based on arrival time. These approaches define two extremes: unrestricted validator control and rigid ordering that ignores validator incentives.

In this work, we investigate a middle ground based on the transaction sequencing framework of Sui [24, 23], which uses the object model. In this framework, transactions are either ordered for execution or selectively deferred to be reprocessed with later blocks when resource limits are exceeded. Sui orders transactions by gas price and processes them sequentially, deferring any transaction whose objects would exceed a protocol-defined execution capacity. We extend Sui's approach by incorporating execution-time estimates [17, 49] and object-conflict information [10, 57] directly into the sequencing process. We specify a sequencing framework that supports the continuous execution model, where the processing of consecutive blocks by different blockchain layers might overlap, and changing congestion conditions. To address this specification, we propose an anytime genetic algorithm[1] that continuously refines the ordering as time permits.

[1] An anytime algorithm returns a valid solution even if interrupted before completion.



(a) Lazy blockchain architecture. **(b)** Continuous execution in a lazy blockchain.

Figure 1 Overview of the sequencing layer in a lazy blockchain. Fig. 1a illustrates the modular architecture we consider, where sequencing bridges consensus and execution by computing schedules based on transaction gas prices, object sets, and predicted execution times, with optional fair ordering constraints. Fig. 1b shows the pipelined processing of blocks across blockchain layers, including two cases where the continuous execution requirement affects sequencing. First, congestion detected during the execution of block 1 causes the dissemination of block 3 to end early, freeing time for additional sequencing of upcoming blocks 3 and 4. Second, an overlap between the execution of blocks 2 and 3 forces early termination of block 3's sequencing to maintain execution continuity.

As a summary, we make the following **contributions**:

- We introduce the first formalization of the transaction sequencing problem within a continuous execution framework. Unlike traditional single-shot approaches, our model explicitly accounts for overlapping block boundaries, transaction object conflicts, and dynamic scheduling deadlines under network congestion. We extend this model to formalize how order-fairness policies can be integrated as optional execution constraints.

- We design and implement an anytime genetic algorithm tailored for the sequencing layer to optimize validator profit using gas prices, object sets, and predicted execution times. The algorithm accommodates real-time execution constraints by immediately outputting a high-quality intermediate schedule if interrupted prematurely. We augment our genetic framework to support optional consensus-determined fair ordering. By strictly enforcing arrival-sequence constraints only for interdependent (conflicting) transactions, our algorithm intelligently allows the concurrent reordering of independent transactions.

- We implement our algorithm and compare it against several sequencing heuristics using real-world Sui data and synthetic Ethereum-based workloads. Our experiments quantify the impact of perturbed execution-time estimates. Using real-world blockchain data, our results demonstrate that the proposed genetic sequencer increases validator profit by up to 16% under sustained congestion and accelerates congestion relief by up to 58% during sporadic transaction spikes.

- We provide the first empirical evidence that enforcing strict fair ordering severely limits independent transaction optimization, imposing a 50% to 60% profit penalty on validators during peak congestion. Based on our model, algorithms, and datasets, our results suggest that rigid fair ordering may eliminate the performance benefits of advanced parallel sequencing layers.

2 Background

2.1 Sequencing

Fig. 1a illustrates the modular architecture of a so-called lazy blockchain, where consensus and execution are decoupled. We focus on the sequencing layer, which receives blocks[2] from the consensus layer. The sequencing layer assigns transactions to execution workers, manages congestion, and may defer transactions that cannot be completed within the available execution window. We assume that execution-time estimates are available for all transactions. These estimates can be derived from transaction properties as early as during the dissemination phase, when transactions become known, although the estimation method itself is outside the scope of this work. We also consider an optional fair ordering constraint, which requires the sequencing layer to preserve the causal order of transactions. The sequencing layer produces an estimated schedule, while the execution layer performs the actual execution. Because execution times may vary and conflicts can introduce additional delays, the realized execution order may differ from the estimated schedule.

2.2 Scheduling Estimation

While we focus on optimizing transaction ordering, evaluating a sequencing algorithm requires estimating the resulting execution schedule, which ultimately determines execution performance. We assume a scheduling function that maps an ordered set of transactions to estimated start times and worker assignments. Although our method is independent of its specific implementation, we use a concrete scheduler to evaluate results.

Because optimal scheduling is NP-hard, we assume a deterministic polynomial-time greedy scheduler that processes transactions sequentially, assigning each to the earliest feasible position on a worker and deferring those that cannot be safely placed within a worker's schedule. Our approach extends Sui's to a fixed number of workers, whereas the original design assumed unbounded parallelism and focused solely on object availability. Alg. 1 details our extended scheduling algorithm.

We use the Earliest Start Time policy, which assigns each transaction to the worker that allows the earliest execution and is the default policy in current blockchains. We further improve scheduling by allowing transactions to be inserted into gaps in existing schedules rather than only appended. While this improves utilization [30], it increases complexity since gap structures must be maintained and queried. For efficiency reasons, we store gaps in a B-tree ordered by duration and search from the smallest feasible gap upward, stopping at the first valid placement. This differs slightly from the original gap-filling formulation but achieves comparable schedule quality with better scalability for large workloads. We therefore adopt this greedy gap-filling scheduler as our evaluation scheduling function.

2.3 Sequencing Objective and Requirements

Objective. We model sequencing as a combinatorial optimization problem that constructs a transaction schedule optimizing a given performance objective. For instance, minimizing makespan targets shortest completion time, while maximizing throughput focuses on the number of transactions executed within a fixed horizon. In this work, we aim at maximizing

[2] We use the term "block" to denote a set of transactions sequenced and executed together. In DAG-based architectures, a single commit may include multiple vertices.

Algorithm 1 Greedy scheduler.

input : L_{all}, d, m
output : Assignment A, where $A[tx] = (t, w_{id})$ or None

```
A ← ∅
for tx in L_all do
    WrkFreeAt ← ∅
    for w in 0, 1, ..., m − 1 do                    // find earliest free slot per worker
        WrkFreeAt[w] ← find_earliest(A, w, tx)
    wrk ← choose_worker(WrkFreeAt)                  // choose worker based on policy
    if wrk = None then
        A[tx] ← None                                // no free workers
    else
        A[tx] ← (WrkFreeAt[wrk], wrk)               // schedule at earliest available slot
return A
```

the total fees of scheduled transactions. A transaction's execution fee is calculated by multiplying its execution time by its gas price. This objective prioritizes less resource-intensive transactions while still admitting more congestive ones when their gas price compensates for their cost. It also aligns sequencing with validator incentives by maximizing expected revenue, encouraging continued participation in the network.

Requirements. The sequencing step should satisfy three requirements: high performance, adaptability, and continuous operation. First, since sequencing is meant to improve execution performance, it should not delay it. Second, during low congestion periods, a simple sequencing strategy suffices and spending excessive time on optimization would only increase latency. Differently, in periods of high congestion, a sequencer should dynamically increase its optimization effort and utilize an anytime algorithm to prioritize transaction parallelism, reordering independent transactions while deferring highly conflicting ones. Finally, sequencing must support continuity so execution can proceed to the next block without waiting for the previous one to fully complete. Instead, workers should continuously consume available transactions. As they finish, they immediately take on new work, potentially from later blocks, while earlier transactions may still be executing. This leads to fully overlapping execution across multiple blocks, with no strict barrier between them.

As illustrated in Fig. 1b, these requirements enable cross-layer parallelism. The system can dynamically adjust sequencing effort and timing under congestion, while still maintaining continuous execution and overlapping the processing of successive blocks. Sequencing can be stopped early to immediately feed execution, reducing idle time while allowing later optimization to refine future schedules. We do not assume any specific coordination mechanism between layers. Instead, these requirements are reflected in the design of our ordering algorithm, which must be anytime, and inherently support continuous operation.

2.4 Greedy Ordering Baseline

Sui's current sequencing mechanism uses a heuristic that considers transactions in descending order of their gas prices. Transactions are considered one at a time and either included or deferred based on their estimated impact on shared-object capacity. This greedy approach can perform poorly under contention. A highly congestive transaction may be selected first,

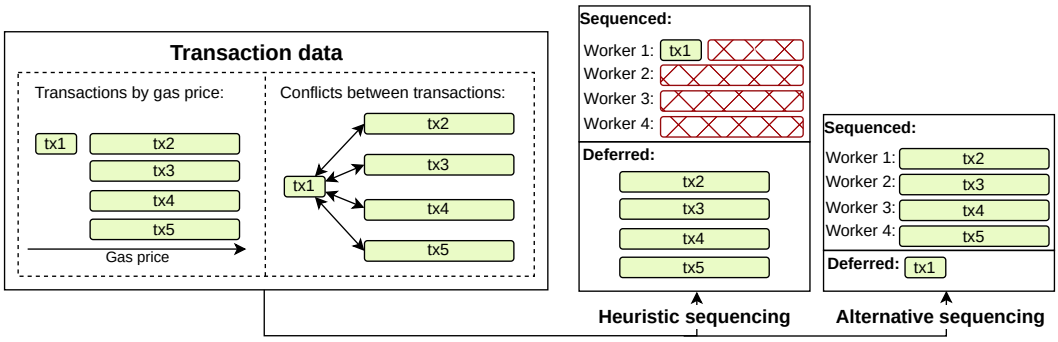


Figure 2 Greedy transaction sequencing based on gas price (left) compared to a feasible alternative (right). The width of each transaction corresponds to its execution time. The greedy algorithm is sub-optimal and defers transactions $tx2$–5, while the alternative algorithm only defers $tx1$.

blocking many smaller transactions that could otherwise execute in parallel. As illustrated in the example of Fig. 2, scheduling transaction 1 first causes transactions 2 to 5 to be deferred, leaving execution workers underutilized. In contrast, deferring transaction 1 and scheduling transactions 2 to 5 increases parallelism, reduces deferrals, and may yield higher gas revenue. Replacing gas price with another single ordering criterion does not resolve this issue, as a highly congestive transaction may still be scheduled first. Instead, we incorporate both execution-time estimates and transaction conflicts into the sequencing process to construct schedules that better utilize available parallelism and improve overall performance.

3 Modeling Continuous Transaction Sequencing

In this section, we present what is, to our knowledge, the first formalization of sequencing within a continuous execution model. We begin by outlining our foundational assumptions, followed by the formalization of both single-shot and multi-shot continuous sequencing. Through these definitions, we explicitly establish the structural constraints that any valid sequencing algorithm, including ours, must satisfy. We introduce our notation throughout the text as needed, and provide a comprehensive summary in Table 1.

3.1 Assumptions

We consider a system of m execution workers on which transactions are scheduled. We assume two conditions: (i) transaction read and write sets are available; and (ii) transaction execution times are known.

The first assumption is directly satisfied in the pessimistic execution model utilized by Sui and Solana, where each transaction must explicitly declare its read and write sets [6, 22]. Even in environments without this structural requirement, static analysis tools can accurately predict or infer these sets beforehand [57, 10]. The second assumption is significantly more challenging to guarantee. Determining a transaction's exact execution path and runtime dynamically is notoriously difficult. Even with pre-execution, the actual processing time can vary based on its final position within a block. Nevertheless, extensive research into historical data analysis and predictive simulation has demonstrated high accuracy in estimating these runtimes [17, 49]. In our experiments we explicitly introduce prediction inaccuracies to evaluate their impact on system performance.

Table 1 Notations.

Symbol	Description
Single-shot sequencing	
d, m	Execution deadline ($d \in \mathbb{R}^+$), Number of workers ($m \in \mathbb{N}^+$)
$T_{\text{all}}, L_{\text{all}}$	Set (T_{all}) and ordered list (L_{all}) of all transactions $\{tx_1, \ldots, tx_n\}$
tx_i	Transaction triplet (O_i, t_i, g_i) with execution time t_i and gas price g_i
O_i	Object accesses tuple (R_i, W_i) where $R_i, W_i \subseteq \mathcal{I}$ are read and write sets
σ	Scheduling function: $(L_{\text{all}}, d, m) \to \Pi_{L_{\text{all}}}$
$\Pi_{L_{\text{all}}}$	Schedule mapping: $tx_i \to \text{Either}\big(\texttt{None}, (s_i, w_i)\big)$ with start time s_i and worker w_i
T_{sched}, P	Scheduled transactions subset ($T_{\text{sched}} \subseteq T_{\text{all}}$) and validator profit ($P \in \mathbb{R}^+$)
Continuous sequencing (at block i)	
$d_i, d_{i_{\text{new}}}$	Base execution deadline and extended deadline
m_i, θ	Number of workers and maximum deferral rounds ($\theta \in \mathbb{N}^+$)
$T_{i_{\text{inc}}}, T_{i_{\text{all}}}, T_{i_{\text{sched}}}$	Incoming, total considered, and scheduled transactions subsets
δ_i, C_i^w	Maximum slack time and latest finish time of worker w
O_{tx_j}	Touched objects set $R_j \cup W_j$ for transaction tx_j
α_k^i, S_i^k	Latest scheduled time for object k, and transactions subset touching object k
Fair ordering	
$L_{\text{fair}}, G_{\text{fair}}$	Total transaction order list, and transaction dependency graph (T_{all}, E)

3.2 Single-Shot Sequencing

Input. Let the execution deadline $d \in \mathbb{R}^+$ be the maximum amount of time a worker can be active for a given block. We consider a system that relies on $m \in \mathbb{N}^+$ workers to execute transactions in parallel. We are given a scheduling function σ that takes as input an ordered list $L_{all} = [tx_1, tx_2, \ldots, tx_n]$ of n transactions and schedules all or part of them on the m workers over a time period d, producing a schedule $\Pi_{L_{all}}$.

A set of all transactions $T_{all} = \{tx_1, \cdots, tx_n\}$ consists of transactions tx_i represented as a tuple (O_i, t_i, g_i). The tuple $O_i = (R_i, W_i)$ of object ids that tx_i accesses consists of a set R_i of object ids that it reads from and a set W_i of object ids that it writes to. Variable $t_i \in \mathbb{R}^+$ is the execution time of tx_i, and $g_i \in \mathbb{R}^+$ is the gas price of a unit of execution time paid by the transaction.

A scheduling function $\sigma : (L_{all}, d, m) \to \Pi_{L_{all}}$ is set in advance and represents a scheduler. It takes L_{all}, d and m as input and produces a schedule $\Pi_{L_{all}} : tx_i \to \text{Either}\big(\texttt{None}, (s_i, w_i)\big)$. The tuple (s_i, w_i) corresponds to the start time s_i and the worker id w_i on which the transaction has been scheduled. Transactions that are mapped to `None` are deferred to the following block.

Constraints. Any schedule $\Pi_{L_{all}}$ produced by σ must be subject to several constraints. First, a scheduled transaction tx_i must be scheduled at a positive time ($s_i \geq 0$) and adhere to the deadline ($s_i + t_i \leq d$). It also must be assigned to a valid worker id $w_i \in \{0, 1, \cdots, m-1\}$. Finally, the execution must be conflict-free: if tx_j is scheduled on the same worker as tx_i ($w_i = w_j$) or if the transactions conflict i.e., $((W_i \cap (R_j \cup W_j)) \neq \emptyset) \vee ((W_j \cap (R_i \cup W_i)) \neq \emptyset)$ then their executions must not overlap ($s_j + t_j \leq s_i \ \vee \ s_i + t_i \leq s_j$).

Objective. Our goal is to find an ordering L_{all} of transactions T_{all}, such that for a given σ we maximize the validator profit $P := \sum_{(O_i,t_i,g_i)\in T_{sched}} t_i \cdot g_i$ where $T_{sched} := \{tx_i \in T_{\text{all}} \mid \Pi_{L_{all}}(tx_i) \neq \text{None}\}$ is the set of scheduled transactions.

3.3 Multi-Shot Continuous Sequencing

Our single-shot sequencing formulation captures the transaction sequencing one block at a time. But as opposed to consensus, which operates in discrete intervals, transaction execution is a continuous process: there is no need to wait for all the transactions of a previous block to be executed before executing non-conflicting transactions from the next block. Our goal in this section is to turn this isolated, single-block problem into one whose parameters are shaped by the outcome of the previous block: the only new input carried over between iterations is the initial busy status of the workers, while every other effect of continuity is captured through changes to the problem's parameters. To allow for transaction execution continuity, we extend our formulation of sequencing in several ways.

We label parameters based on the block number for which we generate the sequence: $T_{i_{\text{inc}}}$, d_i, m_i are respectively the set of transactions, execution deadline and the number of workers at block i. $T_{i_{\text{inc}}}$ denotes the set of new transactions that just came in during the new round. The full set of transactions that will be scheduled in round i is denoted $T_{i_{\text{all}}}$. We also define θ, a parameter which determines the maximum number of rounds for which a transaction can be deferred before it is canceled.

We cancel transactions that have been deferred for too many rounds, and add the rest to the transaction set of the block $T_{i_{\text{all}}} = T_{i_{\text{inc}}} \cup \{tx_j \mid \Pi_{L_{i-1}}(tx_j) = \text{None} \wedge tx_j \notin T_{(i-\theta)_{\text{inc}}}\}$.

We extend the deadline by the difference between the time the earliest worker is free and the deadline at the previous block: let δ_i be the maximum slack time at the end of round i, $\delta_i = \max_{w\in[0,1,..m_i-1]} \left(d_{i_{\text{new}}} - C_i^w\right)$, where C_i^w is the latest execution end time for worker w at round i $C_i^w = \max\{s_j + t_j \mid \Pi_{L_i}(tx_j) = (s_j, w)\ ,\ tx_j \in T_{i_{\text{sched}}}\}$. We define the extended deadline to be $d_{i_{\text{new}}} = d_i + \min(\delta_{i-1}, d_{i-1})$. We limit the maximum additional execution time per round by the execution time of the previous round, to disallow propagation of multiple rounds.

To ensure that transactions cannot be scheduled before the executor is fully done with transactions from a previous block we introduce a constraint: if a transaction tx_j from block i is scheduled at $\Pi_{L_i}(tx_j) = (s_j, w_j)$, then $s_j \geq \delta_{i-1} - (d_{i-1} - C_{i-1}^{w_j})$

Finally, we have to make sure that transactions do not access objects before all the transactions that read from or write to the overlapping objects of the previous block have finished: let $O_{tch_j} = R_j \cup W_j$ for a given $tx_j \in T_i$, then α_k^{i-1} is the latest time at which a transaction touching object k has been scheduled in round $i-1$: $\alpha_k^{i-1} = \max_{tx_j\in S_i^k}\left(s_j + t_j\right)$ with S_i^k the set of transactions that touch the object k in round i: $S_i^k = \{tx_j \in T_{i_{sched}} \mid k \in O_{tch_j}\}$. Then a transaction can only be scheduled after the latest time that an object has been freed in the previous round, so if a transaction tx_j from block i is scheduled at $\Pi_{L_i}(tx_j) = (s_j, w_j)$, then $s_j \geq \delta_{i-1} - (d_{i-1} - \max_{k\in O_{tch_j}}(\alpha_k^{i-1}))$.

This approach globally aligns the start times for the new block with the worker that is free the earliest in the previous schedule, while shifting other executors by how late their execution is compared to the earliest executor.

The main benefit of this approach is that it enables the continuity of execution without major changes to the problem structure. The model remains the same, barring the required addition of new hard-coded transactions that represent objects and workers not being free at the start of the round. This allows us to handle sequencing of each block as an independent combinatorial problem and additionally enables varying system parameters, such as d, m or σ, dynamically based on the demand.

Algorithm 2 Order-fair greedy scheduler.

input : L_{all}, d, m
output : Assignment A, where $A[tx] = (t, w_{id})$ or None

$Def \leftarrow \emptyset$ // deferred txs that will be added to the next block
$A \leftarrow \emptyset$
for tx ***in*** L_{all} **do**
  if does_conflict_with_any(tx, Def) **then** // tx conflicts with prev. deferred
    $A[tx] \leftarrow$ None
    $Def \leftarrow Def \cup \{tx\}$ // defer tx
    continue
  $WrkFreeAt \leftarrow \emptyset$
  for w ***in*** $0, 1, \cdots, m-1$ **do** // find earliest free slot per worker
    $WrkFreeAt[w] \leftarrow$ find_earliest(A, w, tx)
  $wrk \leftarrow$ choose_worker$(WrkFreeAt)$ // choose worker based on policy
  if $wrk =$ None **then** // no free worker
    $A[tx] \leftarrow$ None
    $Def \leftarrow Def \cup \{tx\}$ // defer tx
  else
    $A[tx] \leftarrow (WrkFreeAt[wrk], wrk)$ // schedule at earliest available slot
return A

3.4 Fair Ordering

So far, our formalization of sequencing can arbitrarily reorder transactions. A strictly opposite approach would enforce fair ordering [37, 60, 53, 51] and focus on maintaining an order of transactions closely aligned with the order in which they have been submitted to the mempool. We extend our continuous sequencing model to include this constraint in order to measure its impact on the performance.

To do so, we modify our models in two ways. First, T_{all} is replaced by $L_{\text{fair}} = [tx_1, tx_2, \cdots, tx_n]$ which corresponds to the total order over transactions produced by fair ordering. Additionally, the scheduling function must respect the causal order established by this order within L_{fair}, meaning that for every two conflicting transactions tx_i, tx_j where $i < j$: if the transaction positioned later in the order is scheduled, i.e., $\Pi(tx_j) = (s_j, w_j)$, then the earlier transaction must also be scheduled before it, i.e., $\Pi(tx_i) = (s_i, w_i)$ and $s_i + t_i \leq s_j$.

The scheduling algorithm must also be adapted to satisfy the fair ordering constraint. To preserve a simple and greedy design, we assume that the transactions passed to the scheduler are first totally ordered in a way that respects the causal order derived from L_{all}. As a result, the set of candidate schedules available to the scheduler is restricted. Specifically, this set consists of all possible traversals of the dependency graph G_{fair} constructed from L_{fair}.

More specifically, the scheduling algorithm is modified in the following ways. First, if a transaction is deferred, all conflicting transactions that are ordered after the deferred one are also deferred. Second, each transaction can only be scheduled after all conflicting transactions that were ordered before it.

The pseudocode is shown in Algorithm 2. Note that the inner workings of the find_earliest function have to be changed to return a time greater than the end time of any previously scheduled conflicting transaction.

4 Genetic Sequencing Algorithm

Rather than optimizing for an ideal scheduling function, a problem that could be formulated for a constraint solver, we opt against this approach due to its computational inefficiency. As noted by Chahoki et al. [9], using a constraint solver would introduce a new performance bottleneck, ultimately diminishing the throughput benefits of sequencing.

Instead, we focus on transaction ordering under a deterministic scheduling algorithm. This design choice explicitly accounts for imperfect information: the sequencer only has access to execution time estimates, which inherently slightly deviate from actual runtimes in practice, resulting in discrepancies between predicted and actual schedules [17].

Under this deterministic model, the most straightforward approach is to utilize a heuristic to order transactions. Because this problem closely resembles a multidimensional extension of the knapsack problem [19], sorting transactions by gas price serves as our baseline heuristic. For comparison, we also evaluate three alternative heuristics: (i) a randomized order; (ii) the historical arrival order (the sequence in which transactions appeared in the source data); and (iii) shortest execution time first.

4.1 Genetic Ordering

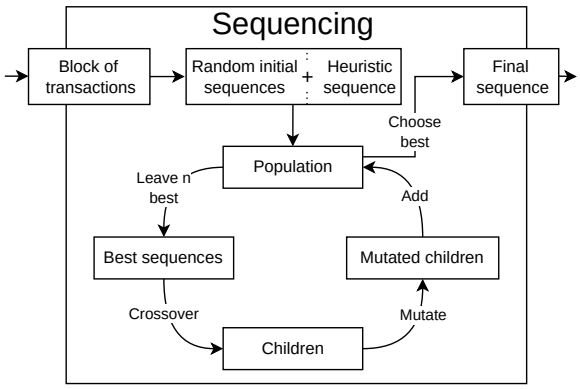


Figure 3 High level overview of our genetic transaction sequencing algorithm.

By fixing the scheduling function, the sequencing problem maps naturally onto a genetic algorithm framework. Here, each candidate solution is encoded as a permutation of the available transactions, with its sequence mapping directly to the scheduler's processing order. The fitness function is determined by applying the scheduling function and calculating the resulting profit. As illustrated in Fig. 3, the algorithm initializes a population of transaction sequences and iteratively optimizes them through crossover and mutation operations, retaining only the fittest half of the population at each generation.

Because the gas price heuristic often yields high-quality baselines in practice, we seed our initial population with this heuristic ordering rather than starting from a random ordering. Consequently, our approach shifts the genetic algorithm's objective from an expensive global optimum search to a highly localized refinement of an already strong initial solution.

Alg. 3 details the pseudocode of our genetic sequencing algorithm, which uses seeding, elitism, order crossover (OX) and insertion mutation, which have been shown to be effective for scheduling problems [11]. We discuss these techniques in the following:

Algorithm 3 Basis of our genetic sequencing algorithm.

input : $budget, pop\_size, T_{\text{all}}, \sigma, d, m$
output : L_{all}

$L_{\text{init}} \leftarrow$ highest_gas_price_order(T_{all}) // seeding
$P_{\text{init}} \leftarrow$ random_sequences(T_{all}, pop_size) $\cup \{L_{\text{init}}\}$
$P_{\text{eval}} \leftarrow$ eval($P_{\text{init}}, \sigma, d, m$)
$budget \leftarrow budget - (pop\_size + 1)$
while $budget > 0$ **do**
  $P_{\text{kids}} \leftarrow$ generate_kids($P_{\text{eval}}, pop\_size$) // OX
  $P_{\text{kids}} \leftarrow$ mutate(P_{kids}) // insertion
  for $kid \in P_{kids}$ **do**
    $P_{\text{eval}} \leftarrow P_{\text{eval}} \cup \{\text{eval}(kid, \sigma, d, m)\}$
    $budget \leftarrow budget - 1$
  $P_{\text{eval}} \leftarrow$ order_by_profit(P_{eval})
  $P_{\text{eval}} \leftarrow$ limit_population($P_{\text{eval}}, pop\_size$) // elitism
return $P_{\text{eval}}[0]$

- **Seeding.** An initial solution is added to the starting population. This solution is produced heuristically by ordering transactions based on their gas price [33].
- **OX.** To generate a child from two parent solution sequences, a random slice (random location and random length) is taken from one sequence and is inserted into another sequence at the same spot. Then the resulting sequence is scanned, and the possible second occurence of a transaction is removed [16].
- **Insertion mutation.** To mutate a sequence, a random transaction is removed and inserted into a different position [31].
- **Elitism.** A $(\mu + \lambda)$ survival strategy is employed. This means that each iteration parents (μ) and children (λ) are combined and among them a new population is formed by choosing the sequences with the highest fitness [32].

Seeding and elitism help to ensure faster convergence. To slightly offset this, we found it beneficial to set the mutation probability to 60%. The result is that the genetic algorithm acts similarly to a local search procedure, with a higher degree of exploration within the earlier iterations. Initially, an explicit local search procedure was also tested. It provided little benefit to the solution quality compared to extending the budget for the genetic algorithm, while requiring a lot of sequence evaluations. In our experiments, we use $pop\_size = 100$, $budget \in \{1000, 5000\}$, $m = 4$. The value we use for parameter d depends on the dataset ($62,500$ for Sui and $1,250,000$ for ETH).

4.2 Adding Fair Ordering

When adapting the genetic algorithm to maintain a fair order established by the consensus layer, we have to ensure that the transaction encoding represents a valid total order that respects the causal relations specified by L_{fair}. To do so, we apply the following changes. First, we change the initial seed L_{init} to correspond to the sequence obtained through a topological traversal of G_{fair} by prioritizing transactions with a higher gas price. Then, we generate the rest of the initial parents P_{init} by traversing G_{fair} uniformly at random. We modify the crossover operation so that it iterates over the parents and selects the first transaction whose dependencies are satisfied. The process continues until a valid total order consistent with G_{fair} is generated. We also modify the mutation operator as follows. First, for

a randomly selected transaction, we determine its maximal neighborhood of non-conflicting transactions within the current total order by identifying the closest conflicting transactions before and after it. Then, we relocate the selected transaction to a position that is chosen uniformly at random within this neighborhood.

5 Performance Evaluation

5.1 Settings

In our evaluation, we do not report the runtime of the algorithms, and instead focus on execution performance. The scheduling algorithms are allowed to run until the next block is generated and interrupted if needed. The genetic algorithm is an anytime algorithm, so result quality depends on the allotted time. We do not estimate the possible real-world execution time. Instead, we report it in terms of iterations of the estimated sequencer. For reference, on the continuous Sui workload, median per-round wall-clock time stayed under 300 ms for all heuristic sequencers, versus roughly 1.0 s for GE10 and 4.4 s for GE50. The runtime of the genetic sequencer is proportional to the number of total evaluations of the estimated sequencer. The number of evaluations is equal to the number of children per epoch multiplied by the number of epochs. Additionally, the evaluations of children within a single epoch can be done in parallel. To measure the impact of the allotted execution time on performance, we do two runs of the genetic sequencer with the population of 100: for 10 and 50 epochs, corresponding to $1,000$ and $5,000$ iterations of the heuristic sequencer, respectively. This allows us to quantify the benefit of additional computation time.

We focus on two scenarios with different solution quality metrics: sustained congestion and a single spike in congestion. In the sustained congestion scenario, we have congestion for the whole duration of sequencing and measure solution quality as the sum of gas prices paid by the scheduled transactions. In the single spike scenario we have congestion for a period of ten blocks and determine the solution quality by how quickly the sequencers managed to clear congestion. We define congestion clearance as the index of the first block after the spike period that contains no deferred transactions.

5.2 Datasets and Parameters

The performance of our sequencing approach is highly dependent on the structure and parameters of the transactions to be scheduled. To obtain measurements that are representative of the expected real-world performance, we base our datasets on two widely used blockchain systems that support smart contracts: Sui and Ethereum. We took transactions that were executed within a single day on both chains and streamed them to different sequencers. For Ethereum, we took the transactions from block 23359822 up to block 23369821, and for Sui we used transactions from epoch 840. Both represent recent active network conditions and span a sufficient transaction volume to capture realistic congestion dynamics.

To evaluate sequencing algorithms, we extract or generate the following parameters for a transaction tx_i:

- Touched objects (O_i) – object set of transactions.
- Execution time (t_i) – amount of computation the transaction takes.
- Gas price (g_i) – price that the party submitting the transaction is paying per execution unit of the transaction.

The following parameters are independent of transaction data and control how congested the blockchain is. We vary them across experiments to simulate different scenarios.

- Transactions per block ($|T_{inc}|$) – the number of transactions submitted for one block.

Table 2 Derivation of model parameters from real-world data.

Parameter	Sui Dataset	Ethereum Dataset
Execution Time	Transaction Effects	Transaction Receipt
Gas Price	Transaction Data	Transaction Receipt
Touched Objects	Shared Objects	N/A (Account-based)

- Maximum execution time (d) – the maximum execution time that a single worker can work for per one block.
- Maximum number of deferrals (θ) – the maximum number of blocks for which a transaction can be deferred before it gets canceled.
- Number of executors (m) – the number of executors that can execute the transactions in parallel.

All transactions are initially ordered according to their original positions within the blockchains to better capture temporal fluctuations in gas price trends, demand, and potential conflicts between contemporaneous transactions that may access overlapping sets of objects. To construct a continuous sequence of blocks, we partition the ordered transaction sequence into consecutive groups according to a predefined block size.

We map platform-specific parameters to the general model as summarized in Table 2. Execution time was derived from transaction effects for Sui and transaction receipt for Ethereum. The gas price was taken from the transaction data for Sui and from the receipt for Ethereum. Touched objects were determined from changed and unchanged objects specified in transaction effects for Sui and were not collected for Ethereum due to them not being explicitly listed in the available data and requiring high-overhead trace analysis of transactions. We explain below how we add object sets to Ethereum transactions.

We note that Ethereum uses a base fee system to determine the minimum gas price. Therefore, to better compare transactions across different time periods, we normalize the reported gas used value by this base fee. To do this, we divide the gas price by the base fee for a given block made during different time periods.

Since the results heavily depend on the data used, we want to first demonstrate the structure of the data and examine the distribution of values, as well as show their correlation with each other. Fig. 4 demonstrates the distribution of values for the Sui dataset, and Fig. 5 shows the distribution of values for the ETH dataset. From the datasets, we can see that the only two significantly correlated parameters are the number of touched objects and the execution time of transactions ($r = 0.45$), while neither execution time nor number of touched objects is significantly correlated with gas price.

As we mentioned before, we could not infer the accounts that were touched by Ethereum transactions. Instead we resampled Sui's dataset. To construct the Ethereum-based dataset, we projected Sui's data on object conflicts over Ethereum's execution and gas price records. Specifically, we picked a random point in the Sui dataset and appended this information to the Ethereum stream one transaction at a time, preserving the order in which both datasets were scheduled on the blockchain. This approach introduces a key limitation: it removes the correlation between the number of touched objects and the execution time. For this reason, our Ethereum's dataset is partially synthetic and we refer to it as the "Ethereum-based" dataset.

As shown in Fig. 6, we measured to what degree transactions overlap depending on the order in which they were submitted. The average Jaccard similarity follows a power-law decay as a function of transaction distance, with a sharp initial drop and a heavy tail. This confirms the significance of temporal locality and also shows the presence of several frequently touched global objects.

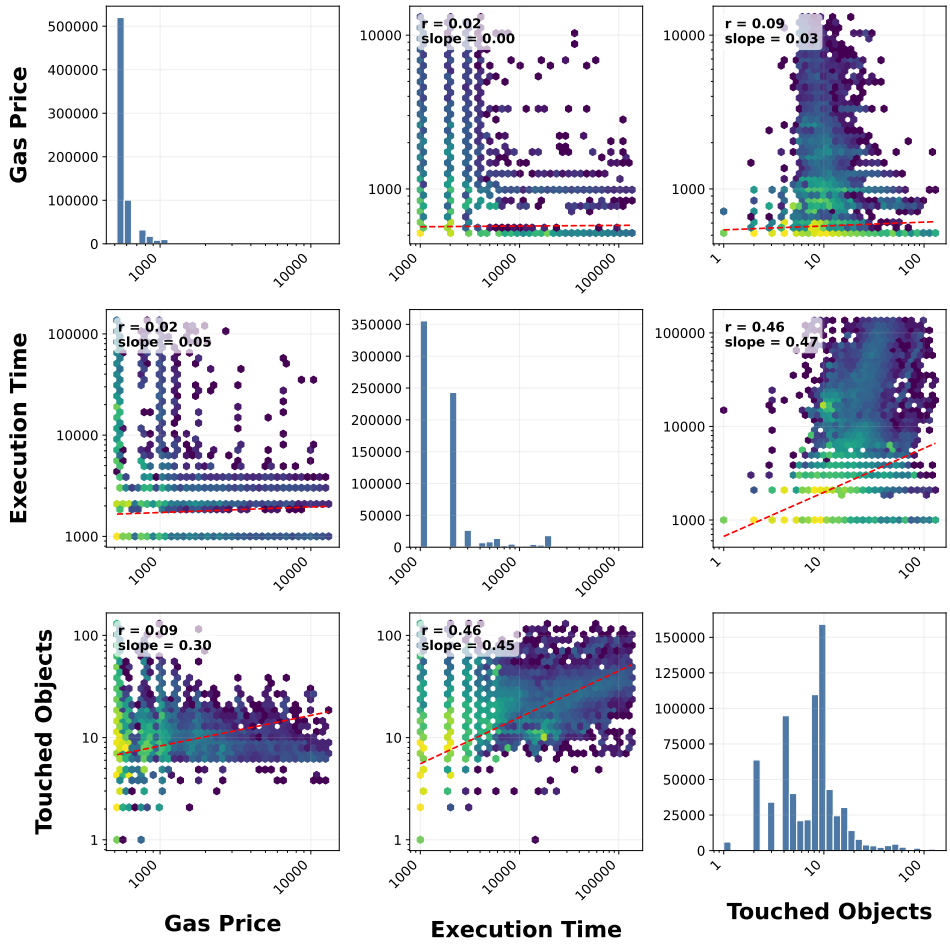


Figure 4 Log-scaled distribution and density plots for gas price, execution time, and total number of touched objects per transaction for SUI data. The units of these metrics are defined by the specific blockchain. Plots include a least-squares regression fit (slope is indicated) and Pearson correlation (r) for each pair of variables. Plots on the diagonal report the counts for each metric, other plots report the value of a metric depending on another one (e.g., number of touched objects depending on gas price).

The prevalence of temporal locality implies that a realistic dataset must consist of locally dependent transactions. It cannot be represented by independent sampling which would lead to a much lower degree of transaction conflicts. Furthermore, the heavy tail indicates the existence of heavily congested shared objects and reaffirms the need for congestion control.

5.3 Sequencing Baselines

We evaluated five different sequencing algorithms that rely on heuristics. All of them, except for the one labeled as "Sui", additionally use the gap-filling modification. The sequencers are as follows:

- Sui – sequencer that orders transactions by descending gas price and does not use gap-filling.
- Gas Price (GP) – sequencer that orders transactions by descending gas price and does use gap-filling.

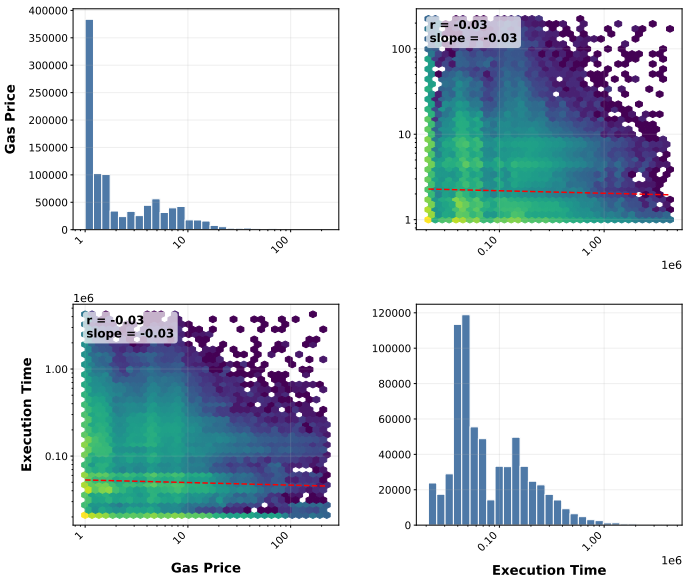


Figure 5 Log-scaled distribution and density plots for gas price and execution time for Ethereum data. Plots include a least-squares regression fit (slope is indicated) and Pearson correlation (r) for each pair of variables.

- Lowest Execution Time (LET) – sequencer that orders transactions by ascending execution time.
- Given – sequencer that orders transactions according to the order in which they have been executed on the blockchain.
- Random – sequencer that shuffles the order of transactions.

We use two versions of our genetic sequencing algorithm with two sets of parameters to demonstrate the impact of search depth on solution efficiency.

- Genetic 100/50 (GE50) – genetic sequencer with population of 100 that runs for 50 epochs.
- Genetic 100/10 (GE10) – genetic sequencer with population of 100 that runs for 10 epochs.

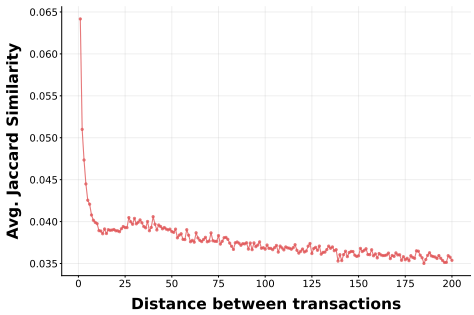
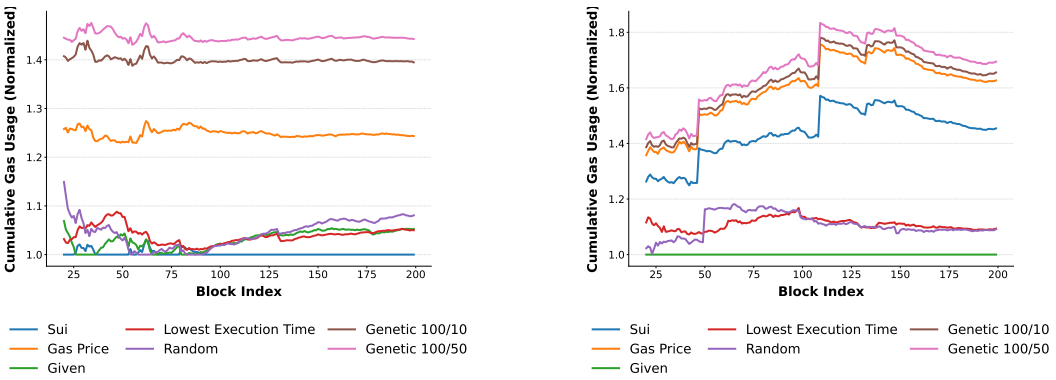


Figure 6 Average Jaccard similarity between transaction object sets as a function of execution-time distance, computed from a random sample of 50,000 transactions.



(a) Sui dataset.

(b) Ethereum dataset.

Figure 7 Total normalized validator profit under sustained congestion of 150 transactions per block for 200 blocks starting at block 20.

5.4 Sustained Congestion

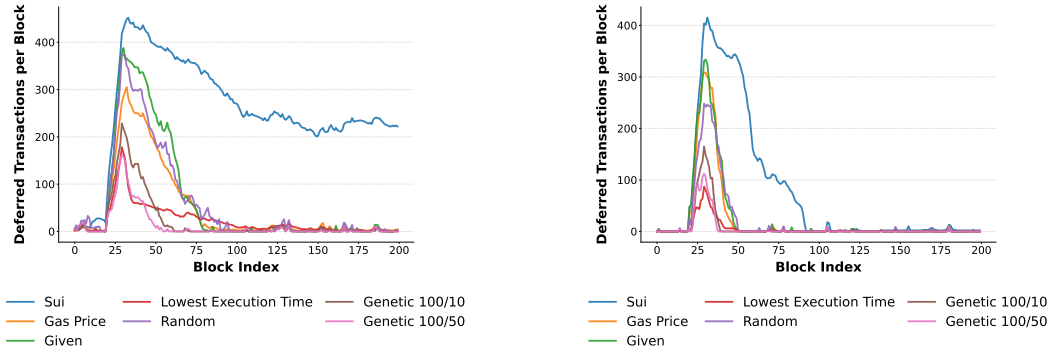
Unless stated otherwise, we use $m = 4$ executors, a maximum of 5 deferrals, 150 transactions per block, and an execution deadline of 62,500 time units, yielding an inclusion rate of approximately 50–60% for the GP sequencer. Simulations run for 200 blocks, and we report the cumulative validator profit normalized to the worst-performing sequencer. To highlight steady-state behavior, results are shown from block 20 onward.

Figure 7a presents the results for the Sui dataset. All heuristic baselines except GP achieve similar performance. GP improves validator profit by about 20% over the median heuristic, while the genetic sequencers provide a further 20% gain. Increasing the number of generations from 10 (GE10) to 50 (GE50) yields only a modest additional improvement of roughly 5%, suggesting diminishing returns beyond 10 generations.

For the Ethereum-based dataset, all of the parameters were identical, except for execution time, which was set to $1,250,000$ to provide the same $50-60\%$ inclusion rate. The Ethereum dataset results, shown in Fig. 7b, display smaller improvements compared to the Sui dataset. GE10 had only a slight improvement over GP, while GE50 had the same proportional improvement, as it did for Sui. We believe that the main reason for this is the difference in gas price distribution. The gas prices for Sui have low variance as can be seen in Fig. 4, while Ethereum's gas prices are much more varied as seen in Fig. 5. This means that most of the profit is received by scheduling high-paying transactions. Since the first epochs of the genetic algorithm focus on global search with random initialization of transaction ordering, the algorithm is much less likely to find better sequences, because sequencing a high-paying transaction late will have a more severe negative impact on profit. This is also suggested by the Sui sequencer having a much better performance with the Ethereum-based than with the Sui dataset, since it prioritizes gas price. Due to elitism, the later epochs focus much more on local search, as the transaction set gets filled with similar transactions.

5.5 Transaction Spike

To determine how well our genetic sequencing algorithm works in a scenario where congestion is not sustained but happens in sporadic bursts, we conducted an experiment where initially blocks contain a small number of transactions, then a high number of transactions for some time, and finally a small number of transactions again. We kept the number of executors at 4



(a) Sui dataset.

(b) Ethereum dataset.

Figure 8 Total absolute number of transactions deferred under a spike in transaction volume from block 20 to block 29. Blocks $0-19$ contain 30 transactions, blocks $20-29$ contain 100 transactions, and blocks $30-199$ contain 30 transactions.

and the execution deadline at $62,500$ units for the Sui dataset. The maximum number of deferrals was set to 200, so that no transactions would get canceled. The results showing the absolute number of deferred transactions for each block for Sui are shown in Fig. 8a. We see greater variations in the results of the heuristic baselines. The lack of gap filling prevented the Sui sequencer from recovering from the congestion. The Given and GP sequencers achieved similar results, recovering by block 130. The LET and Random sequencers recovered around block 90 with the LET sequencer showing a more rapid reduction of congestion, but then decreasing more slowly. Both of the genetic sequencers cleared congestion the fastest, with GE50 doing it by block 60 and GE10 by block 70.

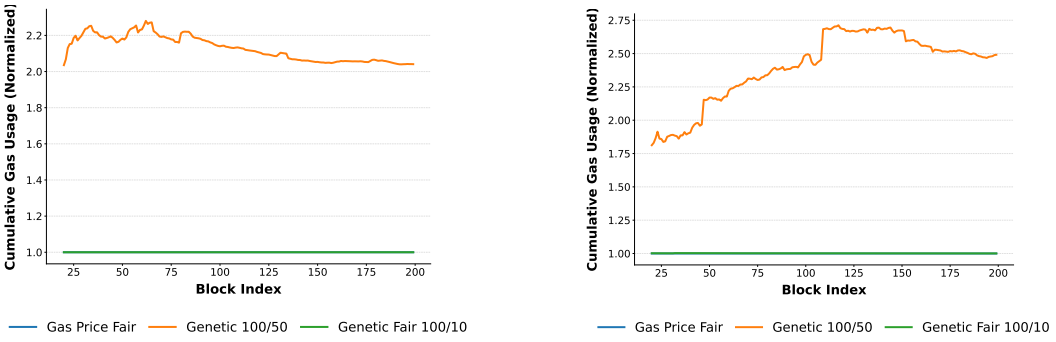
For the Ethereum-based dataset the parameters were the same, with the only difference of execution deadline being, once again, set to $1,250,000$ units. The results are shown in Fig. 8b. For the Ethereum-based dataset the congestion was less severe and generally cleared much sooner than in the case of Sui. Besides that, the trend of how quickly sequencers manage to resolve congestion mirrors the Sui results with the exception of the Random sequencer performing on par with the Given and GP sequencers. This is likely because gas prices do not play a role in this scenario.

5.6 Fair Ordering

To evaluate the impact of fair ordering, we repeated the sustained-congestion experiment with fair-ordering constraints enabled. Figs. 9a and 9b respectively report the results for the Sui and for the Ethereum-based datasets. For both cases the fair variants of GP and GE10 produce the same results, suggesting that the fair ordering constraint does not leave sufficient room for reordering causally independent transactions, rendering the genetic algorithm ineffective. Additionally, this experiment highlights that under sustained congestion, the addition of a fair ordering constraint reduces the profit by 50 to 60%. We attribute this cost to the high rate of transaction interdependence in the dataset (see Fig. 6).

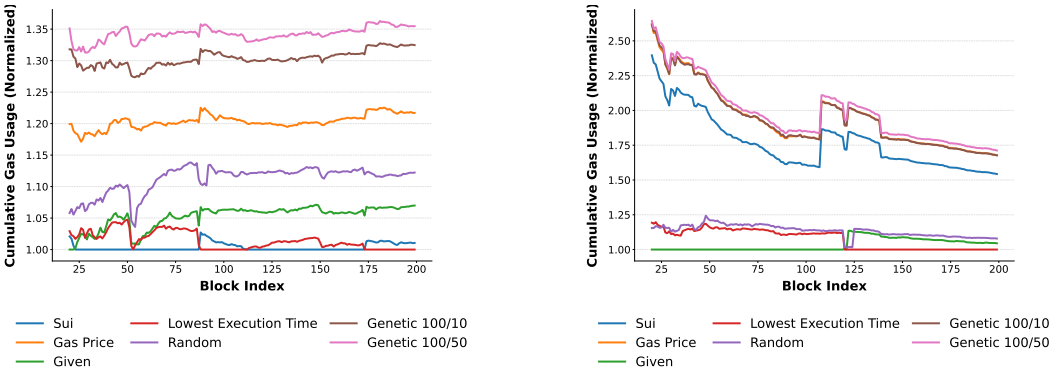
5.7 Robustness to Execution Time Estimation Errors

The strongest assumption in our model is the availability of accurate execution-time estimates. To evaluate robustness to prediction errors, we repeated our experiments with perturbed execution times. Transactions were sequenced using the original execution times as predictions,



(a) Sui dataset. **(b)** Ethereum dataset.

Figure 9 Total normalized validator profit under sustained congestion of 100 transactions per block for 200 blocks starting at block 20. Sequencers include the fair-ordering schedulers and GE50, the most efficient non-fair scheduler.



(a) Sui dataset. **(b)** Ethereum dataset.

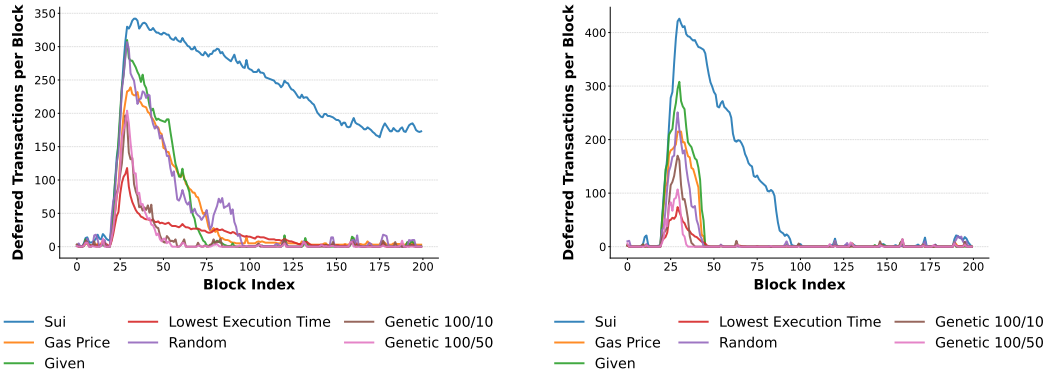
Figure 10 Total normalized validator profit under sustained congestion of 150 transactions per block for 200 blocks starting at block 20 with perturbed transaction execution times.

but executed using perturbed values. Perturbations were generated by multiplying each execution time by a log-normal random variable: for each transaction, we sampled $X \sim \mathcal{N}(0,\, 0.347^2)$ and used e^X as the scaling factor, yielding a 2σ range of approximately $[0.5, 2]$. This configuration matches realistic prediction errors reported by Lima Cabral et al. [17], corresponding to an average absolute error of about 28%.

Both the sustained congestion and the spike in congestion scenarios were tested with the addition of perturbation for both datasets. The sustained congestion for Sui seen in Fig. 10a and congestion spike for Sui and Ethereum seen in Figures 11a and 11b showed no statistically significant difference in speed of congestion clearance compared to the non-perturbed case. The sustained congestion for Ethereum seen in Fig. 10b shows that the genetic sequencers performed worse compared to their performance in the analogous experiment without perturbations, with only GE50 showing a very slight improvement over GP.

5.8 Throughput, Latency and Full Summary

In this work, we optimize for validator profit, reflecting the transaction ordering objective used by most existing blockchains, rather than maximizing throughput or minimizing latency. Since we make no assumptions about the underlying hardware or blockchain implementation,



(a) Sui dataset.

(b) Ethereum dataset.

Figure 11 Total absolute number of transactions deferred under a spike (blocks $20 - 29$) for Sui and Ethereum datasets with perturbed transaction execution times. Baseline traffic is 30 transactions/block, spiking to 150 transactions/block during the spike.

we report normalized performance metrics. Specifically, throughput is defined as the average number of transactions executed within a time budget d, representing the maximum execution time on a single worker (62,500 units for the Sui dataset and 1,250,000 units for the Ethereum-based dataset).

We measure latency as the time it takes from transaction sequencing until it has been executed in the units of d. Since our experiments simulated only the sequencing and execution steps, we assume no additional significant overhead at the dissemination and consensus steps and we do not account for the existing latency that they introduce. Additionally, since the simulation has no mechanisms to dynamically vary the time dedicated to sequencing, we assume that in each case sequencing runs until a worker becomes available. Latency values below 1 indicate early execution due to worker availability. Values between 1 and 2 indicate normal execution, and values above 2 indicate that the transaction has been deferred. Canceled transactions are unaccounted for, meaning that the reported latency does not capture the full cost of congestion for affected users.

Results are given in Tables 3 and 4 in Appendix A of our extended paper [50]. The genetic sequencer consistently achieves the highest validator profit under sustained congestion, outperforming the GP baseline by respectively 16% and 4% for the Sui and Ethereum-based datasets. For the congestion spike scenario, the genetic sequencer clears congestion faster than the baseline, reducing the post-spike recovery time from 60 to 25 blocks on the Sui dataset and from 20 to 8 blocks on the ETH-based dataset, with consistent results even under perturbation. LET achieves the highest throughput at a significant cost to validator profit while the genetic sequencer consistently improves on the baseline throughput. Latency is broadly comparable across non-fair sequencers, though these figures should be interpreted cautiously, as the assumption of equal sequencing time is unrealistic. Fair ordering imposed a profit penalty of approximately 50% and failed to clear congestion on the Sui dataset.

6 Related Work

We review four related literature areas. First, we examine transaction sequencing in production blockchains, ranging from full validator reordering to protocol-level congestion control. We then consider fair ordering approaches that favor transparency and manipulation resist-

ance over congestion-aware optimization. Next, we review prior work framing sequencing as a discrete combinatorial optimization problem, with an overview provided in Table 5 in Appendix B of our full paper [50]. Finally, we briefly survey work on execution optimization.

Sequencing in production blockchains. In Bitcoin, transaction sequencing within a block was historically of limited importance. The main design goal was to maintain a consistent total order of blocks and ensure immutability unless an adversary controls a majority of hash power. Within a block, ordering was fully determined by the miner. Because Bitcoin uses a UTXO model, each transaction independently defines its inputs and outputs, making intra-block order irrelevant for validity. Only inclusion matters. In practice, transactions are ordered using ancestor fee rate policies, with dependency handling via Child-Pays-for-Parent (CPFP) [43].

Ethereum initially followed a similar approach, but its account-based design makes ordering critical. Transactions read and modify shared state, so execution depends on ordering. This effect is strengthened by smart contracts [7], which allow arbitrary state transitions. Early Ethereum clients such as Geth commonly ordered transactions greedily by gas price [38, 28]. However, Daian et al. [13] showed that validators can increase profit by reordering transactions and inserting their own, a phenomenon known as Miner Extractable Value (MEV). This led to MEV-boost and external block builders, with around 90% adoption by 2022 [47], increasing validator rewards by roughly 250–400% [45, 42] but reducing transparency and decentralization. MEV is commonly extracted through private transaction submission or bundle-based ordering constraints [21].

Sui is a DAG-based system that distinguishes between owned and shared objects, corresponding roughly to UTXO-like and account-like semantics. Owned-object transactions can be partially ordered without global sequencing, while shared objects require global ordering [46]. Originally, conflict resolution for owned objects relied on certification, but in current designs conflicting transactions must be resolved either during consensus or via sequencing. Sui's sequencing layer also performs congestion control: transactions are first ordered by gas price and then selectively deferred based on estimated execution time and object-level capacity constraints [24, 23].

Solana has also modified its sequencing design in pursuit of high throughput [59]. Early designs used per-thread queues with local priority ordering, which produced partial and non-deterministic global order. Later versions introduced a global priority heap before thread assignment (v1.18). However, sequencing remains largely implementation defined, which has enabled MEV-oriented validator clients such as Jito [41]. This illustrates a recurring pattern: when sequencing is not specified at the protocol level, economic incentives tend to drive systems toward opaque, profit-maximizing ordering strategies.

Fair ordering. The lack of transparency in transaction ordering and the rise of MEV-boosting have motivated research on fair transaction ordering [36]. Its goal is to align transaction order as closely as possible with real submission times. Two main approaches exist. Ordering linearizability, introduced in Pompē [60], uses validator-provided timestamps and orders transactions by the median of a subset of its timestamps. Batch order fairness, as in Themis [37], aggregates full per-validator orderings into a global sequence. Fair ordering improves protocol-level fairness and reduces MEV opportunities, but it imposes a rigid global order, lacks congestion control, and ignores validator incentives.

Combinatorial optimization for sequencing. Under deterministic execution, sequencing reduces to block construction. With sequential execution and position-independent execution times, the problem becomes a knapsack instance and is therefore NP-hard [19]. Karmegam et al. [34] formulate block construction as a mixed-integer program maximizing validator revenue under predefined dependencies and implicit fair-ordering constraints. Their heuristic performs similarly to a gas-price greedy baseline, consistent with our finding that genetic optimization offers little benefit when fairness constraints are enforced. Conthereum [9] models sequencing as a job-shop scheduling problem, optimizing makespan under unrestricted reordering while requiring all transactions to fit within a block. In contrast, we consider continuous execution with overlapping blocks, evaluate robustness to execution-time prediction errors, and study both unconstrained and fair transaction ordering.

Execution. Most prior work targets improvements in the execution layer. Pilotfish [39] is a distributed engine that executes transactions in order while dynamically assigning independent ones to workers, enabling parallelism under a crash-recovery model with low communication overhead. Extensions of Hyperledger Fabric, such as HTFabric and Fabric++ [55, 54], further increase parallelism by reordering transactions during execution. While this improves throughput, it biases against interdependent transactions. Other approaches avoid requiring predeclared object sets. Anjana et al. [4] use software transactional memory (STM) to speculatively execute transactions in parallel with rollback on conflicts. Block-STM [27] refines dependency estimates after rollbacks, reducing redundant execution.

7 Discussion

Gas price. Users do not seem to price the congestion they impose in their transactions, as shown by the weak correlation between gas price and the number of accessed objects (Figs. 4 and 5). Furthermore, we assume a static, user-specified gas price throughout execution. Incorporating dynamic, multi-dimensional pricing mechanisms tailored to parallel execution environments [1, 58] is a promising direction for future work.

Fair ordering constraints and conflict resolution. We derived causal order from total ordering. The observed 50% to 60% profit penalty under sustained congestion highlights the strictness of enforcing this causal chain, at least with the algorithms and datasets we considered. Whether this penalty is verified with all schedulers and datasets is an open question. Alternatively, batch order fairness [37, 51] introduces dependency cycles that could offer scheduling flexibility, though such cycles are rare in practice. Exploring alternative optimization methods like GRASP or simulated annealing remains future work.

External gameability and input resilience. Adversaries cannot game placement via gas bidding without incurring direct execution fees. Furthermore, our evaluation in Section 5.7 demonstrates that our genetic algorithm remains robust against significant execution-time estimation errors. To prevent the case where adversaries declare unused shared objects to trigger inaccurate deferrals, our framework seeds optimization with gas price heuristics to protect higher-paying honest transactions. A possible direction might be the development of native localized fee markets targeting specific shared objects, introducing dynamic fee penalties for declared-but-unaccessed objects or localized pricing mechanisms based on state-declaration accuracy [1, 58].

Real-Time Sequencing. While prior work [9] found genetic algorithms and constraint solvers too slow for online sequencing, our approach uses an anytime algorithm with strict execution deadlines. Seeding the initial population with a gas-price heuristic provides an immediately deployable baseline schedule, while hard execution deadlines interrupt search to ensure continuous worker utilization.

Architectural assumptions. Our model assumes deterministic sequencing over committed consensus batches. While single-leader block builders could act as privileged clients to censor transactions, our target deployment environment relies on DAG-based consensus ([14, 56, 5]). Because transactions are broadcast to multiple nodes and committed to consensus blocks before sequencing, individual validators cannot selectively exclude transactions during optimization without violating state consistency.

8 Conclusion

We have formalized a blockchain-independent model for evaluating the impact of transaction ordering on validator profit and congestion clearance speed. We extended this model to account for continuity of execution and fair ordering. We then proposed a genetic ordering algorithm that takes into account transaction execution time estimates and object conflict information. This approach outperforms the heuristic baseline, increasing validator profit by approximately 15%, and improves congestion control in a high congestion scenario. We have also shown that the performance of this approach does not degrade significantly under realistic execution time estimation errors for the purpose of congestion control. Performance also does not degrade significantly in terms of profit maximization when gas prices exhibit little variation. We finally integrated a fair ordering constraint and found that under such an assumption, sequencing provides no benefit. Introducing fair ordering constraints lead validators to lose around 50 to 60% of their profit under sustained congestion.

References

1 Bahar Acilan, Andrei Constantinescu, Lioba Heimbach, and Roger Wattenhofer. Transaction fee market design for parallel execution. *arXiv preprint arXiv:2502.11964*, 2025. doi:10.48550/arXiv.2502.11964.

2 Elvira Albert, Pablo Gordillo, Alejandro Hernández-Cerezo, Albert Rubio, and Maria A Schett. Super-optimization of smart contracts. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 31(4):1–29, 2022. doi:10.1145/3506800.

3 Mohammad Javad Amiri, Divyakant Agrawal, and Amr El Abbadi. Parblockchain: Leveraging transaction parallelism in permissioned blockchain systems. *CoRR*, abs/1902.01457, 2019. arXiv:1902.01457.

4 Parwat Singh Anjana, Sweta Kumari, Sathya Peri, Sachin Rathor, and Archit Somani. An efficient framework for optimistic concurrent execution of smart contracts, 2019. arXiv:1809.01326.

5 Kushal Babel, Andrey Chursin, George Danezis, Anastasios Kichidis, Lefteris Kokoris-Kogias, Arun Koshy, Alberto Sonnino, and Mingwei Tian. Mysticeti: Reaching the latency limits with uncertified dags. In *NDSS*, 2025.

6 Sam Blackshear, Andrey Chursin, George Danezis, Anastasios Kichidis, Lefteris Kokoris-Kogias, Xun Li, Mark Logan, Ashok Menon, Todd Nowacki, Alberto Sonnino, Brandon Williams, and Lu Zhang. Sui lutris: A blockchain combining broadcast and consensus, 2024. arXiv:2310.18042.

7 Vitalik Buterin. Ethereum white paper: A next generation smart contract & decentralized application platform, 2013. URL: https://github.com/ethereum/wiki/wiki/White-Paper.

8 Christian Cachin et al. Architecture of the hyperledger blockchain fabric. In *Workshop on distributed cryptocurrencies and consensus ledgers*, volume 310(4), pages 1–4. Chicago, IL, 2016.

9 Atefeh Zareh Chahoki, Maurice Herlihy, and Marco Roveri. Conthereum: Concurrent ethereum optimized transaction scheduling for multi-core execution. In *2025 7th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, pages 1–10. IEEE, 2025. doi:10.1109/BRAINS67003.2025.11302903.

10 Atefeh Zareh Chahoki and Marco Roveri. Static analysis for detecting transaction conflicts in ethereum smart contracts. *arXiv preprint arXiv:2507.04357*, 2025. doi:10.48550/arXiv.2507.04357.

11 Runwei Cheng, Mitsuo Gen, and Yasuhiro Tsujimura. A tutorial survey of job-shop scheduling problems using genetic algorithms—i. representation. *Computers & Industrial Engineering*, 30(4):983–997, 1996. doi:10.1016/0360-8352(96)00047-2.

12 Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer. On scaling decentralized blockchains. In Jeremy Clark, Sarah Meiklejohn, Peter Y.A. Ryan, Dan Wallach, Michael Brenner, and Kurt Rohloff, editors, *Financial Cryptography and Data Security*, pages 106–125, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

13 Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE symposium on security and privacy (SP)*, pages 910–927. IEEE, 2020. doi:10.1109/SP40000.2020.00040.

14 George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. Narwhal and Tusk: a DAG-based mempool and efficient BFT consensus. In *Proceedings of the Seventeenth European Conference on Computer Systems*, EuroSys '22, pages 34–50, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3492321.3519594.

15 Sourav Das, Vinith Krishnan, and Ling Ren. Efficient cross-shard transaction execution in sharded blockchains, 2021. arXiv:2007.14521.

16 Lawrence Davis et al. Applying adaptive algorithms to epistatic domains. In *IJCAI*, volume 85, pages 162–164, 1985.

17 Danilo Rafael de Lima Cabral, Pedro Antonino, and Augusto Cezar Alves Sampaio. Demystification and near-perfect estimation of minimum gas limit and gas used for ethereum smart contracts. *Journal of Cloud Computing*, 14(1):29, 2025. doi:10.1186/S13677-025-00751-Y.

18 Thomas Dickerson, Paul Gazzillo, Maurice Herlihy, and Eric Koskinen. Adding concurrency to smart contracts. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, PODC '17, pages 303–312, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3087801.3087835.

19 Saulo Dos Santos, Chukwuka Chukwuocha, Shahin Kamali, and Ruppa K. Thulasiram. An efficient miner strategy for selecting cryptocurrency transactions. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 116–123, 2019. doi:10.1109/Blockchain.2019.00024.

20 François Ezard, Can Umut Ileri, and Jérémie Decouchant. Nemo: Faster parallel execution for highly contended blockchain workloads. In *2025 7th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, pages 1–5. IEEE, 2025. doi:10.1109/BRAINS67003.2025.11302922.

21 Flashbots. Understanding bundles, 2025. Flashbots Documentation. URL: https://docs.flashbots.net/flashbots-mev-share/searchers/understanding-bundles.

22 Solana Foundation. Sealevel - parallel processing thousands of smart contracts, September 2019.

23 Sui Foundation. Object-based local fee markets. Sui Documentation, 2024. Accessed: 2026. URL: https://docs.sui.io/guides/developer/objects/local-fee-markets.

24 Sui Foundation. Streamlining transactions with sui's shared object congestion control. Sui Blog, September 2024. Accessed: 2026. URL: https://blog.sui.io/shared-object-congestion-control/.

25 Yingzi Gao, Yuan Lu, Zhenliang Lu, Qiang Tang, Jing Xu, and Zhenfeng Zhang. Dumbo-ng: Fast asynchronous bft consensus with throughput-oblivious latency. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1187–1201, 2022. doi:10.1145/3548606.3559379.

26 Luciano García-Bañuelos, Alexander Ponomarev, Marlon Dumas, and Ingo Weber. Optimized execution of business processes on blockchain. In *Business Process Management: 15th International Conference, BPM 2017, Barcelona, Spain, September 10–15, 2017, Proceedings 15*, pages 130–146. Springer, 2017. doi:10.1007/978-3-319-65000-5_8.

27 Rati Gelashvili, Alexander Spiegelman, Zhuolun Xiang, George Danezis, Zekun Li, Dahlia Malkhi, Yu Xia, and Runtian Zhou. Block-stm: Scaling blockchain execution by turning ordering curse to a performance blessing. In *Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, pages 232–244, 2023. doi:10.1145/3572848.3577524.

28 go-ethereum contributors. go-ethereum miner/worker.go (commit 290e851), 2017. Accessed: 2026. URL: https://github.com/ethereum/go-ethereum/blob/290e851f57f5d27a1d5f0f7ad784c836e017c337/miner/worker.go.

29 Yue Hao, Yi Li, Xinghua Dong, Li Fang, and Ping Chen. Performance analysis of consensus algorithm in private blockchain. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 280–285, 2018. doi:10.1109/IVS.2018.8500557.

30 Can Umut Ileri, Andrew Cullen, Olivia Saa, Roman Overko, and Luigi Vigneri. Enhanced transaction sequencing for modular distributed ledgers. In *2025 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–3. IEEE, 2025. doi:10.1109/ICBC64466.2025.11114535.

31 Khalid Jebari, Mohammed Madiafi, et al. Selection methods for genetic algorithms. *International Journal of Emerging Sciences*, 3(4):333–344, 2013.

32 Kenneth A. De Jong. *An Analysis Of The Behavior Of A Class Of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.

33 Bryant A. Julstrom. Seeding the population: improved performance in a genetic algorithm for the rectilinear steiner problem. In *Proceedings of the 1994 ACM Symposium on Applied Computing*, SAC '94, pages 222–226, New York, NY, USA, 1994. Association for Computing Machinery. doi:10.1145/326619.326728.

34 Arivarasan Karmegam, Lucianna Kiffer, and Antonio Fernández Anta. Exploiting multi-core parallelism in blockchain validation and construction, 2026. doi:10.48550/arXiv.2602.03444.

35 Idit Keidar, Eleftherios Kokoris-Kogias, Oded Naor, and Alexander Spiegelman. All you need is dag. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, pages 165–175, 2021. doi:10.1145/3465084.3467905.

36 Mahimna Kelkar, Soubhik Deb, and Sreeram Kannan. Order-fair consensus in the permissionless setting. In *Proceedings of the 9th ACM on ASIA Public-Key Cryptography Workshop*, pages 3–14, 2022. doi:10.1145/3494105.3526239.

37 Mahimna Kelkar, Soubhik Deb, Sishan Long, Ari Juels, and Sreeram Kannan. Themis: Fast, strong order-fairness in byzantine consensus. In *Proceedings of the 2023 acm sigsac conference on computer and communications security*, pages 475–489, 2023. doi:10.1145/3576915.3616658.

38 Seoung Kyun Kim, Zane Ma, Siddharth Murali, Joshua Mason, Andrew Miller, and Michael Bailey. Measuring ethereum network peers. In *Proceedings of the Internet Measurement Conference 2018*, pages 91–104, 2018. URL: https://dl.acm.org/citation.cfm?id=3278542.

39 Quentin Kniep, Lefteris Kokoris-Kogias, Alberto Sonnino, Igor Zablotchi, and Nuda Zhang. Pilotfish: Distributed execution for scalable blockchains, 2025. arXiv:2401.16292.

40 Wenzheng Li and Mingsheng He. Comparative analysis of bitcoin, ethereum, and libra. In *2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)*, pages 545–550, 2020. doi:10.1109/ICSESS49938.2020.9237710.

41 Lostin. The truth about solana local fee markets, January 2025. Helius Blog. URL: https://www.helius.dev/blog/solana-local-fee-markets.

42 Davide Mancino, Alberto Leporati, Marco Viviani, and Giovanni Denaro. A role and reward analysis in off-chain mechanisms for executing mev strategies in ethereum proof-of-stake. *Distrib. Ledger Technol.*, 4(3), August 2025. doi:10.1145/3672405.

43 Johnnatan Messias, Mohamed Alzayat, Balakrishnan Chandrasekaran, Krishna P Gummadi, Patrick Loiseau, and Alan Mislove. Selfish & opaque transaction ordering in the bitcoin blockchain: the case for chain neutrality. In *Proceedings of the 21st ACM Internet Measurement Conference*, pages 320–335, 2021. doi:10.1145/3487552.3487823.

44 Bulat Nasrulin, Georgy Ishmaev, Jérémie Decouchant, and Johan Pouwelse. Lo: An accountable mempool for mev resistance. In *Proceedings of the 24th international middleware conference*, pages 98–110, 2023. doi:10.1145/3590140.3629108.

45 Nero.eth. Is it worth using mev-boost?, June 2024. Ethereum Research Forum. URL: https://ethresear.ch/t/is-it-worth-using-mev-boost/19753.

46 Roman Overko. A study on shared objects in sui smart contracts. In *2024 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–7, 2024. doi:10.1109/ICBC59979.2024.10634357.

47 Burak Öz, Danning Sui, Thomas Thiery, and Florian Matthes. Who wins ethereum block building auctions and why? *arXiv preprint*, 2024. doi:10.48550/arXiv.2407.13931.

48 Nikita Polyanskii, Sebastian Müller, and Ilya Vorobyev. Starfish: A high throughput bft protocol on uncertified dag with linear amortized communication complexity. *IACR Cryptol. ePrint Arch.*, 2025:567, 2025. URL: https://api.semanticscholar.org/CorpusID:277637515.

49 Miroslav Popovic, Branislav Kordic, and Ilija Basicevic. Estimating transaction execution times for a software transactional memory. In *2016 Sixth International Conference on Information Science and Technology (ICIST)*, pages 137–141. IEEE, 2016.

50 Artjom Pugatsov, Can Umut Ileri, and Jérémie Decouchant. The blockchain execution dilemma: Optimizing revenue xor fair ordering. *arXiv preprint arXiv:2604.23266*, 2026. doi:10.48550/arXiv.2604.23266.

51 Marko Putnik and Jérémie Decouchant. Herring: Parallel batch-order-fairness on dag-based blockchain consensus. *arXiv preprint arXiv:2605.23648*, 2026. doi:10.48550/arXiv.2605.23648.

52 Pingcheng Ruan, Dumitrel Loghin, Quang-Trung Ta, Meihui Zhang, Gang Chen, and Beng Chin Ooi. A transactional perspective on execute-order-validate blockchains. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, pages 543–557, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3318464.3389693.

53 Giulio Segalini, Yigit Çolakoğlu, Marko Putnik, and Jérémie Decouchant. Tilikum: Transaction fair ordering on a dag without weak edges. *arXiv preprint arXiv:2606.27250*, 2026. doi:10.48550/arXiv.2606.27250.

54 Ankur Sharma, Felix Martin Schuhknecht, Divya Agrawal, and Jens Dittrich. Blurring the lines between blockchains and database systems: the case of hyperledger fabric. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, pages 105–122, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3299869.3319883.

55 Jaeyub Song, Juyeong Jeong, Jemin Lee, Inju Na, and Min-Soo Kim. Htfabric: A fast re-ordering and parallel re-execution method for a high-throughput blockchain. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, CIKM '24, pages 2118–2127, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3627673.3679606.

56 Alexander Spiegelman, Neil Giridharan, Alberto Sonnino, and Lefteris Kokoris-Kogias. Bullshark: Dag bft protocols made practical. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2705–2718, 2022. doi:10.1145/3548606.3559361.

57 Sergei Tikhomirov, Ekaterina Voskresenskaya, Ivan Ivanitskiy, Ramil Takhaviev, Evgeny Marchenko, and Yaroslav Alexandrov. Smartcheck: Static analysis of ethereum smart contracts. In *Proceedings of the 1st international workshop on emerging trends in software engineering for blockchain*, pages 9–16, 2018. doi:10.1145/3194113.3194115.

58 Sarisht Wadhwa, Aviv Yaish, Fan Zhang, and Kartik Nayak. Perils of parallelism: Transaction fee mechanisms under execution uncertainty, 2026. URL: https://eprint.iacr.org/2026/649.

59 Anatoly Yakovenko. Solana: A new architecture for a high performance blockchain v0. 8.13, 2018. URL: https://solana.com/solana-whitepaper.pdf.

60 Yunhao Zhang, Srinath Setty, Qi Chen, Lidong Zhou, and Lorenzo Alvisi. Byzantine ordered consensus without byzantine oligarchy. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 633–649, 2020. URL: https://www.usenix.org/conference/osdi20/presentation/zhang-yunhao.

Inside Qubic's Selfish Mining Campaign on Monero: Evidence, Tactics, and Limits

Suhyeon Lee ✉
ShardLab, Seoul, Republic of Korea
Hashed Open Research, Seoul, Republic of Korea

Hyeongyeong Kim
Korea University, Seoul, Republic of Korea

Abstract

In 2025, the blockchain network Qubic conducted a mining campaign on Monero, a privacy-focused cryptocurrency. It publicly presented the campaign as a "51% takeover" and a demonstration of selfish mining. The episode provides a rare opportunity to test such claims in a privacy-preserving proof-of-work network, where pool attribution and private release decisions are difficult to observe. We combine Monero node measurements, Qubic pool observations, community-shared artifacts, and disclosed view keys to attribute blocks and identify candidate withholding periods. Rather than sustained majority control, we observe elevated orphaning and deeper reorganizations. Qubic did not follow a single optimized selfish mining strategy, instead varying between standard selfish mining and more defensive block-release decisions. Across the campaign, it gained no reward advantage over honest mining. Community monitoring and Qubic's countermeasures reveal how real-world mining campaigns evolve in response to an active ecosystem. Overall, the campaign disrupted Monero without exhibiting a stable profit-maximizing strategy, leaving open disruptive incentives beyond direct mining rewards.

2012 ACM Subject Classification Security and privacy → Distributed systems security; Computer systems organization → Peer-to-peer architectures; Networks → Network measurement

Keywords and phrases Blockchain, Proof-of-Work, Selfish Mining, Monero

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.28

Related Version *Full Version*: `https://arxiv.org/abs/2512.01437`

Supplementary Material *Dataset (Data and Source Code)*: `https://github.com/shlee-lab/Qubic-selfish-mining-study`, archived at `swh:1:dir:0abc35958926898433f7dd17c43e31a822a4378e`

Funding *Suhyeon Lee*: Suhyeon Lee received support from ShardLab.

Acknowledgements We are grateful to the Monero community for sharing artifacts and empirical observations that complemented our measurements. We especially thank DataHoarder (`https://github.com/WeebDataHoarder`) and Sergei Chernykh (sech1) for providing access to supplemental datasets and answering technical questions that helped us validate and refine parts of our analysis. We thank the anonymous reviewers for their thoughtful feedback, which helped improve the paper.

1 Introduction

Selfish mining can undermine the incentive compatibility of proof-of-work systems [2]. By withholding and strategically releasing blocks, a miner can orphan competing blocks and waste other miners' resources. Despite extensive theoretical study [10, 16], large-scale deployments have rarely been documented on live networks.

In August 2025, Qubic, a blockchain network built around redirecting mining resources to external computations, launched a high-profile mining campaign on Monero (XMR). Qubic reported selling the mined XMR to fund buybacks and burns of its native QUBIC token and to pay miner bonuses, tying the operation to its own token economy [14, 15]. It publicly

© Suhyeon Lee and Hyeongyeong Kim;
licensed under Creative Commons License CC-BY 4.0

Table 1 Research questions and main findings.

Research question	Main finding
RQ1. Can Qubic activity be reliably identified despite Monero's privacy design?	Yes. Community-observed data, extra-nonce patterns, pool observations, and disclosed view keys jointly support Qubic labels and interval-level aggregate measurements.
RQ2. Did Qubic sustain majority mining power, and what direct chain-level impact did its activity have?	We do not observe sustained majority control. Qubic's share rose sharply in identified periods, but the clearer impact is elevated orphaning and deeper reorganizations.
RQ3. Do static selfish mining models explain Qubic's observed main-chain share?	No single fixed strategy explains all periods. Qubic's observed behavior generally fell between standard selfish mining and a more conservative release strategy, while its execution did not appear consistently optimized.
RQ4. Did Qubic gain mining rewards from this behavior, and what execution factors shaped gains or losses?	The withholding intervals do not show consistent reward gains over honest mining. Later difficulty-adjustment spillovers mitigated the shortfall, but Qubic remained below the honest mining baseline across the identified periods and subsequent gaps.
RQ5. How did the Monero ecosystem respond, and how should those responses affect interpretation of the incident?	The community monitored Qubic's mining activity and relayed reconstructed blocks when possible. Qubic treated this exposure as an operational risk and progressively strengthened its countermeasures, reducing visibility into its internal mining activity over time.

described the campaign as a "51% takeover" and a demonstration of selfish mining. The incident therefore raised practical questions that public dashboards could not answer. We ask what occurred on chain, how the campaign was executed operationally, and whether it was economically effective or primarily disruptive.

This framing also raises a question that block-reward profitability alone cannot answer. Kroll et al. describe a *Goldfinger attack* as an attack whose utility comes from weakening the target cryptocurrency rather than from earning native mining rewards. We restrict our profitability analysis to mining rewards and revisit the available market-price evidence in Section 9.

Answering these questions is especially challenging in Monero. Blocks do not carry explicit pool identifiers, and key signals of selfish mining are off-chain by design. We therefore study the campaign by combining multiple vantage points: blocks on the main chain and orphan blocks from Monero nodes, Qubic pool job notifications, and Qubic-network artifacts shared by the Monero community. Table 1 summarizes the five research questions and our main findings.

We make four contributions:

- **Multi-source attribution and dataset.** We combine node-observed blocks, Qubic pool signals, and community-observed data, and validate the resulting attribution against disclosed view keys while quantifying its coverage limits.
- **Empirical campaign measurement.** We reconstruct Qubic's mining activity, assess its public claim of majority mining, and measure its effects on orphaning and chain reorganizations.
- **Strategy and economic evaluation.** We compare the observed behavior with the standard selfish mining model and a conservative-release variant, analyze race outcomes, and account for delayed difficulty-adjustment spillovers when evaluating mining rewards.
- **Operational response and inference limits.** We examine community monitoring and Qubic's countermeasures as operational factors that affected block observability and limit what can be inferred about Qubic's intended release strategy.

Organization

Section 2 reviews related work. Section 3 describes the data sources, block attribution, and coverage limits. Section 4 measures Qubic's mining share and its effects on orphaning and chain reorganizations. Sections 5–7 analyze Qubic's selfish mining behavior and its immediate and delayed effects on mining rewards. Section 8 analyzes community monitoring, Qubic's countermeasures, and the resulting limits on strategy inference. Section 9 discusses the broader economic interpretation and mitigation implications, and Section 10 concludes.

2 Related Work

Selfish mining and strategy variants

Selfish mining was formalized as a profitable deviation under specific network and propagation assumptions [2]. Follow-on work generalized the attacker's strategy space, including eclipse variants [10]. Other work studied optimal and efficiently computable selfish mining strategies using Markov decision processes [16, 24], and examined how profitability and thresholds change across protocols such as Ethereum [4]. Intermittent selfish mining has also been studied in the context of difficulty adjustment [11]. Our work builds on these foundations but focuses on what can be validated in a real campaign where internal attacker state is not observable and attribution is difficult. Recent work has also provided statistical and empirical tools for detecting behavior resembling selfish mining in real PoW systems [9].

Propagation, forks, and difficulty dynamics

Propagation delays and network asymmetries shape stale blocks and tie-breaking, and are central to selfish mining performance [1, 3]. Beyond generic propagation effects, Monero's network-layer attack surface has been studied in the context of eclipse-style isolation [17]. Prior work further showed that difficulty adjustment can interact with variable hash power and adversarial strategies to create unstable throughput or cyclic effects [6].

Defenses and mitigations

A diverse set of defense approaches have been proposed, including Fresh Bitcoins [5], Zeroblock [18], and Publish-or-Perish style mechanisms [23]. Alternative designs such as Fruitchains [12] and StrongChain [20] explore fairness and transparency in PoW consensus. In parallel, operational discussions in the Monero community have proposed pool-level detective mining and Publish-or-Perish-inspired mitigations [19, 21]. Finally, work on Monero mining-pool data publication shows that pool-published data can expose otherwise hidden information, illustrating why operational data can matter for incident validation [22].

3 Data Collection and Qubic Block Attribution

This section describes the data sources and processing steps used in the analysis. The dataset and analysis code are available as supplementary material.

3.1 Block and mining information collection

We combine three data sources: Monero block data from nodes, mining job data from Qubic pool job-notification traffic, and Qubic-related block data collected by the Monero community from Qubic network traffic.

Monero block information

We operated a pruned Monero full node, which validates the blockchain while retaining only a subset of prunable transaction data to reduce storage. This node let us collect main-chain blocks and locally observable orphan blocks that are not available from public explorers. We operated it from September 29 to October 17, 2025. For earlier periods, we queried public Monero full nodes via RPC for historical block and coinbase data. To improve orphan coverage, we probed 20 public nodes, selected the five most responsive nodes, and periodically synchronized their orphan observations. Public-node collection still does not guarantee complete orphan coverage, and some pool-specific orphan blocks may remain unobserved.

Qubic mining pool

Qubic distinguishes consensus nodes called *Computors* from miners that receive tasks and return solutions [13]. Our observations cover the public pool API and community-collected job and solution traffic, not Qubic's full internal network. Because we initiated this study after the campaign was already underway, the job-notification data cover its later phase, from September 26 to October 17, 2025. During this period, we queried the RPC API of the Qubic mining pool at 5-second intervals. The API has a structure similar to the Stratum protocol, which is widely used by PoW mining pools, and is openly accessible. Specifically, the `job_notify` method returns multiple values, including the mining block height and the previous block hash, enabling miners to obtain the most recent mining jobs. These records show which block tip Qubic instructed miners to work on.

Monero community observations

We also obtained an additional dataset of Qubic-related blocks from Monero community users Sergei Chernykh and DataHoarder, who collected it by monitoring Qubic-related activity. We refer to these records as *community-observed Qubic blocks*. The community used multiple observation channels, including Qubic network traffic and related artifacts, but we do not treat the dataset as a complete Qubic-provided ground truth. We use it as an independent view of Qubic-related blocks, especially orphan blocks that may be missed by Monero-node observation. Merging the community and node-derived datasets by block hash adds 93 community-observed orphan blocks that are absent from the node-derived block table, suggesting limited propagation.

3.2 Qubic block attribution

View-key verification

Monero transaction outputs, including coinbase rewards, use one-time addresses that cannot be publicly linked to the recipient's address. Blocks therefore do not reveal which miner controls a reward output. A disclosed view key allows the corresponding reward outputs to be identified retrospectively. Qubic disclosed its view keys only after each weekly epoch ended, so they could not support real-time attribution. This verification was also limited for periods before our node began operating because we could not fully collect historical orphan-block data. The full version lists the disclosed view keys used in this study.

Initial heuristic attribution

Before obtaining the community dataset and without using the view keys disclosed by Qubic, we constructed an initial seed set from orphan-fork blocks attributed to Qubic. We examined the extra-nonce area of their coinbase transactions and iteratively refined two structural regex patterns to identify additional candidate blocks in the node data. These patterns extended attribution coverage but were not treated as independent proof of ownership. The full version provides representative examples and the two patterns.

Cross-source validation and final labels

We compared the initial labels with the community dataset over the overlapping observation period. Of the 13,000 initially attributed Qubic blocks, 12,989 also appear in the community dataset, corresponding to a 99.92% confirmation rate among the initial positive labels. We observe no disagreement in chain or orphan status among the matched blocks. We retain the remaining 11 node-observed orphan candidates because each matches a Qubic extra-nonce pattern and shares the relevant prefix with a verified Qubic main-chain block at the same height. Their coinbase data were not preserved, however, so they cannot be verified directly with a view key.

The community dataset contains 13,771 Qubic blocks in the study period, 782 of which were missed by our initial heuristic attribution. Of these, 689 were present but unlabeled in our pruned-node dataset, while 93 were absent from it. We relabel the former and add the latter community-only orphan blocks. The resulting union contains 58,944 observed blocks, of which 13,782 are attributed to Qubic. Our final labels therefore follow the community-observed hash set, supplemented by the 11 node-observed orphan candidates.

Coverage and parent-block checks

The 93 community-only orphans were observable in the Qubic network and verifiable with disclosed view keys, but absent from observations from our Monero node. Excluding them changes the reported quantities only slightly and does not alter our conclusions. Separately, for community-observed Qubic orphans with raw block blobs, we parse the Monero header `prev_id` and check whether the parent hash appears in either dataset. This identifies the parent for 1,218 of 1,228 such orphans overall and for 1,213 of 1,216 within our observation window. Because the global node-observed table does not preserve raw blobs or parent hashes for every non-Qubic orphan, this check does not establish complete parent coverage for all observed forks. The full version reports the detailed results.

4 Qubic's Mining Share and Network Impact

This section examines Qubic's mining activity on Monero during the measurement period. We first quantify its mining power based on attributed blocks, and then examine its impact on orphan blocks and reorganizations. A focused analysis of Qubic's selfish mining strategy follows in Section 5.

4.1 Observed mining share

Figure 1(a) shows Qubic's observed block-production share in the Monero network, computed as the ratio of Qubic-attributed blocks to all observed main-chain and orphan blocks over weekly, daily, and hourly windows. Because the pool's physical hashrate is difficult both to

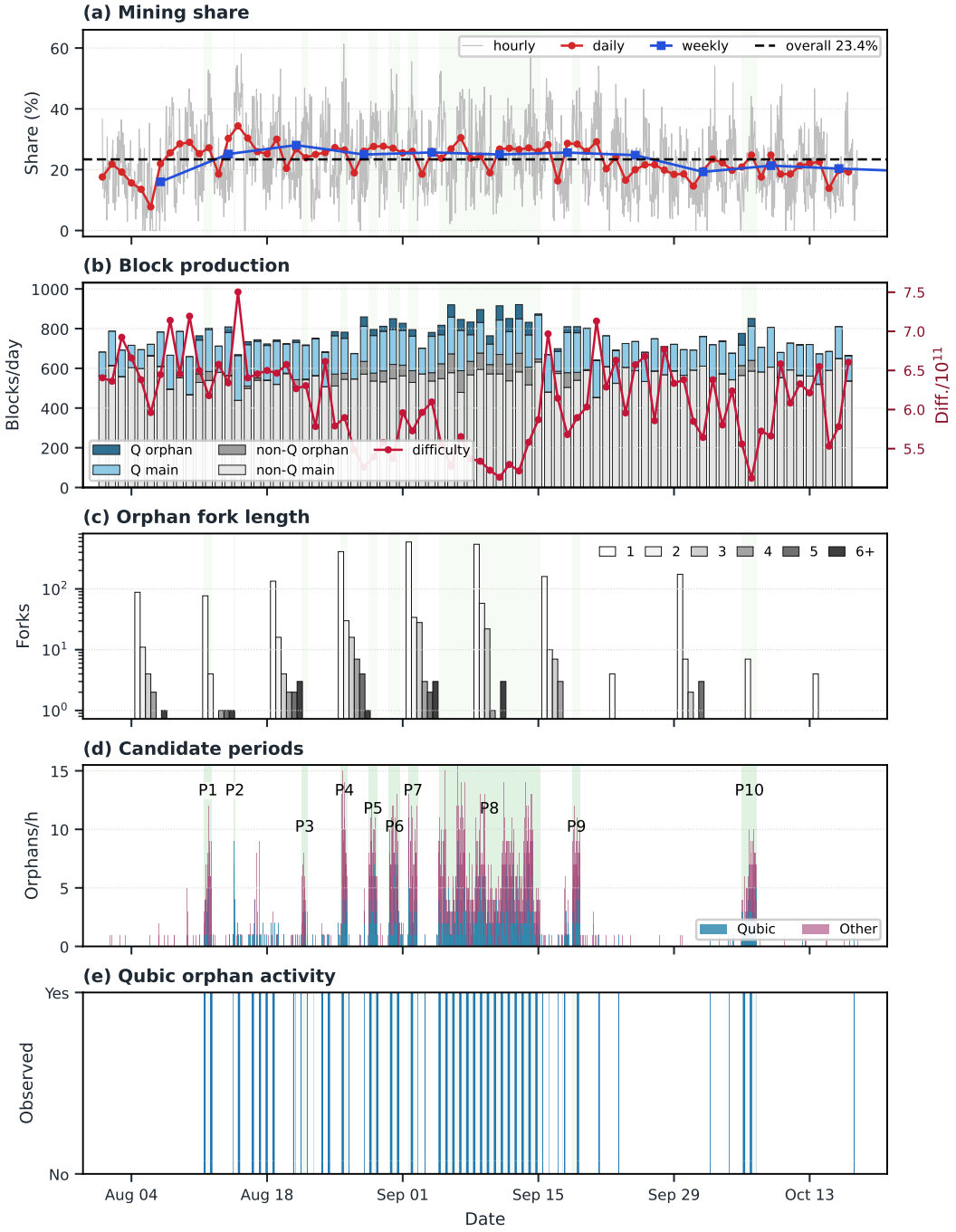


Figure 1 Overview of Qubic's mining activity and selfish mining indicators on a common timeline. Panels show (a) Qubic's observed block-production share, (b) daily block production and average difficulty, (c) weekly orphan-fork length distribution, (d) hourly orphan counts and candidate selfish mining periods, and (e) hours containing observed Qubic orphan blocks. Lightly shaded bands denote the ten candidate periods used in the selfish mining analysis.

measure directly and to reconstruct retrospectively, we use this quantity as an empirical estimate of its relative hashrate α in our selfish mining models. This calculation uses the combined attribution set described in Section 3.2, including community-observed Qubic labels. Across the measurement period, the resulting estimate averages 23.38%.

A central point in public discussions was Qubic's claim, echoed by several media outlets, that it had achieved a 51% mining position on Monero [15]. However, the notion of a "51% attack" is often left ambiguous, and our measurements do not support a persistent majority. In the hourly series, we observe several short intervals where Qubic's share approaches or briefly exceeds 50%. In contrast, the corresponding daily and weekly aggregates never reach 51%, and Qubic does not maintain a stable majority at any point in our dataset. These results indicate that while Qubic temporarily concentrated substantial mining power, it did not achieve the sustained control typically associated with a practical 51% attack on the Monero network.

4.2 Orphaning and reorganizations

Although Qubic did not sustain majority mining power, periods of elevated Qubic activity coincided with a marked increase in orphaning and deeper forks on Monero.

Figure 1(b) presents the daily counts of main-chain and orphan blocks, separated into Qubic-attributed and non-Qubic blocks, alongside the average network difficulty. During periods of elevated Qubic activity, both Qubic and non-Qubic orphan counts increase markedly. The resulting forks contain blocks from other miners as well as Qubic's own blocks that failed to enter the main chain.

Among the cases for which raw block blobs are available, we directly verify eleven heights at which at least two distinct blocks attributed to Qubic share the same parent. At nine heights, one Qubic block entered the main chain and another was orphaned. At two additional heights in P2, both Qubic sibling blocks were orphaned in favor of a non-Qubic block. Nine of the eleven cases occurred between August 14 and 17, including six in P2. Mining a second block on the same parent does not extend a private chain, so this behavior wastes the pool's own work and is inconsistent with a rational mining strategy. We suspect that these events resulted from an early job coordination failure in Qubic's selfish mining operation. Because our job telemetry begins later, we cannot directly confirm the internal cause.

Figure 1(c) shows the distribution of orphan fork lengths over time. Outside periods of elevated Qubic activity, orphan forks are almost exclusively of length one. During high-activity periods, the distribution shifts toward longer orphan chains and more frequent multi-block forks. This pattern shows that the campaign coincided with deeper reorganizations rather than only more isolated one-block orphans.

5 Qubic's Selfish Mining Strategy and Revenue

We examine how Qubic's selfish mining behavior varied over time and compare its observed mining revenue with model predictions and honest mining baselines.

5.1 Selfish mining strategies

Selfish mining, introduced by Eyal and Sirer [2] and later optimized by Sapirshtein et al. [16], describes how a rational miner (or pool) withholds blocks and selectively publishes a private chain to seek a revenue share exceeding its relative hashrate. We use the original Eyal–Sirer strategy as our standard selfish mining model. In practice, miners face uncertainty due to

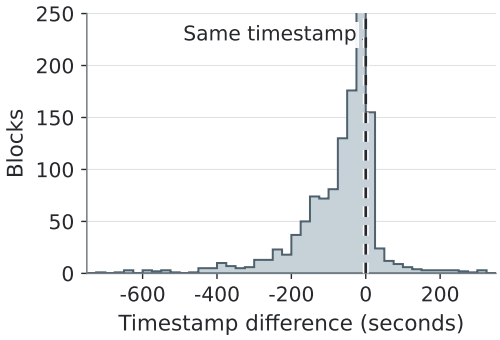


Figure 2 Qubic blocks' timestamp difference distribution with same-height competing blocks, including orphan blocks.

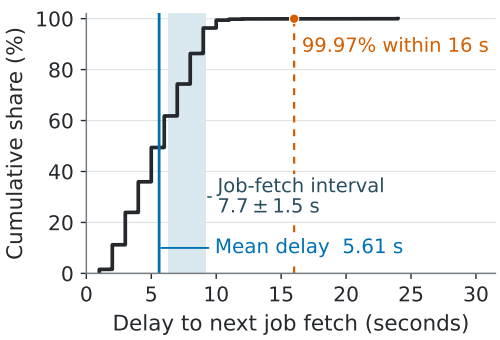


Figure 3 Empirical CDF of the delay between Qubic block timestamps and the first subsequent job-fetch record that references the block. The mean delay is 5.61 seconds. 86.3% of matched blocks fall within 8 seconds and 99.97% within 16 seconds.

network asynchrony and incomplete information, and may adapt their withholding and release rules accordingly. Qubic's observed behavior suggests that its withholding and release rules differed from the standard strategy, although its exact internal strategy remains unobservable.

We use the following terminology throughout the analysis. The attacker's relative hashrate is denoted by α. A *private chain* is a sequence of valid blocks that the attacker has mined but has not yet broadcast. The *lead* is the number of blocks by which this private chain is ahead of the public chain. For example, if the public chain is at height 100 and the attacker privately holds blocks up to height 102, the lead is 2. In the state-machine figures below, state i means that the attacker has lead i. State 0 means there is no private lead. State $0'$ is the tie-breaking event that occurs when the attacker publishes a withheld block at the same height as an honest block. The parameter γ is defined only for this state $0'$ tie-breaking event: it is the probability that honest miners who find the next block extend the attacker's branch rather than the competing honest branch. Thus, γ is not an orphan rate and is not estimated from arbitrary non-tip orphan blocks.

Because Qubic's internal block-discovery times and private-chain states are unavailable, we infer its behavior from Monero main-chain and orphan data, Qubic attribution labels, and block timestamps. We first assess whether the timestamps can approximate event ordering, then use orphan activity to identify sustained selfish mining periods.

5.2 Reliability of Qubic block timestamps

We assess whether Qubic's block timestamps approximate discovery and release timing. To do so independently of the fork patterns analyzed below, we measure the delay between each timestamp and the first subsequent job-fetch record that references the block as its predecessor.

The mining client is configured to fetch new jobs approximately every 7 seconds, and the observed average fetch interval in our logs is 7.74 ± 1.46 seconds. The mean delay is 5.61 seconds, with 86.3% of matched blocks falling within 8 seconds and 99.97% within 16 seconds (Fig. 3). This delay distribution is consistent with timestamps being generated within Qubic's normal mining workflow. Although the comparison cannot rule out selective manipulation,

we find no evidence of systematic timestamp shifts large enough to invalidate relative temporal comparisons. Additionally, community observers who monitored Qubic-network traffic reported no indications of systematic timestamp manipulation.[1]

Using these timestamps, Fig. 2 shows that most Qubic blocks competing at the same height have timestamps earlier than or comparable to those of their competitors. This timing pattern supports early block discovery followed by selective withholding. The small number of unusually late timestamps occurs predominantly among orphaned Qubic blocks.

We therefore use Qubic's timestamps, together with orphan dynamics, as approximate indicators of relative event ordering and selfish mining activity. This interpretation assumes that Qubic did not systematically alter timestamps to conceal private-chain lead lengths or release decisions.

5.3 Periods of sustained selfish mining activity

Qubic's strategy may vary over time, and selfish mining, if present, is unlikely to be applied uniformly across the entire observation window. To separate periods of sustained abnormal orphan activity from the rest of the observation window, we use the heuristic in Alg. 1. We treat such activity as an indicator of potential selfish mining.

Algorithm 1 Selfish Mining Period Heuristic.

Require: Blocks B with timestamps and orphan flags, thresholds $\tau_{\min}$, $d_{\min}$, $g_{\max}$
Ensure: Valid period spans M
Step 1: Construct the complete hourly series
$H \leftarrow$ complete hourly range covered by B
for all $h \in H$ **do**
    $C[h] \leftarrow |\{b \in B : b.\text{is\_orphan} \wedge \lfloor b.\text{timestamp} \rfloor_{\text{hour}} = h\}|$
end for
Step 2: Find maximal contiguous segments
$S \leftarrow$ maximal segments of $\{h \in H : C[h] \geq \tau_{\min}\}$
$S \leftarrow \{(s_i, e_i) \in S : |e_i - s_i| \geq d_{\min}\}$
Step 3: Merge segments with gaps $\leq g_{\max}$ hours
$M \leftarrow \text{merge}(S, g_{\max})$
return M

The heuristic operates in three steps. First, we construct a complete hourly series over the observation window and compute the orphan count $C[h]$ for each hour, assigning zero to hours with no observed orphan blocks. Second, we identify maximal contiguous segments in which every hour satisfies a minimum orphan-count threshold $\tau_{\min}$ and the total segment length exceeds a minimum duration $d_{\min}$. Third, we merge neighboring segments separated by gaps shorter than $g_{\max}$ hours to tolerate brief fluctuations in activity. The result is a set M of candidate selfish mining periods.

For Fig. 4, we distinguish fork runs involving Qubic and other miners from Qubic self forks in which two blocks attributed to Qubic share the same parent and height. The latter do not identify a private chain lead and are excluded from inference about Qubic's release strategy.

[1] Personal communication with Monero community observers.

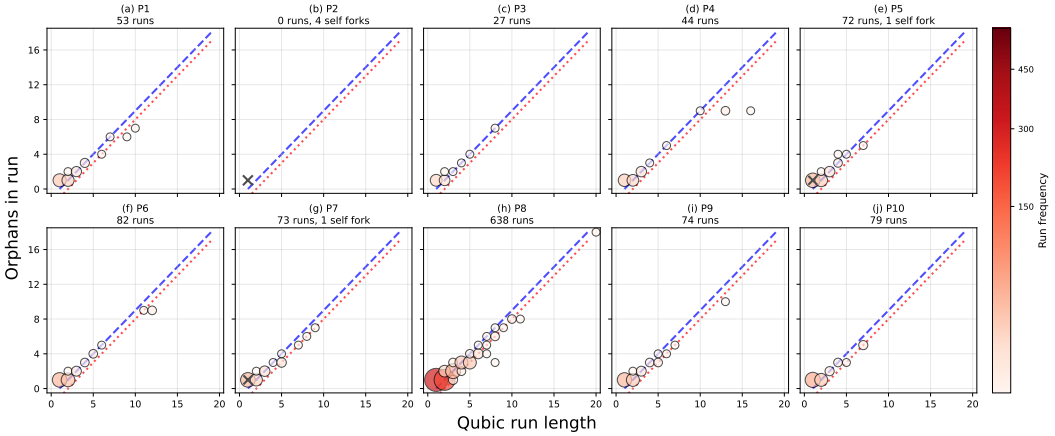


Figure 4 Distribution of orphan blocks per Qubic run length across ten candidate periods, shown in panels (a) through (j). Red circles visualize the frequency of fork runs involving Qubic and other miners. Gray crosses mark Qubic self forks associated with an accepted Qubic block, which are excluded from inference about Qubic's release strategy because they do not identify a private chain lead. Same-parent Qubic pairs in which both blocks were orphaned fall outside this run-based plot. Reference lines show $y = x - 1$ (blue dashed) and $y = x - 2$ (red dotted).

In our evaluation, we set $\tau_{\min} = 2$ orphan blocks per hour, $d_{\min} = 4$ hours, and $g_{\max} = 6$ hours. These settings capture sustained orphan activity while preventing short gaps from splitting a continuous span of activity into multiple periods. Our heuristic captures sustained, observable phases of Qubic's selfish mining operation and may miss brief or sporadic activity. The resulting P1–P10 candidate periods form the basis of the focused analyses that follow. To check robustness to the threshold choices, we repeat the analysis over all 80 parameter combinations. In every setting, Qubic's target-rate-normalized yield remains below the corresponding estimated α, so the aggregate underperformance is not specific to our default parameters. The full version provides the detailed sensitivity analysis, and the artifact includes the full grid.

Figure 1(d) shows the resulting candidate periods on the timeline, while Fig. 1(e) shows the hours in which Qubic orphan blocks were observed. Applying this heuristic yields ten periods (P1–P10) during which Qubic's block share and involvement in orphan blocks are noticeably higher than their global averages. Qubic's estimated share averages 28.33% across these periods, compared with 23.38% over the full measurement window. This association suggests that sustained, observable selfish mining activity was more common when Qubic's estimated share was relatively high.

Figure 4 further illustrates how Qubic's behavior varies across the identified periods by plotting the lengths of runs controlled by Qubic against the number of associated orphan blocks. In P1, P3, and P4, most runs that provide release information lie close to the blue line ($y = x - 1$), corresponding to release at lead 1. P2 instead contains four plotted Qubic self forks, two additional same-parent Qubic pairs that were both orphaned, and no run that provides release information.

During P8, where Qubic's activity is most intensive, we observe many runs aligned with patterns indicative of release at lead 2, suggesting a more conservative release strategy that avoids exposing the pool to tie situations when the lead is only one block. This shift supports the view that Qubic adapted its release rule over time, motivating the need for an analytical model that captures this conservative release behavior.

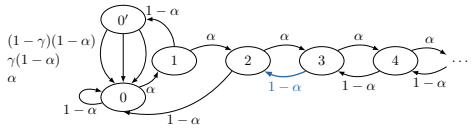


Figure 5 Standard selfish mining state machine. State i is the attacker's private lead, and state $0'$ denotes the tie-breaking event.

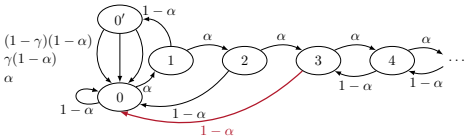


Figure 6 State machine for the conservative strategy. At lead 3, an honest discovery triggers full publication and a return to state 0.

5.4 Standard and conservative selfish mining models

To interpret the observed periods, we compare them against analytical models of selfish mining. We first recall the standard selfish mining revenue function $R_{\text{selfish}}(\alpha, \gamma)$, which gives the pool's expected main-chain revenue share as a function of its relative hashrate α and the tie-breaking parameter γ. Figure 5 shows the corresponding state machine. When the selfish miner has a private lead, it keeps mining privately. When the honest network catches up, the selfish miner publishes withheld blocks to create or win a race. In state $0'$, the race can resolve in three ways. An honest miner may extend the honest branch with probability $(1-\alpha)(1-\gamma)$, an honest miner may extend the selfish branch with probability $(1-\alpha)\gamma$, or the selfish miner may find the next block with probability α. Equation 1 gives the Eyal–Sirer revenue expression [2].

$$R_{\text{selfish}}(\alpha, \gamma) = \frac{\alpha(1-\alpha)^2(4\alpha + \gamma(1-2\alpha)) - \alpha^3}{1 - \alpha(1 + (2-\alpha)\alpha)} \tag{1}$$

We then consider a conservative-release variant of selfish mining, which we call the conservative strategy. The conservative strategy follows the Eyal–Sirer state transitions except at private lead 3, as shown in Fig. 6. If the honest network finds a block in that state, the attacker fully publishes its three-block private chain and returns to state 0 instead of retaining a private lead. This additional transition represents a more conservative release decision motivated by Qubic's observed behavior. We compare Qubic's observations with both models without assuming that either model captures its exact time-varying strategy.

Using this model, we derive a closed-form expression for the expected revenue under the conservative strategy as a function of α and γ.

▶ **Proposition 1.** *Let $R_{\text{mod}}(\alpha, \gamma)$ denote the selfish pool's long-run fraction of accepted blocks under the conservative strategy encoded by the state machine in Fig. 6, where $\alpha \in (0, \frac{1}{2})$ is the pool's relative hashrate and $\gamma \in [0, 1]$ is the tie-breaking parameter. Then*

$$R_{\text{mod}}(\alpha, \gamma) = \frac{\alpha\big(-2\alpha^3\gamma + 3\alpha^3 + 5\alpha^2\gamma - 9\alpha^2 - 4\alpha\gamma + 4\alpha + \gamma\big)}{1 - \alpha - 2\alpha^2 + \alpha^3 - \alpha^4}. \tag{2}$$

The proof is given in the full version.

5.5 Model predictions and observed revenue

We now apply the analytical model using parameters inferred from the identified selfish mining periods. For each period, we estimate Qubic's relative hashrate α from its share of main-chain and orphaned blocks. We estimate γ only from resolved state $0'$ tie-breaking events: cases where exactly one Qubic block competes with a block attributed to another miner at the same height and the next main-chain block reveals which branch the network extended. If the next block is mined by an honest miner and extends Qubic's branch, we

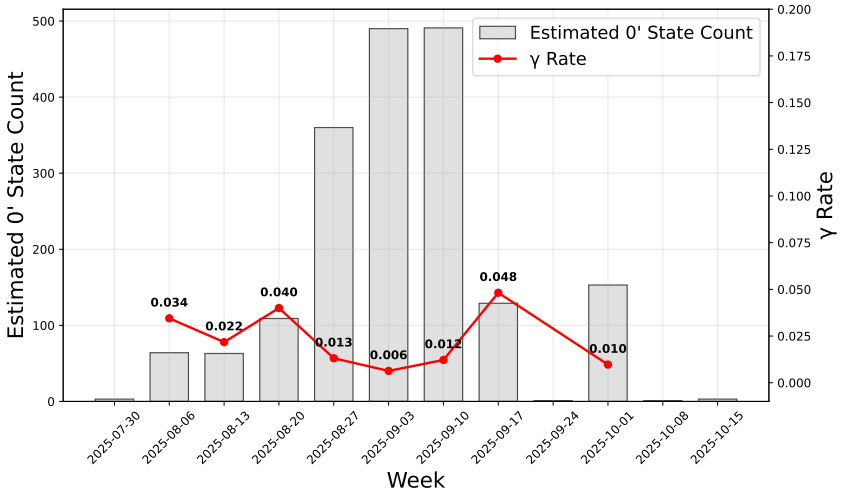


Figure 7 Weekly estimates of the tie-breaking parameter γ and the number of observed state $0'$ tie-breaking events. We use only resolved races containing exactly one Qubic block and a competing block attributed to another miner. Heights containing multiple Qubic blocks and non-tip orphan blocks are excluded because they do not reveal a binary tie-breaking choice.

count it as a γ success. If it extends the honest branch, we count it as a failure. We exclude non-tip orphan blocks from this calculation, because they may be discovered after the race has already been resolved and therefore do not measure tie-breaking behavior. Figure 7 shows that the weekly estimates of γ are close to zero. We attribute this to delayed propagation because Qubic blocks tend to arrive at nodes after competing blocks despite having earlier timestamps.

At the average hashrate share of 28.33% during the identified selfish mining periods and $\gamma \approx 0$, the standard selfish mining model yields an expected revenue ratio of $R_{\text{selfish}} \approx 24.56\%$, while the conservative model yields $R_{\text{mod}} \approx 23.71\%$. Both are below the honest mining baseline at the same α. Qubic's P1–P10 main-chain share is 25.75%, also below the honest mining baseline. The same relation holds without P8, with a main-chain share of 29.88% against an estimated α of 31.53%. Separately, its share of the blocks expected at Monero's target rate is 25.11%, a 3.22 percentage-point shortfall from α that measures the active-period economic opportunity cost. The net contribution of transaction fees was below 3% of realized block rewards and did not materially affect our conclusions.

The period-level points, however, do not align cleanly with either reference-model prediction. Some periods lie closer to the standard model, some are closer to the conservative model, and P1 and P3 outperform the honest baseline despite the aggregate underperformance. The full version reports the detailed period values. The next section examines observed tie-breaking outcomes as one direct reason why period-level shares can depart from the corresponding $\gamma \approx 0$ model predictions.

6 Race Outcomes and Limits of Static Models

The preceding comparison treats the standard and conservative models as fixed reference strategies. We next examine Qubic's outcomes in state $0'$ tie-breaking events as one directly observable source of deviation.

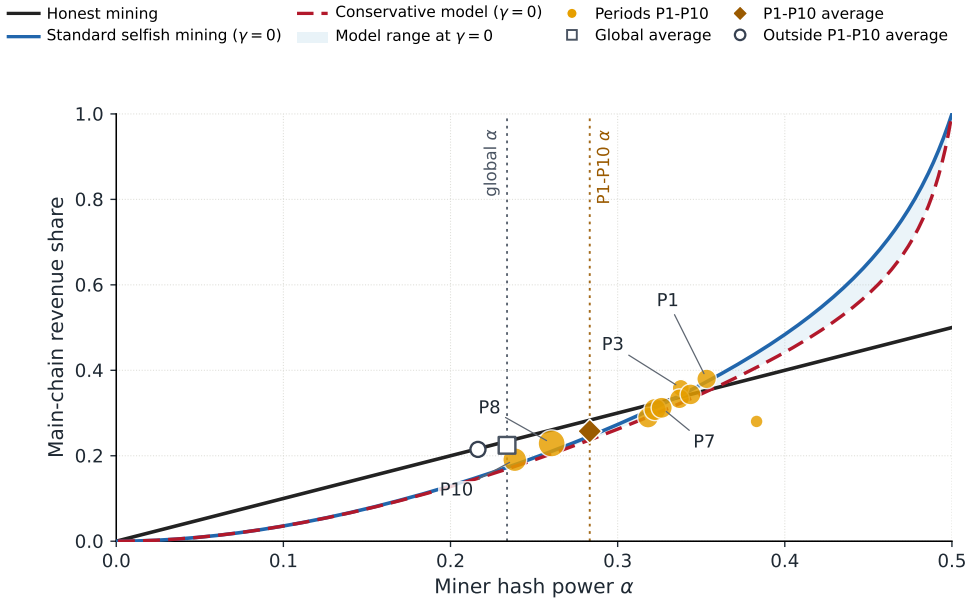


Figure 8 Theoretical main-chain revenue shares predicted by the standard and conservative models at $\gamma \approx 0$, compared with Qubic's observed main-chain shares. The shaded area shows the revenue range obtained by mixing the two fixed strategies in different proportions. Period markers show P1–P10, with marker size proportional to period scale. The filled diamond shows the P1–P10 average, while hollow markers show the global average and the average outside P1–P10.

The reference-model predictions use the global estimate $\gamma \approx 0$. Local race outcomes can nevertheless differ across periods, and Qubic sometimes won observed state $0'$ tie-breaking events at rates above its period-level mining share. These are cases where exactly one Qubic block competes with a block attributed to another miner at the same height and the next accepted block reveals which branch the network extended. We exclude heights containing multiple Qubic blocks because these self forks do not represent a binary tie-breaking event.

As illustrated in Fig. 9, Qubic's winning rate in these race conditions was higher than its period-level estimated share α in eight of the ten periods. For example, P1 has an observed race winning rate of 0.49 against $\alpha = 0.35$, P3 has 0.60 against $\alpha = 0.34$, and P7 has 0.40 against $\alpha = 0.33$.

By contrast, P2 recorded no Qubic-branch wins (0/14) and contained six parent-linked Qubic self forks within roughly 95 minutes, suggesting temporary operational instability. P4 also remained below α. More broadly, these results show that period-average α does not fully capture the local conditions under which Qubic released withheld blocks.

These local advantages can help explain why some observed main-chain shares exceed the corresponding $\gamma \approx 0$ model predictions. They do not explain the aggregate target-rate shortfall or establish consistently profitable execution. Race counts are limited in some periods, and high win rates can coexist with orphan losses and missed honest mining rewards. The next section therefore examines whether lower difficulty following the identified periods increased Qubic's rewards during the subsequent gaps and offset its active-period losses.

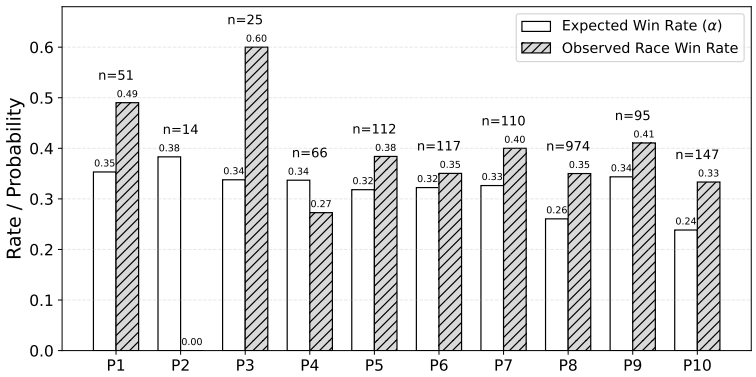


Figure 9 Qubic's winning ratio in observed state $0'$ tie-breaking events. The reference win probability is $\alpha + (1-\alpha)\gamma$, which reduces to the period-level block share α under the global estimate $\gamma \approx 0$. The observed rate counts events in which the next main-chain block extends Qubic's branch. Heights containing multiple Qubic blocks are excluded.

7 Intermittent Mining and Difficulty-Adjustment Spillovers

Qubic earned fewer blocks than the honest mining baseline in most of the identified selfish mining periods. Evaluating only these periods may miss a delayed benefit because slower main-chain growth can lower Monero's sliding-window difficulty in the intervals that follow. Intermittent selfish mining has been studied as a difficulty-adjustment phenomenon [11]. To capture both the immediate and delayed effects, we pair each identified period P_k with its following gap G_k and analyze them as a single accounting sequence.

7.1 Accounting model

Each G_k begins after P_k and ends when the next identified period starts. For each P_k, we define the signed balance relative to honest mining as

$$\Delta_{\text{active}}(P_k) = B_Q^{\text{main}}(P_k) - \alpha_k N_{\text{target}}(P_k), \tag{3}$$

where $N_{\text{target}}(P_k)$ is the number of blocks expected at Monero's target rate. A negative value indicates that Qubic obtained fewer accepted blocks than expected from honest mining at its estimated hashrate. This target-rate accounting expresses the period's effect in block-equivalents, allowing direct comparison with the subsequent DAA effect.

7.2 Estimating delayed effects on gap rewards

If selfish mining reduces main-chain production during P_k, Monero's sliding-window DAA can leave the following gap with lower difficulty. To estimate this delayed effect, we add synthetic main-chain slots to restore target-rate production during each P_k, then replay Monero's sliding-window DAA over the reconstructed timeline. This replay estimates the difficulty path that would have followed if main-chain production during P_k had remained at the target rate. We then compare the observed difficulty at each Qubic-attributed accepted block in G_k with the corresponding replayed difficulty. The full version discusses the time-scale implications of Monero's sliding-window DAA.

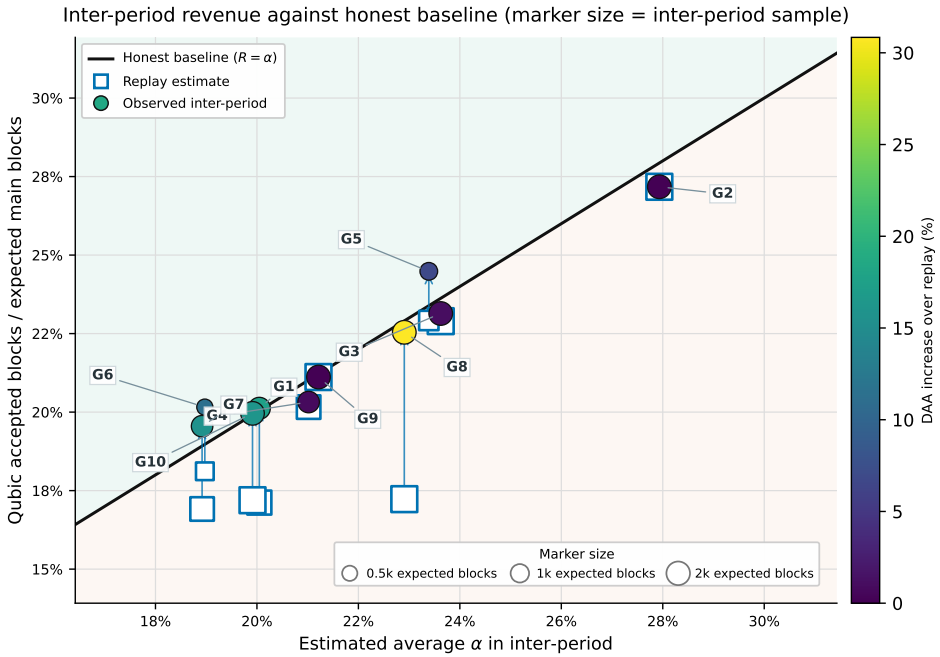


Figure 10 Qubic's gap revenue under the observed and replayed difficulty paths. Each G_k denotes the gap after identified period P_k. The black line is the honest mining baseline $R = \alpha$. Hollow squares show the replay estimate, filled circles show observed gap revenue, and vertical arrows show the DAA spillover. Marker size encodes gap sample size, capped at the 2k expected-block guide for readability.

Let D_i be the observed difficulty at a Qubic-attributed accepted block in G_k, and let $\widehat{D}_i$ be the corresponding replayed difficulty. The block-equivalent DAA spillover is

$$\Delta_{\mathrm{DAA}}(G_k) = \sum_{i \in Q(G_k)} \left(1 - \frac{D_i}{\widehat{D}_i}\right), \tag{4}$$

where $Q(G_k)$ is the set of Qubic-attributed accepted blocks in the gap. If the observed difficulty is lower than the replayed difficulty, the block contributes a positive fractional block-equivalent.

7.3 Empirical spillover

Figure 10 compares Qubic's observed gap rewards with the replay estimates. We measure the DAA effect as the increase from the replay estimate to the observed outcome. This comparison is distinct from whether the observed point lies above the honest mining line. Across all gaps, Qubic obtained 7,806 accepted blocks, while the replay yields 7,344.2 block-equivalents. The difference is +461.8 block-equivalents, or +6.3% relative to the replay estimate. Against a target-rate honest mining expectation of 7,855.3 blocks, the observed gap total remains 49.3 blocks lower. Summed across P1–P10, the period balance is −410.7 block-equivalents. Across the identified periods and subsequent gaps, Qubic obtained 11,045 accepted blocks against a target-rate honest mining expectation of 11,505.0, a shortfall of 460.0 block-equivalents, or 4.0%.

P8 and its following gap provide a representative example. During P8, Qubic's estimated hashrate was 26.05%. It obtained 1,700 accepted blocks, compared with an honest mining expectation of 1,961.5, giving $\Delta_{\mathrm{active}}(P_8) = -261.5$ block-equivalents. During G_8 (P8→P9),

Qubic obtained 541 accepted blocks, while the replay estimate was 413.5 block-equivalents. The resulting $\Delta_{\mathrm{DAA}}(G_8) = +127.5$ block-equivalents substantially raised gap rewards. However, the target-rate honest mining expectation for G_8 was 549.7 blocks, leaving the gap 8.7 blocks lower and the P8–G_8 pair 270.2 block-equivalents below the corresponding honest mining baselines.

The estimated DAA spillover increased Qubic's rewards in the later gaps. Even with this increase, its combined reward across the identified periods and subsequent gaps remained 4.0% below the target-rate honest mining baseline.

8 Community Monitoring and Qubic Countermeasures

The preceding sections analyzed Qubic's campaign on Monero quantitatively, using observed data and theoretical mining models. The campaign, however, was a real-world operation in which one blockchain project targeted another live network, with active communities on both sides. The Monero community responded quickly by monitoring Qubic's mining activity and relaying observations and reconstructed blocks to Monero nodes. Qubic, in turn, introduced countermeasures. This section examines how that interaction affected block observability and informs our interpretation of Qubic's observed release behavior.

8.1 Community monitoring and block relay

To detect Qubic's private mining activity, Monero community members deployed monitoring nodes across the network. By connecting to Qubic's low-level mining network from multiple observation nodes, collecting packet-level data, and analyzing local logs, these community observers reconstructed the timing of mining jobs and their corresponding solution messages on Qubic's *Computor* network.

Each job encodes a block template, a candidate block assembled for mining that specifies the header and the set of transactions. This allowed the community observers to track which template Qubic used at a given time and when the corresponding solution was observed. They checked the origin and integrity of the collected messages using the digital signatures of Qubic's dispatcher and *computor* nodes and matched each job to its corresponding solution. They then used the PoW hash in the Result field of each solution message to match the solution to a reconstructed Monero block. Reception-log timestamps provided the timing information.

When these records were sufficient to reconstruct a valid Monero block, community members attempted to propagate it before Qubic's intended release. The relay was not integrated with a protocol-level mechanism for coordinating Monero nodes, making it difficult to translate these efforts into a reliable chain-level response.

The processed records also helped the community identify Qubic-related blocks and assemble the dataset used in our analysis. We assessed the resulting dataset against observable block patterns and Qubic's later-disclosed view keys, obtaining the strong agreement reported in Section 3.2.

8.2 Qubic countermeasures

Qubic nevertheless appears to have regarded the possibility of early private-block exposure as an operational risk and soon introduced countermeasures.[2] First, to prevent leakage of mining-solution data, it introduced encryption for solution messages circulating on its *Computor* network. Initially, Qubic applied a simple XOR-based scheme. Because the messages contained fixed or readily inferable fields, we infer that the key could be recovered through known-plaintext analysis. Community observers also reported recovering the key. Qubic then changed its encryption method and keys multiple times, rotated keys regularly, and adjusted message-routing paths, making real-time observation increasingly difficult. These measures reduced the community's access to solution data, although the open *Computor*-network architecture continued to expose some activity to external observers. Over time, Qubic's network appeared to shift toward a more controlled and centralized configuration, with tighter control over task dissemination and increasing reliance on a small set of dispatching points.

As a more fundamental countermeasure, Qubic began withholding transactions from block templates shared on its network. The missing transactions were revealed only when Qubic later published or relayed the block, preventing observers from reconstructing the complete Monero block in real time. Once the missing transactions became available, community observers reconstructed past blocks and relayed them for retrospective verification. Because this reconstruction occurred only after disclosure, it did not provide a timely mechanism for influencing chain selection.

8.3 Operational impact and inference limits

This sequence shows that Qubic's operation was shaped by more than hashrate dynamics. The community's relay efforts were not part of a coordinated Monero protocol response, so their direct effect on chain selection or Qubic's realized rewards cannot be established. Qubic responded with encryption, routing changes, and withheld transactions. These countermeasures progressively reduced the observability of its later activity.

Observed orphan lengths and release patterns may therefore reflect a mixture of Qubic's intended strategy, network propagation, monitoring pressure, and Qubic's own countermeasures. Our analysis describes the behavior visible in the data rather than reconstructing Qubic's exact internal strategy. The reference strategies therefore serve as benchmarks for interpreting release behavior that changed over time, not as exact descriptions of individual forks.

9 Discussion

Economic interpretation

Qubic's campaign may not have been motivated solely by mining revenue. In the Goldfinger model, an attacker may benefit by undermining confidence in the target cryptocurrency [7]. Qubic's promotional framing and token-linked reward structure suggest incentives beyond Monero block rewards [14, 15]. However, daily CoinGecko prices do not indicate a lasting effect on Monero. From August 1 to October 17, XMR ranged from $235.58 to $343.44 and

[2] The operational details in this subsection are based primarily on personal communication with Monero community observers.

ended at \$293.15, 1.0% below its initial price. QUBIC ranged from USD 1.38×10^{-6} to USD 3.27×10^{-6} and ended at USD 1.47×10^{-6}, 43.2% below its initial price. Price data alone cannot determine whether the campaign produced broader benefits. The attention it brought to Qubic may itself have provided promotional value beyond mining rewards.

Mitigation implications

Freshness-aware chain-selection rules, including Publish-or-Perish-style mechanisms discussed after the campaign, could reduce the value of late withheld blocks [5, 18, 23, 21]. Their effectiveness depends on assumptions about timing and propagation. Detective mining could use information exposed by a public selfish mining pool to mine on its leading private branch [19, 8]. In practice, transaction withholding creates operational risks, and intentionally extending the attacker's branch raises social concerns. Both responses address specific tactics, while systems with modest and concentrated hash power remain exposed because of limits in decentralization, propagation, and observability.

10 Conclusion

Qubic's campaign neither sustained majority control nor consistently followed a textbook selfish mining strategy. It increased orphaning and reorganization depth. Nevertheless, across the identified periods and subsequent gaps, Qubic remained 4.0% below the target-rate honest mining baseline despite occasional tie-breaking advantages and later DAA spillovers. Community monitoring and Qubic's countermeasures further show that real-world campaigns evolve with changing observability and operational responses. Evaluating such campaigns requires considering their on-chain effects, underlying incentives, public claims, and ecosystem responses.

References

1 Christian Decker and Roger Wattenhofer. Information propagation in the Bitcoin network. In *13th IEEE International Conference on Peer-to-Peer Computing (P2P)*, pages 1–10. IEEE, September 2013. doi:10.1109/P2P.2013.6688704.

2 Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, pages 436–454. Springer, 2014. doi:10.1007/978-3-662-45472-5_28.

3 Julius Fechner, Balakrishnan Chandrasekaran, and Marc X. Makkes. Calibrating the performance and security of blockchains via information propagation delays: revisiting an old approach with a new perspective. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing (SAC '22)*, pages 282–289. ACM, 2022. doi:10.1145/3477314.3507003.

4 Chen Feng and Jianyu Niu. Selfish mining in ethereum. In *IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 1306–1316. IEEE, 2019. doi:10.1109/ICDCS.2019.00131.

5 Ethan Heilman. One weird trick to stop selfish miners: Fresh bitcoins, a solution for the honest miner (poster abstract). In *Financial Cryptography and Data Security*, pages 161–162. Springer, 2014. doi:10.1007/978-3-662-44774-1_12.

6 Dragos I. Ilie, Sam M. Werner, Iain Stewart, and William J. Knottenbelt. Unstable throughput: When the difficulty algorithm breaks. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–5. IEEE, 2021. doi:10.1109/ICBC51069.2021.9461086.

7 Joshua A. Kroll, Ian C. Davey, and Edward W. Felten. The economics of bitcoin mining, or bitcoin in the presence of adversaries. In *Workshop on the Economics of Information Security*, 2013. URL: https://econinfosec.org/archive/weis2013/papers/KrollDaveyFeltenWEIS2013.pdf.

8 Suhyeon Lee and Seungjoo Kim. Rethinking selfish mining under pooled mining. *ICT Express*, 9(3):356–361, 2023. doi:10.1016/j.icte.2022.03.003.

9 Sheng-Nan Li, Carlo Campajola, and Claudio J. Tessone. Statistical detection of selfish mining in proof-of-work blockchain systems. *Scientific Reports*, 14(1):6251, 2024. doi:10.1038/s41598-024-55348-3.

10 Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 305–320. IEEE, 2016. doi:10.1109/EUROSP.2016.32.

11 Kevin Alarcón Negy, Peter R. Rizun, and Emin Gün Sirer. Selfish mining re-examined. In Joseph Bonneau and Nadia Heninger, editors, *Financial Cryptography and Data Security*, pages 61–78, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-51280-4_5.

12 Rafael Pass and Elaine Shi. FruitChains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 315–324. ACM, 2017. doi:10.1145/3087801.3087809.

13 Qubic Docs. Node types in the qubic network. URL: https://docs.qubic.org/learn/nodes/.

14 retrodrive. Epoch 172 recap - the monero experiment. https://qubic.org/blog-detail/epoch-172-recap-the-monero-experiment, August 2025. Accessed: 2025-10-25.

15 retrodrive. Qubic performs 51% monero network takeover demonstration. https://qubic.org/blog-detail/historic-takeover-complete-qubic-miners-now-secure-monero-network, August 2025. Accessed: 2025-10-25.

16 Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in Bitcoin. In *Financial Cryptography and Data Security*, pages 515–532. Springer, 2017. doi:10.1007/978-3-662-54970-4_30.

17 Ruisheng Shi, Zhiyuan Peng, Lina Lan, Yulian Ge, Peng Liu, Qin Wang, and Juan Wang. Eclipse attacks on monero's peer-to-peer network. In *Network and Distributed System Security Symposium (NDSS 2025)*, 2025. doi:10.14722/ndss.2025.240095.

18 Siamak Solat and Maria Potop-Butucaru. Brief announcement: ZeroBlock: Timestamp-free prevention of block-withholding attack in Bitcoin. In *Stabilization, Safety, and Security of Distributed Systems*, pages 356–360. Springer, 2017. doi:10.1007/978-3-319-69084-1_25.

19 Riccardo Spagni. (github issue #140) implement detective mining (pool software change, no protocol change required). https://github.com/monero-project/research-lab/issues/140, August 2025. Accessed: 2025-10-25.

20 Pawel Szalachowski, Daniël Reijsbergen, Ivan Homoliak, and Siwei Sun. StrongChain: Transparent and collaborative Proof-of-Work consensus. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 819–836. USENIX Association, 2019. URL: https://www.usenix.org/conference/usenixsecurity19/presentation/szalachowski.

21 tevador. (github issue #144) selfish mining mitigations (publish or perish). https://github.com/monero-project/research-lab/issues/144, August 2025. Accessed: 2025-10-25.

22 Dimaz Ankaa Wijaya, Joseph K. Liu, Ron Steinfeld, and Dongxi Liu. Transparency or anonymity leak: Monero mining pools data publication. In *Information Security and Privacy (ACISP 2021)*, volume 13083 of *Lecture Notes in Computer Science*, pages 433–450. Springer, 2021. doi:10.1007/978-3-030-90567-5_22.

23 Ren Zhang and Bart Preneel. Publish or perish: A backward-compatible defense against selfish mining in Bitcoin. In *Topics in Cryptology – CT-RSA 2017*, pages 277–292. Springer, 2017. doi:10.1007/978-3-319-52153-4_16.

24 Roi Bar Zur, Ittay Eyal, and Aviv Tamar. Efficient MDP analysis for selfish-mining in blockchains. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies (AFT)*, pages 113–131. ACM, 2020. doi:10.1145/3419614.3423264.

When Does Being Selfish Pay Off? Temporal Composability and Profitability in Selfish Mining

Colin Finkbeiner
University of Connecticut, Storrs, CT, USA

Connor Shaw
University of Connecticut, Storrs, CT, USA

Ghada Almashaqbeh
University of Connecticut, Storrs, CT, USA

Abstract

Selfish mining undermines incentive compatibility of proof-of-work blockchains, letting a miner earn disproportionate rewards at a hashrate lower than the majority threshold. A decade of research has investigated *whether* a strategy is profitable, however, far less is understood about *when* it becomes profitable. Timing is critical since selfish mining operates at a loss before it turns a profit, typically requiring tens of weeks to break even in the classic case.

In this paper, we present a holistic study of the time-to-profitability (TTP) of existing selfish mining strategies structured around four contributions. First, in the single-attacker setting, we characterize TTP across the full strategy space and find that TTP-minimizing and profit-maximizing strategies frequently diverge, making attack horizon a critical metric. In particular, under realistic fee dynamics, the use of incentive transactions to recruit honest-but-rational miners enables reaching profitability up to 15× faster than classic selfish mining at the same hash rate. Second, we study TTP for the first time in the multi-attacker setting, showing that the difference in strategies between opposing attackers has a dramatic impact on join profitability. Third, we generalize intermittent selfish mining by examining temporal composition over the full strategy space and showing that its purported benefits are largely overstated; alternating strategies rarely outperforms the best static strategy in terms of TTP and long-term profits.

Finally, and building on these findings, we investigate adaptive, state-conditioned strategy selection at the difficulty adjustment period (DAP) level. We compare a general-purpose LLM agent against a fixed decision-tree selector. We find that both selectors reliably identify profit-maximizing strategies from observed network conditions, at a low operational cost, thus lowering the expertise barrier to exploiting adaptive selfish mining.

2012 ACM Subject Classification Security and privacy → Distributed systems security

Keywords and phrases Bitcoin, Proof of work, Selfish mining, Temporal mining strategies

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.29

Related Version *Full Version*: `https://eprint.iacr.org/2026/1629`

Supplementary Material *Software (Source Code)*: `https://github.com/CSSL-UConn/temp-sim` [12] archived at `swh:1:dir:71dbe9c9ecdc32669573f00c1ef9950caad6bd2f`

Funding This work is supported by NSF under grant No. CNS-2226932.

1 Introduction

Selfish mining poses a fundamental threat to incentive compatibility of proof-of-work blockchains such as Bitcoin. By selectively withholding mined blocks, and exploiting Bitcoin's fork resolution process and difficulty adjustment mechanisms, a miner can earn disproportionate rewards at a hashrate well below the majority threshold. In the decade since Eyal and Sirer's seminal result on classic selfish mining [10], a substantial body of work has emerged



© Colin Finkbeiner, Connor Shaw, and Ghada Almashaqbeh;
licensed under Creative Commons License CC-BY 4.0



and expanded our understanding of this threat [2, 3, 5, 6, 13, 14, 16–18, 20–22, 24, 25, 27, 28]. These showed new strategy variants that outperform classic selfish mining, explored optimal policies via analytical models and deep reinforcement learning (DRL), and studied richer model settings that cover propagation delay, multiple competing attackers, etc.

Most of these works focused on profitability in the long term. However, a related basic question that has received little attention is *when* selfish mining becomes profitable. This is a critical question since a selfish miner first has to waste its own blocks, to lower the difficulty target, thus operating at a loss before starting profit reaping. In fact, it has been estimated that classic selfish mining takes on the order of weeks of sustained attack before turning a profit [13]. For a rational attacker, this time-to-profitability (TTP) could be a vital factor in practice. When having a finite attack window, such an attacker may care less about asymptotic gains and more about how long it must absorb losses to collect profits.

Despite the large amount of work on selfish mining, existing studies around TTP are narrow. It has been studied almost exclusively for classic selfish mining and a handful of purpose-built strategies aiming to minimize this time. These are generally intermittent strategies alternating between honest and selfish mining [8, 19, 21], which like most of the selfish mining literature [11], have been studied within the single-attacker and block-reward-only model. To date, there is no *systematic TTP examination* across the strategy space, various reward models, or in the multi-attacker setting. Furthermore, beyond intermittent strategies, there are no studies on the more general case of *temporal strategy composition*, i.e., how alternating between distinct strategies over time may affect profitability and TTP.

Moreover, and as highlighted in [5, 14], optimal selfish mining depends on a set of dynamically changing factors – reward state, network conditions, and behavior of competing miners – that greatly complicate closed-form analysis. While these works have shown that DRL can identify near-optimal strategies, doing so requires per-environment training and DRL expertise, and operates at a block-by-block granularity. To our knowledge, no work has examined whether strategies can instead be adapted at a coarser macro-strategy level, i.e., per difficulty adjustment period (DAP), and whether an automated strategy selector can successfully respond to evolving blockchain state.

Open questions. Thus, we ask the following questions: *How does TTP vary across the strategy space, reward models, and the number of competing attackers? Also, does temporal strategy composition improve upon static strategy choice, and can an automated per-DAP selector effectively adapt to evolving network conditions?*

1.1 Contributions

In this paper, we answer these questions by presenting a holistic study of these issues. It covers TTP for existing selfish mining strategies in the single- and multi-attacker settings, as well as temporal composability of these strategies. We also examine how automated strategy selectors may assist in temporal composition by adaptively selecting strategies in response to evolving observed state. In particular, we make the following contributions.

TTP across the static single-attacker strategy space. We present the first systematic TTP analysis that, beyond classic selfish mining, covers lead-, trail-, fork-stubborn families [18], publish-N [16], and incentive-aware trailing [11]. We introduce a family of *value-aware* strategy variants generalizing improved selfish mining [6] across the strategies above, i.e., strategy variants that conditionally trigger publishing blocks based on their total transaction fees. The central finding is that TTP-minimizing and profit-maximizing strategies frequently

diverge, making TTP a leading metric for an attacker with a finite attack window. Moreover, under realistic fee dynamics, incentivized trailing strategies (where an attacker recruits honest-but-rational miners to build upon its chain) outperform both value-aware and value-agnostic strategies in terms of profitability.

We also characterize how TTP varies across reward models of increasing complexity: block-reward only, block-reward with transaction fees, and block reward with transaction fees and occasional high-value "whale" transactions. This is combined with having honest-but-rational miners who can be swayed by monetary incentives. Our findings are class conditional; while, naturally, non-incentivized strategies are largely insensitive to the fee model, incentivized strategies (relying on honest-but-rational miners) accelerate substantially with whale transaction frequency and magnitude.

TTP in the multi-attacker setting. We provide the first TTP analysis of two competing selfish miners, finding TTP to be less than 1 DAP in the case of value-aware incentive-trail strategies. Moreover, we introduce the *joint TTP* (JTTP) metric, which is the TTP analog of joint profitability. We show that the combination of strategies chosen by attackers has a considerable impact on JTTP. In particular, when attackers adopt strategies of similar type (i.e., incentive-driven, value-aware) JTTP is less than that of adopting different strategy types (for example, in some instances we find the opponent may never reach profitability). We also study TTP for different reward models and in the presence of honest-but-rational-miners, observing that the class-conditional behavior persists.

Temporal strategy composition. We generalize prior intermittent selfish mining to per-DAP composition over the full examined strategy space, rather than just alternating between honest and classic selfish mining. We evaluate this generalized temporal composability in the multi-attacker setting under the presence of whale transactions and rational-but-honest miners. The result is a negative one for the prevailing narrative around temporal strategies; across the space tested, and static network conditions, alternating rarely outperforms the best static strategy for TTP or asymptotic profit gains.

Adaptive strategy selection at the DAP level. We explore whether adaptive temporal composition of selfish mining strategies can be automated at a macro-strategy, per-DAP granularity. We compare two selector implementations: an LLM agent and a fixed rule-based decision tree, both following the same encoded strategy knowledge. We show that a per-DAP selector, with access to recent public blockchain state, can locate the dominant strategy from observable signals alone and reacts to network condition shifts. This trend holds for both agentic and rule-based selectors, and for the multi-attacker setting without knowledge of the opposing attacker's strategy. We believe that this is in part due to a single dominant strategy identified over much of the parameter space, as well as the encoded strategy knowledge mapping the state space to a clean set of strategy categories with distinct applicability conditions. Overall, the security implication is that adaptive selfish mining is operationally feasible in practice, accessible to miners without DRL expertise, and made even easier by using AI agents (compared to manually configuring a rule-based selector).

1.2 Related Work

TTP in selfish mining. The question of when selfish mining becomes profitable has received little attention in the literature. Grunspan et al. [13] showed that classic selfish mining requires on the order of weeks to become profitable. Subsequent TTP-aware works [8, 19, 21]

Table 1 Comparison of prior work on temporally composable selfish mining. † = Classic selfish mining, lead-/trail-/fork-stubborn families and their compositions, incentive-aware trailing strategies [11], value-aware strategies (generalizing Carlsten [6]), and Publish-N [16].

	Intermittent Selfish [19]	**Smart Int. Selfish** [21]	***L*-Stubborn Int.** [8]	**Undetectable Selfish** [2]	**Our Work**
Composition structure					
Strategy set composed over	{honest, classic selfish}	{honest, classic selfish}	{honest, *L*-stubborn}	{honest, selfish}	Full family†
Composition granularity	per-DAP	per-DAP & sub-DAP	per-DAP	per-block	per-DAP
Objective					
Profit-maximizing	✗	✓	✓	✓	✓
TTP minimizing	✓	✓	✓	✗	✓
Model setting					
Single attacker	✓	✓	✓	✓	✓
Multiple attackers	✗	✗	✗	✗	✓
Block-reward only	✓	✓	✓	✓	✓
Transaction fee model	✗	✗	✗	✗	✓
Whale transactions	✗	✗	✗	✗	✓
Honest-but-rational miners	✗	✗	✗	✗	✓
Automated / learned selection	✗	✗	✗	✗	✓

have been confined to the single-attacker and block-reward-only settings, while considering only binary alternation between selfish and honest mining. On the reward model side, Carlsten et al. [6] introduced improved selfish mining that exploits per-block fee variation. Bar et al. [5] introduced WeRLman that jointly models fees and whale transactions but only for the single-attacker setting and long-term profitability. In terms of model setting, multi-attacker selfish mining has been studied for asymptotic profitability (within a range of simulation-based and mathematical modeling approaches) [3, 4, 16, 17, 24, 27]. None of these works characterize TTP across the broader strategy space, richer reward models that account for transaction fees, or the multi-attacker setting where competitive interaction reshapes the realization of individual and joint attacker profitability. We aim to address this gap.

Temporal composability in selfish mining. A few works have studied strategies that vary over time. Table 1 summarizes these along three axes: the strategy set considered, decision granularity, and model setting. Negy et al. [19] introduced intermittent selfish mining, alternating between honest and selfish behavior for every DAP to exploit difficulty-target reductions. It achieves single-DAP profitability at the cost of a higher security threshold (0.37 vs. 0.33 of classic selfish mining based on the attacker connectivity setting). Sarenche et al. [21] introduced smart intermittent selfish mining, with the same strategy alternation but within DAPs, identifying a tradeoff between TTP and the security threshold. Doger et al. [8] extended this analysis with an L-stubborn intermittent variant that offers short-term gains, and interestingly showed that classic selfish mining still dominates in the long run. Undetectable selfish mining [2] can also be viewed as temporal, though attacker's decision around block publication are driven by detection-resistance rather than profitability.

All these temporal strategies compose over a binary set – honest paired with a single selfish variant – leaving the richer strategy space unexplored. Also, similar to above, they are evaluated only under the single-attacker, block-reward-only model and without considering honest-but-rational miners. We address these limitations in this paper.

Optimality in selfish mining. Optimality in selfish mining has been studied under various conditions [2, 7, 15, 16, 18, 20, 25]. Table 2 summarizes these prior efforts organized by the decision making mechanism they employ. Sapirshtein et al. [20] formulated selfish mining as a Markov Decision Process (MDP) and obtained an ϵ-optimal policy in the single-attacker,

Table 2 Comparison of optimal and near-optimal selfish mining works (✔ indicates the property is present; ✘ indicates it is not. POMDP = Partially-observable MDP).

	Optimal [20]	**Optimal MA** [3]	**L^*-Stubborn** [9]	**SquiRL** [14]	**WeRLman** [5]	**Our Work**
Decision model						
Method	MDP	POMDP	analytic	DRL	DRL	sim. study
Decision granularity	per-block	per-block	strategy-level	per-block	per-block	per-DAP
Generalizes across reward models without retraining	✘	✘	✔	✘	✘	✔
Objective						
Profit-maximizing	✔	✔	✔	✔	✔	✔
TTP minimizing	✘	✘	✘	✘	✘	✔
Attacker setting						
Single attacker	✔	✔	✔	✔	✔	✔
Multiple attackers	✘	✔	✘	✔	✘	✔
Reward model						
Block-reward-only	✔	✔	✔	✔	✔	✔
Transaction fee model	✘	✘	✘	✘	✔	✔
Whale transactions	✘	✘	✘	✘	✔	✔
Honest-but-rational miners	✘	✘	✘	✘	✘	✔

block-reward-only setting. Bai et al. [3] extended that to a partially-observable MDP for the multi-attacker case. To circumvent the intractability of large state spaces, i.e., being hard to be captured by a closed form analytical model, DRL has been employed in making optimal decisions. In particular, SquiRL [14] trained a DRL model that replicated [20] within 1% and conjectured that no profitable Nash equilibrium exists in the multi-attacker, block-reward-only setting. WeRLman [5] extended the model to incorporate occasional high-value whale transactions, showing that fee variability directly lowers the security threshold, which falls to ∼20% under frequent whale transactions. The work [9] took a different route with L^*-stubborn mining, operating at strategy granularity rather than per-block – a distinction shared by our work (although their goal is maximizing profitability of double spending).

All these approaches share two properties: they target asymptotic profitability rather than TTP, and require an offline optimization procedure – MDP solution, DRL training, or analytical derivation – tailored to a specific model setting. Altering the model (e.g., inclusion of whale transactions, honest-but-rational miners, or rival attackers) requires re-deriving or re-training the policy. Our LLM-based approach explores an alternative paradigm: rather than learning or solving for a policy offline, we use a general-purpose AI agent as a state-conditioned strategy selector at DAP boundaries informed with mining context. We preface that this is not a drop-in replacement to single-setting optimizations, but as a complementary direction that would be beneficial for navigating model variations.

2 Preliminaries

We offer an overview of Nakamoto consensus and selfish mining, then we present our threat and system models, followed by the strategies and the temporal composition we examine.

2.1 Background

Nakamoto consensus. We consider Nakamoto-style proof-of-work blockchains. Miners package transactions into blocks, solve a computational puzzle whose difficulty is calibrated by a difficulty adjustment mechanism, and extend the longest chain. A miner whose block is adopted by the longest chain earns a block reward and collects the fees of the transactions included in that block. In Bitcoin, the difficulty target is recomputed every $N = 2016$ blocks – a DAP, roughly two weeks – to hold the expected block time near 10 minutes.

Selfish mining. Eyal and Sirer [10] showed that honest mining is not always rational. A selfish miner who withholds newly mined blocks and selectively publishes them will be able to *orphan* honest miners' blocks, thus capturing a disproportionate revenue share. Difficulty adjustment is vital to such behavior; orphaned blocks reduce the rate of appending blocks to the longest-chain, which in turn reduces the next epoch's difficulty target and inflates future rewards. Thus, selfish mining is profitable when the attacker's share of total longest chain rewards is larger than the expected one based on its hashrate.

2.2 System and Threat Model

Network and miners. We model a blockchain network as a set of miners each controlling a fraction of the total network hashrate. Following the conventional model in the literature, block discovery can be viewed as a Poisson process, and the probability that a miner produces blocks depends on its hashrate. Miners in the system are classified into three classes:

- *Honest miners* follow the protocol as described. They mine on the longest chain known to them and publish blocks immediately upon discovery. In the case of a fork, they choose the longest branch, and when receiving multiple same-height blocks, they choose the block that was broadcast first. Honest miners' collective hashrate is denoted by $\beta \in [0, 1]$.
- *Selfish miners*, or simply attackers, deviate from the protocol to maximize their utility. An attacker i is characterized by a hashrate $\alpha_i \in [0, 1]$ (sometimes represented as a percentage), and a selfish mining strategy s_i. Selfish miners also share a connectivity parameter γ, defined below. In the single attacker setting, we denote an attacker's configuration as α, γ, s; in the multi-attacker setting, as γ and a tuple α_i, δ_i, s_i, where δ_i (also defined below) is attacker i's share of γ.
- *Honest-but-rational miners* are those that do not selfishly mine on their own, but their mining choices can be swayed by what maximizes their utility. That is, they can be incentivized to adopt a selfish miner's branch/block if this results in additional profits. Honest-but-rational miners are denoted by a hashrate fraction $\kappa \in [0, 1]$.

The hashrate fractions satisfy $\beta + \alpha + \kappa = 1$ in the single-attacker setting, and $\beta + \sum_i \alpha_i + \kappa = 1$ in the multi-attacker case.

An attacker has complete knowledge of the public chain and can withhold blocks, maintain a private chain, and release blocks strategically. In multi-attacker settings, attackers have partial information in the sense that they can observe each other's published blocks but not private chains or strategies. An attacker's objective could be: (1) maximizing profits, in which accumulated profits in the long term are being maximized; or (2) minimizing TTP, in which the time needed for a strategy to become profitable is being minimized. These objectives are not equivalent; a strategy that maximizes long-run revenue may absorb substantial losses at the beginning, while a strategy that minimizes TTP may forfeit some asymptotic gains.

Fork resolution. When an attacker releases a block that competes with an honest block(s) at the same height, the network is temporarily split. The connectivity parameter $\gamma \in [0, 1]$ determines the fraction of honest miners that mine on top of the attacker's block while the remainder mine on top of the honest block. In effect, $\gamma = 0$ models a poorly connected attacker whose blocks are always broadcast after honest miner's blocks, while $\gamma = 1$ models a well-connected attacker whose blocks always reach honest miners first. Similarly, in the multi-attacker case, when a fork contains blocks from multiple attackers, δ_i denotes the fraction of γ that mines on attacker i's block, where $\sum_i \delta_i = 1$. For simplicity, we assume δ_i is equally split across all attackers (e.g., for $i = 2$, we have $\delta_1 = \delta_2 = 0.5$).

Difficulty adjustment. At the end of each DAP (orphaned blocks do not count toward the N-block total), the target threshold is rescaled by the ratio of the expected time needed to produce N blocks to the actual time observed. The goal is to maintain the expected block production time constant. This rescaling is the mechanism that selfish mining exploits, so we use the DAP as our unit of time throughout the paper.

Reward model. We consider three reward models. First, block-reward only, where each block carries a fixed reward R. Second, block reward + transaction fees, where similar to [6], fees accrue continuously between blocks at a fixed rate. Specifically, fees accumulate proportionally to the time elapsed since the last block, with fee_T denoting expected fees in a block found after interval T. And third, existence of whale transactions, which is a sub-case of the second model in which blocks may contain rare, high-fee transactions. Such events occur with per-block probability p and yield a reward of $R + m \cdot \mathsf{fee}_\mathsf{T}$, where $m > 1$ is the magnitude multiplier over the baseline fee fee_T. Non-whale blocks have a reward of $R + \mathsf{fee}_\mathsf{T}$.

Measuring profitability and TTP. For profitability, we measure an attacker's *cumulative relative advantage*, cumRA, which is the percentage difference between the rewards captured by selfish mining and those captured by mining honestly. An attacker is profitable at a given point if its relative revenue exceeds its hashrate, i.e., $\mathsf{cumRA} > 0$ (or $\mathsf{cumRA_i} > 0$ for attacker i in the multi-attacker setting). Additionally, in the multi-attacker setting, we define joint profitability to be the case when all attackers are mutually profitable, i.e., all $\mathsf{cumRA_i} > 0$.

We define TTP as the earliest DAP after which $\mathsf{cumRA} > 0$ and remains so for the rest of the attack duration (similarly $\mathsf{TTP_i}$ in the multi-attacker case). We use $\mathsf{TTP} = \mathsf{undefined}$ when an attacker's cumRA does not exceed 0. Analogous to joint profitability, in the multi-attacker setting, we additionally define joint TTP (JTTP) as the earliest DAP after which all attackers are simultaneously profitable and remain so.

2.3 Examined Strategies

Base strategies. Honest mining is denoted as H, and classic selfish mining [10] is denoted as S. The stubborn family [18] generalizes S by being more aggressive in three distinct directions: L (stubborn-lead) continues withholding even when ahead of the public chain, thus gambling on extending the private lead further; F (stubborn-fork) keeps mining on the private chain when tied with the public chain rather than conceding; and T_n (stubborn-trailing) continues mining on the private chain even when lagging behind the public chain by up to n blocks. Lastly, publish-N [26], denoted as P_N, withholds blocks until the private branch has a lead of N blocks before releasing. We refer the reader to [11] for a detailed overview of these strategy families.

Stubborn mining composition. We use $\circ$ to denote (static) composition of base strategies. For example, $L \circ T_1 \circ F$ applies stubborn-lead, stubborn-trail-1, and stubborn-fork behaviors jointly. Composition is governed by the rules in [18] – base stubborn strategies govern mutually exclusive mining scenarios, lead (attacker ahead), trail (attacker behind), and fork (tie), thus at most one applies at any decision point making composition well defined.

Value-aware selfish mining. Carlsten et al. [6] introduced *improved selfish mining* in which an attacker is willing to publish a high-fee block early (instead of withholding and extending their private chain) that would otherwise be at risk of being claimed by honest miners. This captures the Bitcoin state when mining rewards decline and transaction fees become the dominant source of miners' income.

A *value-aware* strategy is a generalization of this idea applied to any selfish mining strategy (including composed and incentivized ones). Immediate block publication is triggered if the most recent block carries sufficiently large fees. Formally, let f_{in} be the actual total transaction fees of the most recently mined block on the private chain, and let $\mathbb{E}[f]$ be the expected total fees of a block. Value-aware override check fires when $f_{\text{in}} \geq \tau \cdot \mathbb{E}[f]$. In this case, the attacker immediately publishes its entire private chain, regardless of the strategy. The factor $\tau \geq 1$ quantifies the relative minimum fee magnitude to deviate from the base strategy's publication policy, so τ trades off the expected value of continued withholding against the guaranteed reward of immediate publication. Because inter-block times are exponentially distributed around a 10-minute mean, a fixed fee threshold would be miscalibrated; representing τ as a ratio instead enables the trigger to scale with fee accrual.

Incentivized trailing strategies. Recently, Finkbeiner et al. [11] demonstrated how the presence of honest-but-rational miners enables selfish miners to create a mutually beneficial scenario. Selfish miners inject additional fees in the network to attract rational miners, such that collecting these fees is conditional on adopting the selfish miner's chain. This was found specifically beneficial for stubborn-trail selfish mining; a selfish miner coming from a trailing position improves the chances of having honest miners mine on its block.

As in [11], we use $I(\eta, n)$ to denote incentivized trailing strategies, with an incentive magnitude η (expressed as a multiplier of the expected transaction fees fee_T) and stubborn-trail depth n. From a trailing position, the attacker posts $\eta \cdot \mathbb{E}[f]$ in fees conditional on its branch being extended. $I(\eta, n)$ is only meaningful when honest-but-rational miners are present ($\kappa > 0$) and the reward model includes transaction fees.

Notational examples. VA(1.5)$\circ L$ is value-aware stubborn-lead strategy with $\tau = 1.5$. While VA(1.5)$\circ I(2,1)\circ L$ is a value-aware ($\tau = 1.5$) incentivized ($\eta = 2$, trail depth 1) stubborn-lead strategy.

2.4 Temporal Strategy Composition

Here, an attacker need not commit to a single strategy for the duration of an attack. Prior work on intermittent selfish mining [8, 19, 21] alternates honest mining with a single selfish strategy – classic [19,21] or L-stubborn [8] – at DAP boundaries, exploiting the difficulty target drop from elevated orphan rate. We refer to this more generally as *temporal composition*; DAP-level selection from the strategies introduced in 2.3, of which honest-selfish alternation is one special case. We write $X \leftrightarrow Y$ for strict alternation between strategies X and Y on DAP boundaries, and $X \rightarrow Y$ for a one-shot transition from X to Y.

Destabilization and capture. Prior intermittent strategies can be decomposed into two phases: a *destabilization* phase that maximizes orphan production to drive down the next epoch's difficulty target, and a *capture* phase that maximizes relative revenue at the reduced difficulty. Prior work pairs a selfish variant (destabilization) with honest mining (capture). However, this pairing may be suboptimal, as selfish strategies may capture more revenue than honest mining. We use this decomposition in Section 6 as a heuristic for generating candidate temporal strategy compositions. We utilize this decomposition to provide a principled way to narrow down the search space of compositions to be examined; it is an open question whether it identifies the optimal strategy pairings.

3 Evaluation Setup

The simulator. Our evaluation builds on the discrete-event mining simulator of [11], itself an extension of [6]. We extend this simulator in two ways. First, we add per-epoch logging; at every DAP boundary, we record each attacker's cumulative and per-DAP relative revenue, orphan rate, and realized difficulty. This allows recovering the full TTP curve , and supplies the per-epoch state summaries we need for the automated approaches (Section 7). Second, we add per-epoch selection; an attacker may switch strategies at any DAP boundary based on observed fees and network conditions. This enables evaluating our temporal strategy composition framework, and is needed for the adaptive strategy modeling.

Block production and DAP. Following the Bitcoin protocol, we assume an expected block production time of 10 minutes and a DAP of $N = 2016$ blocks.

Reward models. We examine the three reward models discussed earlier. Under the block-reward-only model, each block carries a fixed reward normalized to 1. While for the other two models (that add fees), transaction fees accrue continuously over time between blocks (captured by fee_T defined earlier). The baseline transaction fees contribute approximately 17.5% of total block rewards for a block produced at the expected 10-minute production rate. We adopt this elevated baseline, consistent with prior work [5, 6], to reflect fees' growing share of future miner revenue [23]. To model noise in realized fees, a block's fee is drawn uniformly from $[0.75 \cdot \mathsf{fee}_\mathsf{T}, \mathsf{fee}_\mathsf{T}]$.

Whale transactions. When enabled, each block independently contains a whale transaction with probability $p \in \{0.05, 0.1\}$ (with $p = 0.0$ being the no-whale baseline), and a high-value fee represented as a magnitude multiplier $m \in \{4\times, 8\times\}$ applied to fee_T for the corresponding inter-block period. We vary p and m to examine their impact on TTP.

Game runs and statistics. Each (simulated) game run continues until the public chain reaches block height $24,192$ – representing 12 DAPs. Reported quantities are averaged, with 95% confidence intervals, over 100 independent runs per parameter configuration.

4 TTP Across the Single-attacker Strategy Space

Baseline strategy dominance. We examine each strategy across the full (α, γ) parameter grid with 0.05 increments. We measure profitability, which is taken as the minimum of the computed 95% CI collected across 100 game runs each of $12,096$ blocks (6 DAPs, around 12 weeks based on the 10-minute block time). Figure 1 shows the dominant strategies, i.e., those with the highest profitability, among the 10 strategies in the block-reward only setting. The dominant strategy varies systematically with α and γ, forming distinct regions rather than a single strategy dominating. While this result is established in the block-reward setting [18], we include it as a baseline comparison for our TTP-analysis that follows.

Min-TTP strategy rationality. The left panel of Figure 2 shows the minimum TTP strategy of the 10 strategies. The landscape is fragmented; no single strategy minimizes TTP across the entire profitable region, mirroring asymptotic profitability findings, though TTP-minimizing regions map to different dominant strategies. For $\alpha \in [0.25, 0.35]$ and $\gamma < 0.3$, the composed strategy $L \circ T_1$ (i.e., stubborn lead-trailing-1-fork) achieves the minimum TTP, while classic selfish mining S dominates for $\gamma > 0.5$. The publish-N families P_3 and P_4 dominate only near

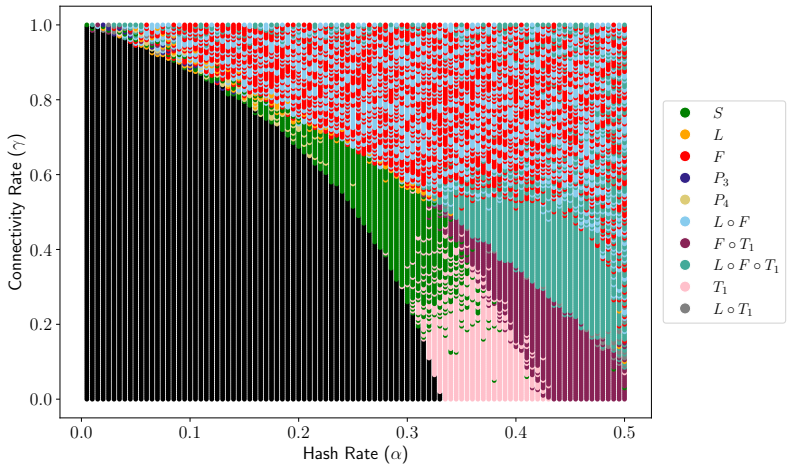


Figure 1 Profit dominant strategy across parameter space (single attacker, block-reward only).

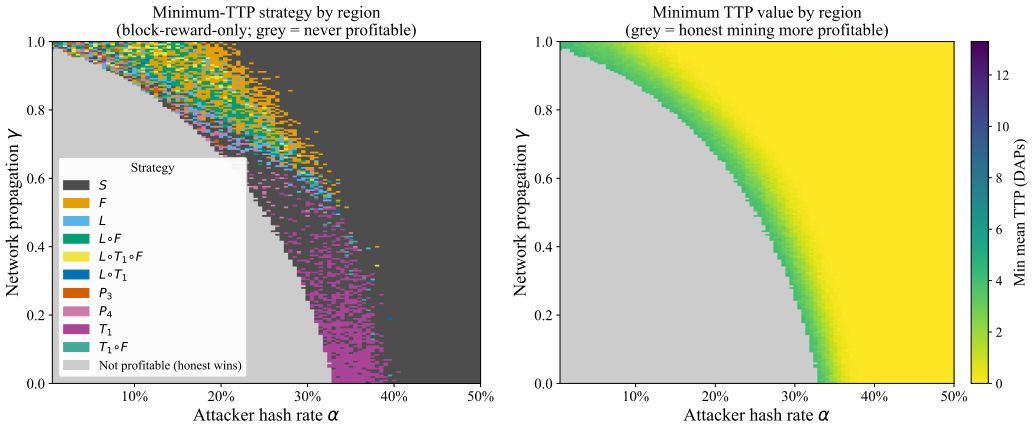


Figure 2 TTP-minimizing strategy across parameter region, and minimal TTP per parameter choice (single attacker, block-reward only).

the right edge, when $\alpha \geq 0.40$, where their publication threshold takes on a key advantage. The grey unprofitable region (within 6 DAPs) is for $\alpha \leq 0.33$, which aligns with asymptotic profitability results.

TTP magnitude. The right panel of Figure 2 shows that even within the profitable zone, TTP ranges from 0 DAPs (immediate profitability at high α) to $5-6$ DAPs near the boundary. TTP drops sharply (from around 4 to 1) when α moves from 0.38 to 0.40 at $\gamma = 0.0$. For $\alpha = 0.35$ and $\gamma = 0.25$, the min-TTP strategy stubborn-lead-fork mining $L \circ F$ reaches TTP ≈ 2 DAPs, while S (despite higher asymptotic profitability at this α value) has TTP ≈ 4.5 DAPs. Profit-maximizing and TTP-minimizing strategies therefore diverge; an attacker with a 3-DAP attack window should choose $L \circ F$ over S despite S's greater long-run advantage.

Value-aware (VA) strategies: Profit and TTP under fees and whale transactions. Figure 3 plots each strategy family at $\alpha = 0.25$ and $\tau = 1.5$ based on its cumRA (expressed as a percentage above what the miner would earn when mining honestly) and its conditional

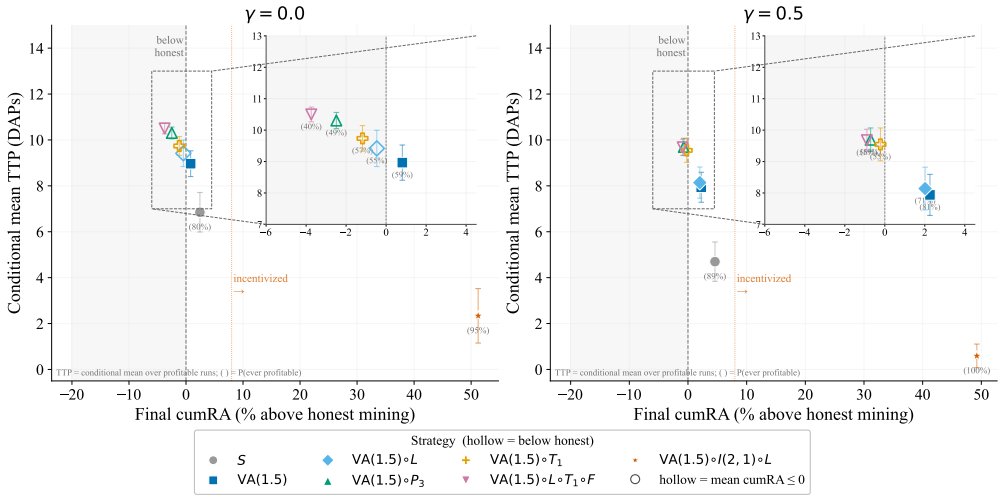


Figure 3 Value-aware strategies; profitability versus TTP at $\alpha = 25\%$, $\tau = 1.5$, $\kappa = 0.25$ (single attacker setting).

mean TTP. Filled markers indicate strategies whose mean revenue is positive across all 100 runs, while hollow markers indicate strategies that are unprofitable in expectation. The conditional mean TTP refers to the DAP of the first sustained revenue profitability, i.e., cumRA remains positive, over the subset of runs where the selfish miner achieves positive revenue advantage at the game end. Cases where a selfish miner never achieves continual profit are excluded from the mean but are reflected in the finite extent of each curve.

Among the non-incentivized VA strategies, every family except the value-aware classic selfish mining VA $\circ S$ earns less than honest mining in expectation at $\gamma = 0$. VA-selfish is marginally above honest at $+0.8\%$, while VA-stubborn-lead VA$\circ L$ sits near zero (-0.5% to $+0.1\%$). Also, VA-Publish-3 VA$\circ P_3$, VA-trail-1 VA$\circ T_1$, and VA-Stubborn-lead-trailing-1-fork VA$\circ L \circ T_1 \circ F$ fall between -1.2% and -3.7%. For these hollow-marker strategies, the reported TTP ($\approx$ 9–10 DAPs) reflects only the 40–57% of runs that turn profitable; the remaining runs never recover. Plain classic selfish mining S, by contrast, earns $+2.4\%$ with 76–80% probability of profitability.

The incentivized strategy value-aware incentived-stubborn-lead-trailing-1 VA$(1.5) \circ I(2, 1)\circ L$ is notably different. It earns approximately $+47\%$ above honest mining, and reaches profitability in a conditional mean of 0.25 DAPs, and is profitable in 96% of runs. In general, incentivized strategies deliver improved gains compared to other strategy variants on both dimensions, where they increase profitability and decrease TTP. Despite this, it is critical to note that this is under the presence of honest-but-rational miners (here, we have $\kappa = 0.25$).

At $\gamma = 0.5$, we observe the same trend as above, with only a slight rightward shift to the final cumRA. Similarly, VA$(1.5) \circ I(2, 1)\circ L$ remains dominant at $+46\%$ profitability with 98% probability with TTP of 0.25 DAPs.

Reward model sensitivity: Transaction fees and whale transactions. Figure 4 shows cumRA and TTP for value-aware and incentivized strategies under three reward models: block rewards and transactions fees only, light whale amplification (i.e., $p = 0.05$, $m = 4\times$), and heavy whale amplification (i.e., $p = 0.10$, $m = 8\times$). We set $\alpha = 0.25$, $\gamma \in \{0.0, 0.5\}$ and $\kappa = 0.25$ (we assess robustness of parameter choices in Appendix A in the paper full version).



Figure 4 Cumulative revenue advantage (% relative to honest mining) for seven selfish-mining strategies in the single attacker setting across three reward-models and two γ values at ($\alpha = 0.25, \kappa = 0.25$, $\tau = 1.5$). In profitable cells indicate mean TTP in DAPs.

Our findings highlight two distinct classes. Firstly, non-incentivized strategies S, VA$\circ S$, VA$\circ P_3$, VA$\circ T_1$, and VA$\circ L \circ T_1 \circ F$ remain in the range of -13% to $+5\%$ across all reward models and are largely insensitive to fee variations where whale events add at most a few percentage points. Secondly, incentivized strategies VA$\circ I(2,1)$ and VA$\circ I(2,1) \circ L$ have consistent profit across all reward models, ranging from $+19\%$ (fees only, $\gamma = 0.5$) to $+81\%$ (heavy whale, $\gamma = 0.5$), with profitability increasing and TTP decreasing consistent with increases of whale transaction frequency and fee magnitude. When $\gamma = 0.5$, we observe that incentivized strategies reach profitability remarkably quickly with VA $\circ\, I(2,1)$ achieving TTP < 0.6 DAP (≈ 8 days) across all fee regimes, and VA $\circ\, I(2,1) \circ L$ as low as 0.2 DAP in the fees-only baseline. When $\gamma = 0$, we see that the same strategies remain profitable but TTP rises to 8–10 DAP ($\approx$ 16–20 weeks).

Finding: Incentivization, not value-awareness, drives fast profitability. Across the static strategy results above, a sharp separation emerges between strategies that include the incentive mechanism $I(\eta, n)$ and those that do not. The separation holds along every axis we test; profitability, TTP, probability of profitability, and sensitivity to whale dynamics.

As seen in Figure 4, for $\alpha = 0.25$, $\gamma = 0.5$, $\kappa = 0.25$, the best non-incentivized value-aware strategy VA$\circ S$ earns $+5\%$ above honest mining with TTP ≈ 5.4 DAPs. Under the same parameters, VA$(1.5) \circ I(2,1) \circ L$ earns $+26\%$ with a TTP of 0.2 DAPs – around $5\times$ improvement in revenue advantage and around $27\times$ reduction in break-even time. Removing the incentive component from the strategy shows a clear collapse, both in terms of revenue that falls to near zero and TTP that rises to unreliable levels.

This leads to a natural question on why value-awareness alone is insufficient for incentivized strategies to achieve gains. In a nutshell, value-aware publication only changes *when* an attacker releases blocks it has already mined; it does not impact *who* mines on the attacker's branch during a fork. Carlsten's improved selfish mining demonstrated that fee-conditional publication modestly improves the return per high-fee block, but the underlying fork-resolution probability is still bounded by γ. The incentive mechanism, by contrast, affects fork-resolution itself. By posting conditional fees on its branch being extended, the attacker recruits honest-but-rational hashrate that would otherwise default to the honest chain. Thus, in general these two mechanisms improve selfish mining in two different directions, but the fork-resolution gains from incentivized strategy is where the larger gains lie.

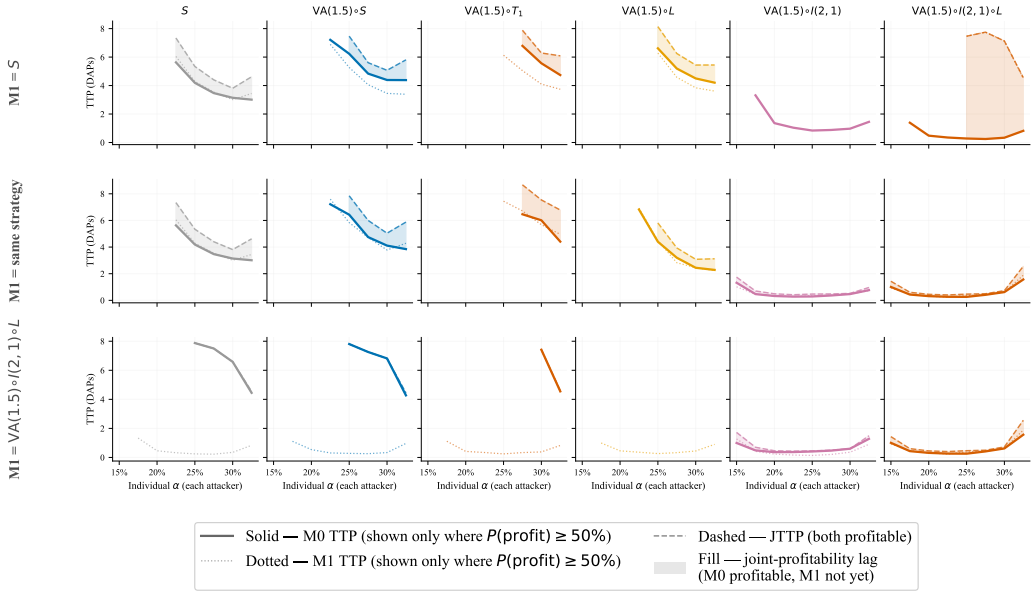


Figure 5 TTP and JTTP in the multi-attacker setting ($\gamma = 0.5$). Lines appear only where $P(\text{profit}) \geq 50\%$.

The result is contingent on honest-but-rational miners being present, i.e., $\kappa > 0$, and on a reward model that admits per-block fee variation. In the block-reward-only model, the incentive mechanism is unavailable and so the strategy landscape collapses back to the map of Figure 2.

5 TTP in the Multi-attacker Setting

We next examine TTP in the two-attacker symmetric setting, where each attacker holds the same individual hash rate α_i. All simulations assume whale transactions with $p = 0.05$ and $m = 4\times$, $\gamma \in \{0.0, 0.5\}$, and $\kappa \in \{0, 0.25\}$. Each parameterization run 100 times over 24,192 blocks, which is equivalent to 12 DAPs. Analogous to the conditional mean TTP, the conditional JTTP is defined over runs where both attacking miners, denoted as M_0 and M_1, reach positive cumulative revenue advantage.

Figure 5 reports conditional mean TTP and JTTP at $\gamma = 0.5$, $\kappa = 0.25$. Rows fix M_1's strategy: $M_1 = S$ (first row), M_1 uses the same strategy as M_0 (second row), and M_1 deploys VA(1.5)$\circ I(2,1)\circ L$ (third row). Columns vary M_0's strategy across six strategies of increasing sophistication (and profitability). Lines are drawn only where the strategy achieves positive revenue advantage in at least 50% of runs (i.e., $P(\text{profit}) \geq 50\%$), and so no line indicates the strategy fails to clear that threshold, i.e., the reliability threshold. The *joint-profitability lag* (shaded band) is the gap between M_0's individual TTP and JTTP.

Against a selfish attacker (first row). Non-incentivized strategies S, VA(1.5)$\circ S$, VA(1.5)$\circ T_1$, and VA(1.5)$\circ L$ show no TTP line below $\alpha_i \approx 22.5$–27.5%. Where lines do appear, TTP clusters between 5.6 and 7.2 DAPs at entry, with VA(1.5)$\circ S$ (TTP = 7.2 DAPs at $\alpha_i = 22.5\%$) trailing classic selfish mining S (TTP = 5.6 DAPs). As before, block-value-awareness alone does not help, and modestly delays the time to break-even without the incentive component.

Incentivized strategies break sharply from this pattern. VA(1.5)$\circ I(2,1)$ is profitable starting at $\alpha_1 = 17.5\%$ with TTP = 3.3 DAPs, reaching 1 DAP at $\alpha_i = 22.5\%$ and 0.84 DAPs at $\alpha_i = 25\%$. VA(1.5)$\circ I(2,1)\circ L$ becomes profitable starting at $\alpha_i = 17.5\%$ with TTP = 1.4 DAPs, converging to 0.35 DAPs at $\alpha_i = 22.5\%$ and 0.28 DAPs at $\alpha_i = 25\%$, which is roughly 15× faster than S at the same hash rate.

The joint-profitability lag for an incentivized M_0 against selfish M_1 is large. For VA(1.5)$\circ I(2,1)$, M_1 never achieves majority profitability across the tested range. This is due to the fact that the incentive mechanism captures high-fee transactions first, suppressing M_1's revenue below the reliability threshold, so JTTP is undefined. For VA(1.5)$\circ I(2,1)\circ L$, M_1 achieves profitability only starting at $\alpha_i = 25\%$ with TTP = 7.5 DAPs, where at this point M_0 has already been profitable for 7.2 DAPs, with JTTP = 7.47.

Self-competition (second row). When M_0 and M_1 deploy identical strategies, the picture changes. Non-incentivized strategies are largely unaffected: S and VA(1.5)$\circ L$ retain profitability entry thresholds near $\alpha_i = 22.5\%$ and TTP in the 4–7 DAP range, similar to the first row. Incentivized strategies benefit from symmetry: VA(1.5)$\circ I(2,1)\circ L$ and VA(1.5)$\circ I(2,1)$ are both profitable starting at $\alpha_i = 15\%$ with TTP $\approx$ 1 DAP for each. The joint-profitability lag shrinks to 0.4–0.5 DAPs at $\alpha_i = 15\%$ and narrows further at higher α, against the 7.2 DAP lag observed in the first row at $\alpha = 25\%$.

Against the strongest adversary (third row). When M_1 adopts VA(1.5)$\circ I(2,1)\circ L$, non-incentivized strategies fare worst. In particular, S shows no profitable line below $\alpha_i = 25\%$ (TTP = 7.9 DAPs), and VA(1.5)$\circ L$ fails to establish majority profitability within 12 DAPs, appearing only marginally at $\alpha_i = 32.5\%$ with TTP = 5.0 DAPs. On the other hand, incentivized strategies fare better. As shown in the figure, VA(1.5)$\circ I(2,1)$ and VA(1.5)$\circ I(2,1)\circ L$ are both profitable starting at $\alpha_i = 15\%$ with TTP $\approx$ 1 DAP, closely matching their second row curves. Thus, we observe that a strong incentivized opponent does not raise the TTP or profitability threshold of a competing selfish miner.

Whale dynamics under competition. Figure 6 plots conditional mean TTP against $P(\mathsf{profit})$ for M_0 across all 38 value-aware strategy configurations against a classic selfish miner M_1 at $\alpha_i = 25\%$, $\kappa = 0.25$, spanning seven strategy families, two threshold values $\tau \in \{1.5, 2.0\}$, and four whale regimes $p \in \{0.05, 0.10\}$, $m \in \{4\times, 8\times\}$.

The figure reveals a sharp separation between incentivized and non-incentivized families. Non-incentivized strategies VA, VA$\circ T_1$, VA$\circ L$, VA$\circ P_3$, and VA$\circ L\circ T_1\circ F$ cluster in the upper-right region at TTP = 6–11 DAPs with $P(\mathsf{profit}) = 40$–84%. Also, several VA$\circ P_3$ and VA$\circ L\circ T_1\circ F$ configurations fall below $P(\mathsf{profit}) = 50\%$ at $\gamma = 0.0$, meaning that they fail to achieve majority block-share profitability within 12 DAPs against $M_1 = S$. Incentivized families VA$\circ I(2,1)$ and VA$\circ I(2,1)\circ L$ occupy the top-left corner (visible in the zoom inset) with TTP < 1 DAP and $P(\mathsf{profit}) > 90\%$ across nearly all configurations for both γ values.

The parameter γ has a varying impact across the two strategy groups. For $\gamma = 0.5$, the non-incentivized cluster shifts leftward and upward relative to $\gamma = 0.0$, where VA strategies improve from TTP $\approx$ 8–10 DAPs to 6.7–8.2 DAPs, reflecting the benefit of better connectivity. The incentivized cluster is largely unaffected; both families already achieve TTP < 0.6 DAPs and $P(\mathsf{profit}) > 92\%$ at $\gamma = 0.0$ and improve marginally to TTP ≈ 0.3 DAPs and $P(\mathsf{profit}) \geq 95\%$ at $\gamma = 0.5$. Within this cluster, larger whale parameters ($m = 8\times$ and $p = 0.10$) push points toward the top-left corner, confirming that both higher whale transaction frequency and magnitude independently accelerate capturing fees.

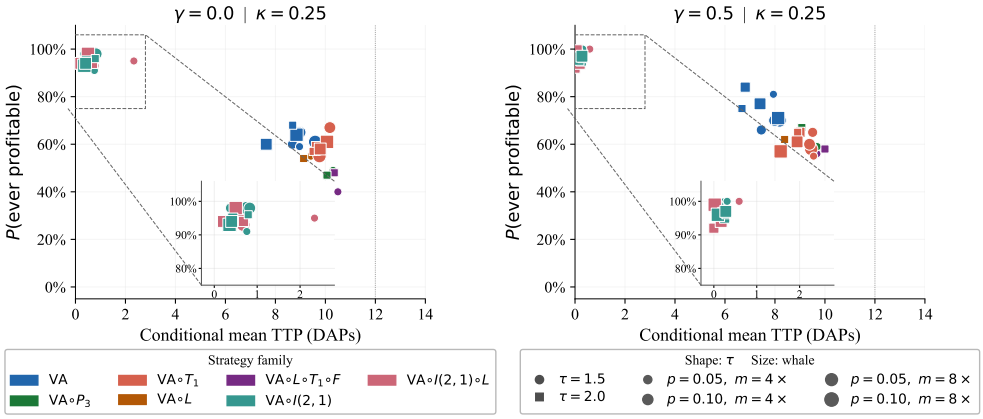


Figure 6 Value-aware strategies in the multi-attacker setting; profitability versus TTP of M_0 when competing against a classic selfish miner M_1 ($\alpha = 0.25$).

Finding: Asymmetric incentive adoption produces the dominant multi-attacker lag. The single-attacker setting finding from Section 4, i.e., incentivization is the driver of TTP reduction, extends into the multi-attacker setting in a specific way. That is, the joint profitability lag is large when only one attacker deploys the incentive mechanism, and is roughly absent otherwise. In the first row in Figure 5, the only pairings producing substantial lag are those where M_0 runs an incentivized strategy against non-incentivized $M_1 = S$: VA(1.5)∘$I(2,1)$∘L at $\alpha_i = 25\%$ has a TTP of just= 0.28 with M_1 not becoming also profitable until = 7.47 DAPs while VA(1.5)∘$I(2,1)$ suppresses M_1 entirely. The four non-incentivized M_0 strategies in the first row in Figure 5 show small lag; both attackers converge together because neither has an advantage in capturing the honest-but-rational miner's hashrate. The second row in the figure validates this, where under symmetric strategy choice the lag becomes very small regardless of whether the shared strategy is incentivized.

Underpinning this observation is the same fork resolution mechanism impact that found beneficial in the single-attacker setting. When one attacker posts branch-conditional fees and the other does not, the rational hashrate κ flows preferentially to the incentivizing party, depriving the other of any boost that would come from honest-but-rational miners. Conversely, in the symmetric setting, competition over honest-but-rational miners is restored as now both attackers can post incentives. We see two implications of this finding. First, a sustained gap between the observed miner revenue trajectories may be a signal of one-sided use of incentives by an attacker running a non-incentivized/value-aware strategy. Second, incentivized strategies reach the symmetric joint-profitability regime with a individual hashrate as low as 15%.

6 Temporal Strategy Composition

Building on the destabilization/capture intuition introduced in Section 2.4, we evaluate temporal compositions that alternate between strategies maximizing orphan production and strategies maximizing revenue capture under reduced difficulty. We highlight that the static-strategy maps in Figures 1 and 2 (generalizable to their value-aware and multi-attacker analogues) already characterize TTP-minimizing and profit-maximizing choice for every cell of the (α, γ) grid. As a result, we do not separately evaluate compositions reacting to shifts in α or γ, since an attacker observing such a shift can simply remap to the corresponding

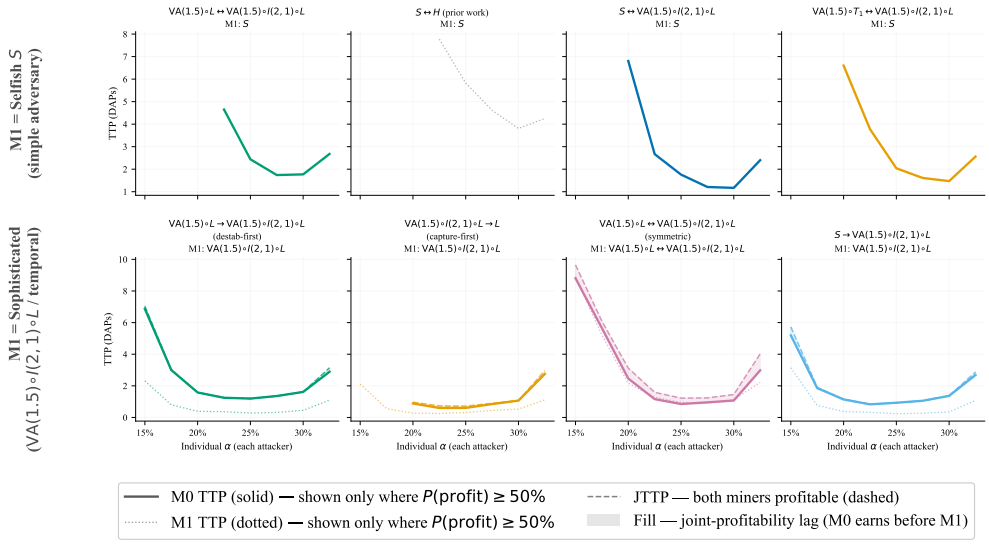


Figure 7 TTP and JTTP vs. α for eight temporal compositions in the multi-attacker setting ($\alpha_i = 0.25, \gamma = 0.5, \kappa = 0.25$). Lines appear only where $P(\text{profit}) \geq 50\%$.

static dominant strategy. Our focus instead is on whether alternating strategy choices at fixed DAP intervals improve TTP or profitability over the best static strategy for the same parameter configuration.

We evaluate alternating temporal composition in the multi-attacker setting across two sub-settings that capture the realistic multi-miner environment covering transaction fees, honest-but-rational miners, and whale transactions. In particular, we have (1) attacker 0 M_0 against a classic selfish attacker $M_1 = S$, and (2) M_0 against a sophisticated attacker $M_1 = \text{VA}(1.5)\circ I(2,1)\circ L$ or another temporal strategy. Given their advantage in the transaction-fee setting, we focus primarily on value-aware strategy variants.

For each sub-setting, we examine how TTP and JTTP vary with α (shown in Figure 7), then we measure cumRA (shown in Figure 8). All experiments assume whale transactions with $p = 0.05$ and $m = 4\times$, $\gamma \in \{0.0, 0.5\}$, 100 independent runs of 12 DAPs per parameter setting, and $\alpha_i = 25\%$ per attacker unless stated otherwise. In Figure 7, lines are drawn only where $P(\text{profit}) \geq 50\%$, and so an absent line indicates that the strategy is too unreliable to characterize a meaningful TTP.

Temporal composition against a simple adversary. The first row of Figure 7 shows TTP and JTTP curves for four representative temporal compositions played against a classic selfish opponent $M_1 = S$. Examining classic intermittent selfish mining $S{\leftrightarrow}H$, as introduced in [19], we find that in the multi-attacker setting this strategy produces no visible line at any tested α; less than half of the games yield a positive cumRA, so reliable profitability is never established. The three novel temporal compositions $\text{VA}(1.5)\circ L{\leftrightarrow}\text{VA}(1.5)\circ I(2,1)\circ L$, $S{\leftrightarrow}\text{VA}(1.5)\circ I(2,1)\circ L$, and $\text{VA}(1.5)\circ T_1{\leftrightarrow}\text{VA}(1.5)\circ I(2,1)\circ L$ reach TTP with $P(\text{profit}) \geq 50\%$ starting at $\alpha_i \approx 22.5\%$, 20%, and 20%, respectively. At $\alpha = 25\%$, all three achieve $P(\text{profit}) = 1$, with TTPs of 2.4, 1.8, and 2.0 DAPs, respectively. For reference, static $\text{VA}(1.5)\circ I(2,1)\circ L$ is reliably profitable starting at $\alpha_i = 17.5\%$ and converges in only 0.6 DAPs at $\alpha_i = 25\%$. Thus, it has an additional 1–2 DAP break-even lag in the temporal variants.

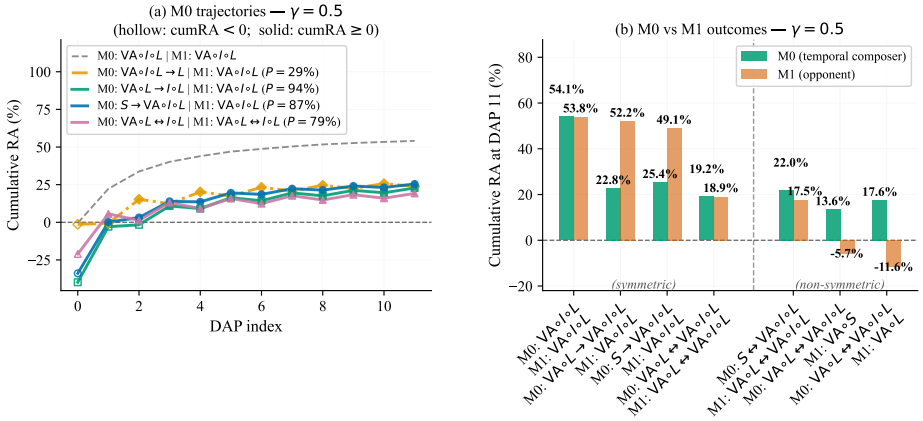


Figure 8 Cumulative revenue advantage trajectories at $\alpha = 25\%$, $\gamma = 0.5$ for five temporal compositions (left), and M_0 vs. M_1 cumRA for symmetric and non-symmetric strategy pairs (right).

Figure 8 confirms that this break-even cost translates into lower final cumRA. As shown, none of the five temporal compositions in the figure exceeds the static VA(1.5)$\circ I(2,1)\circ L$ variant (+45.5% at $\gamma = 0.5$). The $S\leftrightarrow$VA(1.5)$\circ I(2,1)\circ L$ composition initially tracks the static VA(1.5)$\circ I(2,1)\circ L$ trajectory, then falls progressively short as the classic selfish phase suppresses fee-capture every other DAP.

Temporal composition against a sophisticated adversary. The second row of Figure 7 replaces M_1 with VA(1.5)$\circ I(2,1)\circ L$ (static) or an opponent running a temporal strategy.

Two structural differences from the first row emerge here. First, the hashrate threshold is universally lower: all four compositions reach TTP with $P(\text{profit}) \geq 50\%$ starting at $\alpha_i \approx 15$–20%, compared to 20–22.5% against $M_1 = S$. Second, phase order is consequential. Destabilize-first (i.e., VA(1.5)$\circ L\rightarrow$VA(1.5)$\circ I(2,1)\circ L$) enters the reliability region at $\alpha_i = 15\%$ but requires 2.4 DAPs for M_0 to converge at $\alpha = 25\%$. Capture-first (i.e., VA(1.5)$\circ I(2,1)\circ L\rightarrow L$) does not become reliably profitable until $\alpha_i = 20\%$ yet converges in only 1.2 DAPs once it does. The left panel in Figure 8 illustrates why VA(1.5)$\circ I(2,1)\circ L\rightarrow L$ is the weaker choice. As shown, the trajectory remains near zero for most of the game and the strategy is profitable in only 29% of runs at $\alpha_i = 25\%$, $\gamma = 0.5$, making it unreliable in practice despite its lower mean TTP. Where as when M_0 =VA(1.5) $\circ$ $L\rightarrow I(2,1)\circ L$ or $M_0 = S\rightarrow$ VA(1.5)$\circ I(2,1)\circ L$ and $M_1 =$ VA(1.5)$\circ I(2,1)\circ L$, M_0 is effective in capturing high-fee transactions and becoming profitable within 0.6 DAPs. In both instances, we find that JTTP is dictated entirely by the TTP of M_0.

The right panel in Figure 8 shows cumRA for both symmetric and non-symmetric strategy pairs. Under symmetric temporal adoption both miners earn $\approx 39.5\%$ cumRA at $\gamma = 0.5$, compared to $\approx 58\%$ each under symmetric static case of VA(1.5)$\circ I(2,1)\circ L$. The symmetric case, when both miners using VA(1.5)$\circ L\leftrightarrow$VA(1.5)$\circ I(2,1)\circ L$, yields M_0 TTP = 1.7 DAPs and M_1 TTP = 1.9 DAPs at $\alpha = 25\%$. The bar chart highlights this finding explicitly, with mutual adoption of the temporal strategies reducing per-miner revenue below what either could achieve by unilaterally adopting the static strategy.

The $S\rightarrow$VA(1.5)$\circ I(2,1)\circ L$ composition against M_1 running VA(1.5)$\circ I(2,1)\circ L$ offers an alternative result trend, starting as S and alternating with VA(1.5)$\circ I(2,1)\circ L$. At $\alpha = 25\%$, M_1 converges in 0.5 DAPs and M_0 in 1.9 DAPs, with M_0's delayed transition accounting for the JTTP lag. Despite this, $S\rightarrow$VA(1.5)$\circ I(2,1)\circ L$ is reliably profitable starting at $\alpha_i = 15\%$ and offers comparable cumRA to VA(1.5)$\circ L\leftrightarrow$VA(1.5)$\circ I(2,1)$ against M_1 =VA(1.5)$\circ I(2,1)\circ L$.

Finding: Temporal composition rarely outperforms the best static strategy. Across the regimes tested, no alternating temporal composition matches either the TTP or the asymptotic revenue of static VA(1.5)$\circ I(2,1)\circ L$. Against a classic selfish miner at $\alpha_i = 25\%$, the four temporal compositions' lowest TTP across hashrate is still 1.8–2.4 DAPs against the static strategy's 0.6 DAPs, which is a 3–4× slowdown, and none of the five temporal variants in Figure 8 reach this TTP nearly as frequently across the examined hashrates as the static strategy. Against a sophisticated adversary, the picture does not improve much despite producing a lower TTP and smaller lag on average (reinforcing our findings in Section 5). As shown, all miners TTP still fall short of matching that of VA(1.5)$\circ I(2,1)\circ L$.

Where temporal composition does help. The framework still admits compositions over observable shifts in α or γ, as noted at the beginning of this section; that is, an attacker whose connectivity or hashrate genuinely changes mid-attack does benefit from remapping its strategy. The negative finding here is narrower and specific. In particular, alternating between strategies at fixed DAP intervals, in a stationary environment, does not outperform the best static strategy identified in Sections 4 and 5.

7 Adaptive Strategy Selection at the DAP Level

The temporal composition results from the previous section show that fixed-schedule strategy alternation rarely beats the best static strategy. A natural follow-up question is whether *state-conditioned* strategy selection, i.e., in response to observed network conditions, would perform better. Existing controllers that targeted similar problem are typically hand-derived policies or per-environment DRL pipelines [5, 14], both of which require substantial engineering investment and, for DRL, per-environment retraining.

In this section, we investigate whether a per-DAP selector can instead be built directly from the strategy knowledge developed in Sections 5–6, and how much the selector's *implementation*, i.e., the mechanism that turns encoded knowledge into a decision, matters once that knowledge is established. We study two selectors that share an identical interface (state summary in, strategy name or NO-CHANGE out) but differ in how they apply the encoded strategy knowledge. The first is a deterministic fixed policy that we manually crafted using the findings of prior work [11] and those in Sections 5 to 6. In particular, we map the current state into an ordered decision tree according to various applicability conditions and thresholds. While the second is an LLM agent that exercises judgment over the material at each call.

Both selectors draw on the same underlying domain knowledge but receives it at different levels of specification. That is, the fixed policy requires that prior knowledge be hand-translated into explicit, checkable decision rules, while the AI agent receives it as high-level context and performs that translation itself on per-call basis. Varying this level of encoding, under the same knowledge and surrounding control loop, isolates what depends on the knowledge itself from what depends on how, and by whom, it is turned into an action. To the best of our knowledge, no prior work has examined either LLM-driven selfish mining policies or deterministic strategy-level adaptive strategy selectors.

Motivation and impact. We frame this as a proof-of-concept exploration with three motivations and one broader implication as follows.

Feasibility. We want to understand whether such an adaptive selector can identify dominant selfish mining strategies from observable signals alone, react to mid-run shifts, and perform against an unknown competing attacker.

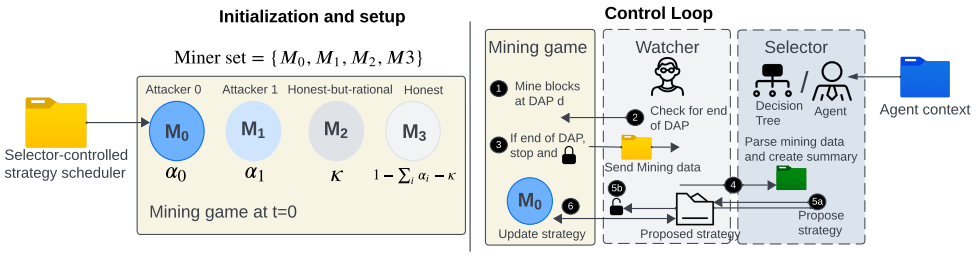


Figure 9 Per-DAP control loop for adaptive strategy selection.

Implementation impact. Given identical inputs and encoded knowledge, we aim to investigate whether an agentic selector reach different decisions and profitability compared to a selector that mechanically applies a fixed rule set.

Selector limiting factors. We wish to understand what limits a per-DAP selector regardless of implementation: whether starting with a non-performant strategy imposes unrecoverable costs, where a mid-run network shift can be recovered once detected, and whether these limits are environment or selector related properties.

Implication: real-world traction. Custom DRL-based selfish mining controllers require DRL expertise, and manually-crafted selectors require extensive exploration to identify decision boundaries. LLM agents lower this barrier substantially: a strategy-selector agent requires only a system prompt and an API key. Even if at some performance cost (in terms of lower profitability) compared to hand-trained DRL/manually-crafted policies, such agents enlarge the population of attackers for whom adaptive selfish mining is operationally feasible.

Architecture. Figure 9 illustrates the system. The left panel shows the per-game setup, where we have four miner classes: selector-controlled attacker M_0, a fixed-strategy attacker M_1, honest-but-rational miner M_2, and honest miner M_3. They are instantiated when the game begins. The right panel shows the per-DAP control loop. A lightweight daemon runs alongside the simulator, triggered once per completed DAP via a file-system lock (steps 1–3). At each trigger, this daemon reads two rolling CSV logs: a 200-block sliding window ($\approx 10\%$ of one DAP, ≈ 33 hours at target difficulty) supplying fee-variance and whale-event signal, and an epoch-level log retaining the two most recently completed DAPs, supplying orphan rate, block fraction, and seconds-per-block. From these, the daemon assembles a state summary – orphan rate, hashrate shares, γ, fee distribution, cumulative rewards – and passes it to the selector (step 4). The decision is written to a handoff file (step 5a); the simulator resumes on lock deletion and applies the strategy to M_0 (step 5b).

Selectors. For the AI agent, we use Anthropic's `claude-sonnet-4-6` [1], invoked with a fixed system prompt, i.e., the "Agent context" input in Figure 9. From a developer's perspective, the development of this context needs only supplying a high-level specification of the strategy space, applicability conditions, and a reference to prior known performance results under various model settings. As such, based on this context, the agent will reason over the current state at call time.

Concretely, the given prompt consists of four parts: (1) the full strategy space (base, value-aware, and incentivized variants) with per-strategy applicability conditions, e.g., incentivized variants require $\kappa > 0$ and per-block fee variation; (2) a reference section giving the joint-profitability thresholds for two attackers across the (γ, κ) space, expected per-epoch revenue at representative α, and structural orphan-rate baselines separating competition noise from

signal worth acting on; (3) the daemon's observation summary; and (4) an output constraint. This material is supplied as background context rather than as an executable rule list (as in the fixed policy); the prompt characterizes when strategies tend to apply and how they tend to perform. The exact translation from context to action is left to the model at time of call. Moreover, within a run, the agent also carries forward a record of which strategies it has already selected and how they performed, and can revise course mid-run if a previously chosen strategy under performs.[1]

For the fixed policy, it is implemented as an ordered list of conditional rules derived from the same base knowledge provided to the agent (i.e., strategy space, applicability conditions, past reference section, and current blockchain state). This policy demands considerably more integration work than the agent, as this background knowledge must be manually translated into an ordered list of rules, each with an explicit, checkable condition (organized as a decision tree) for when it fires. We hand-derived this rule set from findings of the present work as well as individual thresholds of prior work [11]. As conditions must be checkable from the state summary alone, they rely on coarse interpretations, i.e., interpreting κ as present or not, and defaulting to $\gamma = 0.5$ when it cannot be estimated.

The base knowledge was fixed before either selector was run and was not tuned against either selector's results. Neither selector currently carries information *across* runs – the agent's adaptivity is confined to the strategies it has tried within a single run, and decision tree's rule of the fixed policy are static once set. We leave fine tuning the agent's context or the decision tree's conditions based on accumulated outcomes across runs as future work.

Experimental design. We construct two scenarios, each bounded by static baselines, and run both selectors on every configuration. In the *stable environment* ($\gamma = 0.5$, $\kappa = 0.25$, whale fees ($p = 0.05$ and $m = 4\times$), $\alpha_i \in \{0.10, 0.20, 0.30\}$, $n = 10$ runs each), the selector must arrive at the dominant strategy from the observation summary alone, without direct visibility of M_1's strategy. We compare against static $VA(1.5) \circ I_{2,1} \circ L$ and report head-to-head win rates against M_1. In the γ-*switching* scenario (γ steps from 0.25 at DAPs 0–5 to 0.75 at DAPs 6–11; $\kappa = 0.25$; whale fees ($p = 0.05$ and $m = 4\times$); $\alpha \in \{0.10, 0.20, 0.30\}$; $n = 10$ each), we test whether a selector detects and exploits a mid-run propagation shift. We use two static strategies to bound performance: $VA(1.5) \circ I_{2,1} \circ L$ from block 0 (upper bound, the best-performing strategy in the multi-attacker setting) and $VA(1.5) \circ L$ (lower bound, fee-aware but not incentive-aware). We run each scenario under two initial conditions (seeded on $VA(1.5) \circ I_{2,1} \circ L$ and cold-started on $VA(1.5) \circ L$) since the selector's opening commitment turns out to be decisive.

Stable environment. Section 5 identifies $VA(1.5) \circ I_{2,1} \circ L$ as strongly dominant under whale fees, and both selectors (when seeded with it) never leave; NO-CHANGE in all 330 decisions for each, tracking the static run's cumRA within run-to-run variance and winning 10/10 against M_1 at $\alpha_i \geq 0.20$ (Table 3). Instead, starting on $VA(1.5) \circ L$, the picture depends sharply on α. In particular, both selectors adopt the incentive variant only when $\alpha_i = 0.30$, at the first DAP boundary, and issue NO-CHANGE for all twelve epochs at $\alpha_i \in \{0.10, 0.20\}$, leaving those runs as static $VA(1.5) \circ L$ measurements. Neither selector diverges from the other in this scenario, where given the same base knowledge, both reach the same decisions at every trigger.

[1] In terms of monetary cost, per-DAP inference costs $\approx$ \$0.000005 (9,909-token cached system prompt, 455 uncached input and around 26 output tokens per call at `claude-sonnet-4-6` pricing of \$3/\$15 per MTok). Hence, a twelve-epoch run costs around \$0.0001, which is considered negligible compared to mining pool operational costs in practice.

Table 3 Stable environment ($\gamma = 0.5$, $\kappa = 0.25$), $n = 10$ runs, cumRA is computed over 12 epochs; parentheses = represents ±1 std. (– = never profitable within 12 epochs). Win rate: runs (of 10) in which M_0 beats $M_1 = S$ head-to-head (recall S is classic selfish mining). Paired rows are the LLM agent and the fixed policy; slashed entries read LLM / fixed.

	α_i		
	0.10	0.20	0.30
Static VA∘$I \circ L$	$-33.4\%_{(4.0)}$	$+24.4\%_{(4.1)}$	$+55.7\%_{(3.8)}$
TTP (epochs)	–	0.9	0.8
LLM, ($VA \circ I \circ L$ seed)	$-33.5\%_{(3.0)}$	$+25.1\%_{(4.3)}$	$+56.7\%_{(3.4)}$
Fixed policy, ($VA \circ I \circ L$ seed)	$-33.3\%_{(2.9)}$	$+24.2\%_{(2.2)}$	$+55.5\%_{(3.6)}$
TTP (epochs)	–	0.8/0.9	0.7/0.7
LLM, ($VA \circ L$ seed)	$-15.6\%_{(2.6)}$	$-2.5\%_{(1.5)}$	$+7.4\%_{(3.1)}$
Fixed policy, ($VA \circ L$ seed)	$-15.1\%_{(1.9)}$	$-3.0\%_{(1.4)}$	$+7.3\%_{(1.8)}$
TTP (epochs)	–	4.0/6.7	2.6/2.7
Win rate /10, ($VA \circ I \circ L$ seed)	0/0	**10/10**	**10/10**
Win rate /10, ($VA \circ L$ seed)	**8/8**	**7**/3	4/5

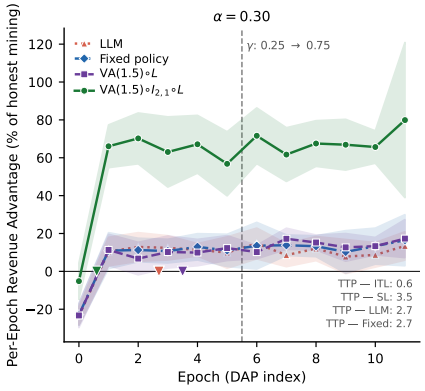


Figure 10 Per-epoch cumRA in the γ-switching scenario at $\alpha_i = 0.30$. Dashed line marks the γ: 0.25 → 0.75 transition at epoch 5. Band represents ±1 std. Both selectors are seeded on $VA(1.5) \circ L$.

γ-switching. What neither selector can resolve is the asymmetry introduced by the initial seed. Figure 10 shows the per-epoch cumRA when γ changes from 0.25 to 0.75 during epoch 5. The two selectors behave similarly and reach profitability at the same epoch (TTP= 2.7). In general, we find that switching mid-run forfeits gains relative to a strategy determined dominant from the start. That is, seeded on $VA(1.5) \circ L$, cumRA is $-1.8\% \pm 1.4$ at $\alpha = 0.20$ and $+7.0\% \pm 2.6$ at $\alpha_i = 0.30$. While a selector seeded on the dominant strategy correctly issues NO-CHANGE in every epoch of every run and closely tracks it: having a cumRA of $+25.1\% \pm 4.3\%$ at $\alpha_i = 0.20$ and $+56.7\% \pm 3.4\%$ at $\alpha_i = 0.30$. When combining this asymmetry with γ-switching, the picture is less optimistic. As shown, seeded on $VA(1.5) \circ L$, the per-epoch cumRA for epochs 6–11 is 1–2%, matching $VA(1.5) \circ L$ rather than the static incentive strategy's ≈ 6%/epoch from epoch 1. cumRA at $\alpha_i = 0.3$ is $+6.3\% \pm 2.0\%$, which is near the `va-L` lower bound ($+7.2\% \pm 2.4\%$) and far below the $VA(1.5) \circ I_{2,1} \circ L$-seed upper bound. Thus, we observe that both selector correctly notice the network change (despite a DAP delay), and switch strategy accordingly, but this switch cannot recover the cumRA gains forfeited by not having started on the dominant strategy.

Insight: Automated strategy selection trades profitability for adaptability, independent of implementation. Both selectors succeed at identifying the dominant strategy from observable signals without direct visibility of M_1. Also, each detects and responds to mid-run parameter shifts – capabilities unavailable to any static strategy. Notably, the LLM agent achieves that given only high-level specification and without requiring an expert-derived deterministic policy, which significantly simplifies integration. On the other hand, both selectors incur the same costs: a cumRA shortfall from the initial seed asymmetry, and a delay in TTP in responding to the γ-shift. An automated selector that does not know the right strategy in advance must spend epochs discovering it, and one that does know cannot anticipate a shift before it happens. This is not a defect of any one selector; it is inherent to per-DAP adaptive selection, whether the decision procedure is a fixed policy or an agentic one. As Section 6 shows, the bar to dominate a static strategy is already high. That is, the gains from adaptability have to exceed the cost of discovery and the profit lag caused by strategy switching, and neither implementation we test changes where that bar sits.

8 Conclusion

In this paper, we revisited selfish mining through the lens of TTP, asking not only what an attacker can earn asymptotically but when and how reliably they realize such profits. We provided a holistic study across several dimensions covering the single-attacker setting, multi-attacker competition, temporal composition, and automated per-DAP strategy selection. Our results show that profitability timing behaved differently compared to long-term revenue trends. In particular, TTP-minimizing and profit-maximizing strategies frequently diverge, making attack horizon a leading metric rather than an afterthought. Also, incentivized recruitment of honest-but-rational miners, rather than value-awareness or temporal composition, is the main driver of shortening the path to profitability. Furthermore, the performance of a fixed rule-based adaptive per-DAP strategy selector matches that of an LLM agent one when both are given the same strategy knowledge. Together, these results point future work away from richer temporal schedules and toward state-conditioned strategy selection, as well as studying TTP across varying network assumptions and models.

References

1 Anthropic. Claude Sonnet 4.6 System Card. Technical report, Anthropic, February 2026. URL: https://www-cdn.anthropic.com/78073f739564e986ff3e28522761a7a0b4484f84.pdf.

2 Maryam Bahrani and S Matthew Weinberg. Undetectable selfish mining. In *ACM Conference on Economics and Computation*, 2024.

3 Qianlan Bai, Yuedong Xu, Nianyi Liu, and Xin Wang. Blockchain mining with multiple selfish miners. *IEEE Transactions on Information Forensics and Security*, 18:3116–3131, 2023. doi:10.1109/TIFS.2023.3275736.

4 Qianlan Bai, Xinyan Zhou, Xing Wang, Yuedong Xu, Xin Wang, and Qingsheng Kong. A deep dive into blockchain selfish mining. In *IEEE international conference on communications (ICC)*, 2019.

5 Roi Bar-Zur, Ameer Abu-Hanna, Ittay Eyal, and Aviv Tamar. Werlman: To tackle whale (transactions), go deep (rl). In *IEEE Symposium on Security and Privacy (SP)*, 2023.

6 Miles Carlsten, Harry Kalodner, S Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016.

7 Michael Davidson and Tyler Diamond. On the profitability of selfish mining against multiple difficulty adjustment algorithms. *Cryptology ePrint Archive*, 2020.

8 Mustafa Doger and Sennur Ulukus. Incentive attacks in btc: Short-term revenue changes and long-term efficiencies. *arXiv preprint arXiv:2511.11538*, 2025. doi:10.48550/arXiv.2511.11538.

9 Mustafa Doger and Sennur Ulukus. When should selfish miners double-spend? *IEEE Transactions on Information Theory*, 2026.

10 Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International Conference on Financial Cryptography and Data Security*, 2014.

11 Colin Finkbeiner, Mohamed E Najd, Julia Guskind, and Ghada Almashaqbeh. Sok: Time to be selfless?! demystifying the landscape of selfish mining strategies and models. In *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2025.

12 Colin Finkbeiner, Connor Shaw, and Ghada Almashaqbeh. CSSL-UConn/temp-sim. Software, swhId: swh:1:dir:71dbe9c9ecdc32669573f00c1ef9950caad6bd2f (visited on 2026-09-17). URL: https://github.com/CSSL-UConn/temp-sim, doi:10.4230/artifacts.27966.

13 Cyril Grunspan and Ricardo Pérez-Marco. On profitability of selfish mining. *arXiv preprint arXiv:1805.08281*, 2018. arXiv:1805.08281.

14 Charlie Hou, Mingxun Zhou, Yan Ji, Phil Daian, Florian Tramèr, Giulia Fanti, and Ari Juels. Squirrl: Automating attack analysis on blockchain incentive mechanisms with deep reinforcement learning. In *Network and Distributed Systems Security (NDSS)*, 2021.

15 Patrik Keller and George Bissias. Generic selfish mining mdp for dag protocols. *arXiv preprint arXiv:2309.11924*, 2023. doi:10.48550/arXiv.2309.11924.

16 Hanqing Liu, Na Ruan, Rongtian Du, and Weijia Jia. On the strategy and behavior of bitcoin mining with n-attackers. In *Asia Conference on Computer and Communications Security (AsiaCCS)*, 2018.

17 Francisco J Marmolejo-Cossío, Eric Brigham, Benjamin Sela, and Jonathan Katz. Competing (semi-) selfish miners in bitcoin. In *ACM Conference on Advances in Financial Technologies (AFT)*, 2019.

18 Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016.

19 Kevin Alarcón Negy, Peter R Rizun, and Emin Gün Sirer. Selfish mining re-examined. In *Financial Cryptography and Data Security*, 2020.

20 Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *Financial Cryptography and Data Security*, 2017.

21 Roozbeh Sarenche, Ren Zhang, Svetla Nikova, and Bart Preneel. Selfish mining time-averaged analysis in bitcoin: Is orphan reporting an effective countermeasure? *IEEE Transactions on Information Forensics and Security*, 20:449–464, 2024. doi:10.1109/TIFS.2024.3518090.

22 Caspar Schwarz-Schilling, Sheng-Nan Li, and Claudio J Tessone. Stochastic modelling of selfish mining in proof-of-work protocols. *Journal of Cybersecurity and Privacy*, 2(2):292–310, 2022. doi:10.3390/JCP2020016.

23 TheMinerMag. Bitcoin transaction fees drop below 1% of block rewards, June 2025. URL: https://www.theminermag.com/news/2025-06-16/bitcoin-transaction-fees.

24 Qing Xia, Wensheng Dou, Tong Xi, Jing Zeng, Fengjun Zhang, Jun Wei, and Geng Liang. The impact analysis of multiple miners and propagation delay on selfish mining. In *IEEE Annual Computers, Software, and Applications Conference (COMPSAC)*, 2021.

25 Jiaping Yu, Shang Gao, Rui Song, Zhiping Cai, and Bin Xiao. Partial selfish mining for more profits. *arXiv preprint arXiv:2207.13478*, 2022. doi:10.48550/arXiv.2207.13478.

26 Ren Zhang and Bart Preneel. Publish or perish: A backward-compatible defense against selfish mining in bitcoin. In *Topics in Cryptology–CT-RSA*, 2017.

27 Shiquan Zhang, Kaiwen Zhang, and Bettina Kemme. Analysing the benefit of selfish mining with multiple players. In *IEEE International Conference on Blockchain (Blockchain)*, 2020.

28 Yiting Zhang, Ming Liu, Jianan Guo, Zhaojie Wang, Yilei Wang, Tiancai Liang, and Sunil Kumar Singh. Optimal revenue analysis of the stubborn mining based on markov decision process. In *International Conference on Machine Learning for Cyber Security*, 2022.

Optimal Reward Allocation via Proportional Splitting

Lukas Aumayr
Common Prefix, Athens, Greece
University of Edinburgh, UK

Zeta Avarikioti
Common Prefix, Athens, Greece
TU Wien, Austria

Dimitris Karakostas
Common Prefix, Athens, Greece

Karl Kreder
Dominant Strategies, Austin, TX, USA

Shreekara Shastry
Dominant Strategies, Austin, TX, USA

Abstract

Following the publication of Bitcoin's arguably most famous attack, selfish mining, various works have introduced mechanisms to enhance blockchain systems' game-theoretic resilience. The only proof-of-work reward rule with a Nash-equilibrium guarantee, FruitChains, demands reward finality on the order of days. The rules that settle in minutes have no such guarantee, and one of them, Reward Splitting, still outperforms FruitChains on most of the metrics that matter in deployment. This paper closes that gap between theory and practice.

We introduce FairChain, a two-level transformation for any proof-of-work Nakamoto-style protocol. At the protocol layer, FairChain records low-difficulty samples called *workshares* alongside blocks. At the reward layer, it applies *Proportional Reward Splitting (PRS)*: each height's reward is divided among the competing work objects in proportion to the intrinsic work behind them, with workshares supplying a fresh power estimate at every height. The fork-choice rule and block-production loop are left untouched, so the host chain's security carries over unchanged. Workshares can be discarded once the corresponding rewards mature, leaving zero on-chain footprint.

We prove FairChain is a ρ-coalition-safe ϵ-Nash equilibrium for sufficiently large parameters, matching FruitChains in theory. To evaluate practical performance, we leverage Markov decision processes and compute the optimal adversarial policy under each utility function, rather than the gain of any one attack. At a six-block confirmation window, FairChain raises the deviation threshold to 38% of mining power and beats every mechanism in that framework on incentive compatibility, subversion gain (except FruitChains above 42%), and censorship susceptibility (except FruitChains below 25%). Our construction has been adopted by a live PoW blockchain, with sub-kilobyte per-block storage overhead in deployment.

2012 ACM Subject Classification Security and privacy → Security protocols; Security and privacy → Distributed systems security; Security and privacy → Social aspects of security and privacy

Keywords and phrases blockchain, proof-of-work, fairness, game-theory, rewards

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.30

Related Version *Full Version*: `https://arxiv.org/abs/2503.10185` [1]

Supplementary Material *Software (Source Code)*: `https://github.com/commonprefix/proportional-reward-splitting-MDP` [10]
archived at `swh:1:dir:4a244dd0f68a10c4abb1fa0bf218527ba36dd085`



© Lukas Aumayr, Zeta Avarikioti, Dimitris Karakostas, Karl Kreder, and Shreekara Shastry;
licensed under Creative Commons License CC-BY 4.0



Funding This work was partially supported by Quai, Common Prefix, the Austrian Science Fund (FWF) through the SFB SpyCode project F8512-N, and the Vienna Science and Technology Fund (WWTF) through the projects 10.47379/ICT25056 and 10.47379/ICT22045, and by Input Output (`https://iohk.io`) through their funding of the Edinburgh Blockchain Technology Lab.

1 Introduction

Proof-of-work (PoW) blockchains pay miners to secure the ledger, and the rule that allocates these payments decides whether honest participation is rational. Selfish mining [7] was the first attack to make this precise: by withholding blocks and releasing them strategically, a miner with a minority of the hashing power can capture a share of the rewards exceeding its share of the power. The lasting contribution of that work was the lens rather than the strategy. It analyzed Bitcoin under proportional, rather than absolute, rewards, and showed that under this utility Bitcoin is not an equilibrium. Subsequent work optimized both the attack and its analysis [20, 9, 15]. Recent events have underscored that these incentives are not merely theoretical. In August 2025, the Qubic mining pool conducted a documented attack against the PoW-based Monero network, causing multiple chain reorganizations and triggering emergency mitigation proposals from the Monero research community [17].

FruitChains [16] answered selfish mining at the level of mechanism design. It guarantees *fairness*: every miner receives rewards approximately proportional to its power, so deviations are unprofitable. The mechanism rewards low-difficulty objects, called fruits, alongside blocks. Fruits are mined far more frequently than blocks and therefore sample the mining-power distribution more finely, so that rewards track a miner's power rather than the variance in block discovery. FruitChains is a ρ-coalition-safe ϵ-Nash equilibrium, the strongest game-theoretic guarantee in this line of work. The guarantee, however, holds only asymptotically. Fairness requires every fruit to be recorded within a window of $R \cdot \kappa$ blocks, with recency parameter $R = 17$ and security parameter κ, which places reward finality on the order of days. For any deployment where reward finality must be fast, this is not a usable regime.

The distance between what is provable and what is deployable was made explicit by Zhang and Preneel [25], who evaluated Nakamoto consensus [14], FruitChains [16], Reward Splitting [25, 13, 4], and Subchains [19] under a single framework. They measured three quantities: (a) incentive compatibility, the fairness of the reward distribution; (b) subversion gain, the profitability of double spending; and (c) censorship susceptibility, the income an attacker can deny honest miners. At realistic parameters, FruitChains was not the best mechanism on most of these metrics. *No prior mechanism in this literature simultaneously achieves asymptotic equilibrium guarantees and state-of-the-art practical performance.*

This paper closes that gap. We present *Proportional Reward Splitting (PRS)*, a reward mechanism that is a ρ-coalition-safe ϵ-Nash equilibrium for large parameters, matching FruitChains in theory, and that achieves state-of-the-art practical performance at a six-block confirmation window, outperforming every mechanism of [25] on incentive compatibility and, with limited exceptions, on subversion gain and censorship susceptibility.

The design begins with separating two roles that existing mechanisms conflate. On the one hand, Reward Splitting [25, 13, 4] makes attacks less profitable by compensating non-canonical work, but it uses a coarse 50–50 split whenever competing objects exist at the same height. On the other hand, FruitChains obtains a finer estimate of mining power from low-work samples, but its theoretical parameters are too large for practical finality. PRS combines these ideas by using low-work objects, which we call *workshares*, only as a local estimator for proportional splitting: it keeps the anti-forking effect of rewarding

competing work, while replacing the coarse split by the measured mining-power distribution. For example, an adversary contributing 30% of the power at a height receives approximately 30% of its reward. This advantage is largest when the adversary's power is meaningfully below one half and when the workshare sample is accurate; as the adversary approaches one half, the proportional split naturally converges to ordinary Reward Splitting.

PRS could be applied to any PoW Nakamoto-style blockchain (e.g., Bitcoin [14], PoEM [12], Monero [22]) by making two changes: blocks record workshares on-chain, and the reward rule is replaced with proportional splitting. Neither change affects the fork choice rule or block production, so security follows directly from the host protocol's guarantees. We call the resulting transformation FairChain. We further provide an optimization in which workshares can be kept off-chain, thereby eliminating any on-chain costs of our transformation. Thus, the storage overhead is minimal: workshare data can be discarded once the corresponding rewards finalize.

To evaluate practical performance, we follow the framework of Zhang and Preneel [25]. For each reward mechanism, we construct a Markov decision process (MDP) whose states encode the adversary's private chain, the public chain, the current fork status, and the recent reward history. Solving this MDP yields the optimal adversarial policy for each utility objective, rather than measuring the performance of a fixed attack strategy.

Under this evaluation, FairChain both increases the mining-power threshold beyond which an adversary profits from deviating and lowers its expected gain from doing so. For incentive compatibility, FairChain requires an adversary to exceed 38% of the power to earn more than its fair share, against 35% for the next-best mechanism, Reward Splitting. At 38% power, an adversary earns 38% of all rewards under FairChain, against 40% under Reward Splitting and more than 50% under Bitcoin or FruitChains. On subversion gain, FairChain matches or beats every alternative across the practical adversarial range, with a strict gap over every other mechanism between 30% and 38% power. On censorship susceptibility, it is the best mechanism above 25% power and within two points of the best below. These advantages hold at every workshare and fork-eligibility window we test.

Finally, we note that the reward mechanism introduced in this paper has been adopted in a deployed system: a live PoW blockchain based on PoEM [12],[1] which records workshares on-chain and allocates rewards proportionally to workshare-estimated mining power. Our discussion reports its workshare cap, freshness window, staleness discount, and resulting sub-kilobyte per-block storage overhead. These parameters ground FairChain in a deployed setting and show that the workshares required by FairChain incur only small overhead.

Summary of Contributions and Roadmap. We make the following contributions:

- We introduce FairChain, a protocol transformation that efficiently incorporates workshares in any PoW Nakamoto-style blockchain without changing the fork choice rule or block production. FairChain uses these workshares to realize PRS, the first reward allocation mechanism for PoW blockchains that is both incentive-compatible and practical for real-world parameters (Section 3).
- We prove that FairChain is a ρ-coalition-safe ϵ-Nash equilibrium for sufficiently large parameters, matching the asymptotic guarantee of FruitChains (Section 4).
- We evaluate FairChain via Markov decision processes under practical parameters, comparing against prior mechanisms on incentive compatibility, subversion gain, and censorship susceptibility (Section 5).

[1] `https://www.qu.ai`

Table 1 Summary of notation.

Symbol	Description
Operators and sequences	
$a.b$	element b of a (complex) object a
$A \,\|\|\, A'$	concatenation of A and A'
$\|\cdot\|$	length of an element (list items or bits)
$L \prec L'$	L is a prefix of L'
Chains and blocks	
$C[i]$	i-th block of a chain C ($C[-1]$ is the tip)
$C[:i],\ C[-i:]$	first (resp. last) i blocks of C
B_{tx}	block in which transaction tx is published
view	execution-trace view
Parties and power	
n	total number of parties
$t = \rho n$	number of adversarial parties
ρ	adversarial mining power, as a fraction of the total
Mining and timing	
Q	hashing-oracle queries per party per round
p	probability that a player mines a block in a round
Δ	network delay, in rounds
w	wait-time (liveness) parameter
k	common-prefix (safety) parameter
Security parameters	
κ	security parameter
λ	output length of $H(\cdot)$, in bits

2 Model and Notation

This section first summarizes the notation (Table 1) and cryptographic primitives and then outlines the execution, cryptographic, and game-theoretic models of our analysis.

2.1 Notation and Cryptographic Primitives

Table 1 summarizes the notation used throughout the paper. Workshare-specific parameters are introduced where they first appear, in Sections 3 and 4.

We rely on two cryptographic primitives. The function $H(\cdot)$ is a hash function that maps an arbitrary bitstring to a value of λ bits and models a random oracle.[2] The function $d(\cdot)$ maps a list of arbitrary bitstrings to a value of λ bits and is collision resistant; it can be instantiated e.g., via a Merkle tree, in which case $d(\cdot)$ returns the tree's root.

[2] H should satisfy the standard hash function properties, namely collision, preimage, and second-preimage resistance.

2.2 Execution Model

Our analysis assumes a multi-party setting, following the model of Bitcoin Backbone [8]. An "environment" program $\mathcal{Z}$ drives the execution of a protocol Π, by spawning for each party $\mathcal{P}$ an instance of an "interactive Turing machine" (ITM) which executes the protocol. Interaction between parties is controlled by a program C, such that $(\mathcal{Z}, C)$ create a system of ITMs. Our analysis applies to "locally polynomially bounded" systems, ensuring polynomial time execution.

Network. The execution proceeds in (a polynomial number of) rounds of size Δ. In each round r, each party is activated and performs various operations based on its algorithm. An honest message produced at round r is delivered to all other parties at the beginning of round $r + 1$, i.e., our analysis assumes a *synchronous network*. Finally, we assume a diffuse functionality, i.e., a gossip protocol which allows the parties to share messages without the need for a fully connected graph.

Parties. The set of n parties $\mathbb{P}$ remains fixed for the duration of the execution. Each party $\mathcal{P}$ can make a fixed number of queries to a hashing oracle, which enables participation in the distributed ledger protocol. The adversary controls at most $t = \rho \cdot n$ parties, out of the total n at any given time, and its power is the sum of the corrupted parties' power.

Proof-of-Work. All parties have access to a random oracle $\mathcal{O}_{\text{rnd}}$. Each party can perform Q queries to the oracle in each round, while the adversary can perform $t \cdot Q$ queries per round. Additionally, the protocol defines a *difficulty parameter* p, which denotes the probability that a player successfully creates a block in a round. In practice, the random oracle is instantiated as a hash function $H(\cdot)$. We assume that the codomain of H is large enough such that the outcomes of queries within a batch are practically independent.

2.3 Cryptographic Model

In the cryptographic model, some parties are assumed honest (i.e., they follow the protocol) whereas the rest are Byzantine (i.e., they act arbitrarily). Note that incentives do not play a role in the security analysis under this model, and instead will be considered in the game theoretic analysis.

Adversary. The adversary A is an ITM which, upon activation, may "corrupt" a number of parties by sending a relevant message to the controller after $\mathcal{Z}$ instructs it to do so. Subsequently, when a corrupted party is supposed to be activated, A is activated instead. Furthermore, A is "adaptive", i.e., corrupts parties on the fly, and "rushing", i.e., decides its strategy and acts after observing the other parties' messages.

Randomized Execution. The execution of a protocol is probabilistic, with randomness coming from both honest parties and the adversary, denoted as A, which controls all corrupted nodes, as well as the environment Z, which provides inputs to honest nodes throughout the protocol's execution.

We denote a randomly sampled execution trace as $\mathsf{view} \xleftarrow{\$} EXEC^{\Pi}(A, Z, \kappa)$, where $|\mathsf{view}|$ represents the number of rounds in the execution trace. Specifically, view is a random variable that captures the joint perspective of all parties, encompassing their inputs, random coins, and received messages (including those from the random oracle). This joint view uniquely determines the entire execution.

Restrictions on the Environment. Constraints on (A, Z) [8]. The environment Z and the adversary A must respect certain constraints. We say that a p.p.t. pair (A, Z) is (n, ρ, Δ)-respecting w.r.t. Π, iff for every $\kappa \in \mathbb{N}$, every view in the support of $EXEC^{\Pi}(A, Z, \kappa)$, the following holds:

- Z activates n parties in view;
- For any message broadcast by an honest player at any time t in view, any player that is honest at time $t + \Delta$ or later must have received the message. This means that in the case of newly spawned players, it instantly delivers messages that were sent more than Δ rounds ago. As long as this Δ constraint is respected, A is allowed to delay or reorder honest players' messages arbitrarily.
- At every round r in view, A controls at most $\rho \cdot n$ parties.

Let $\Gamma(\cdot, \cdot, \cdot)$ be a boolean predicate. We say that a p.p.t. pair (A, Z) is Γ-compliant w.r.t. protocol π iff: (i) (A, Z) is (n, ρ, Δ)-respecting w.r.t. π; (ii) $\Gamma(n, \rho, \Delta) = 1$.

2.4 Game Theoretic Model

In the game theoretic model, we assume that all parties are rational. Specifically, each party $\mathcal{P}$ employs a strategy S, which is a set of rules and actions that the party makes depending on its input. In other words, S defines the part of the distributed protocol that $\mathcal{P}$ performs.

Each party has a well-defined utility U and chooses a strategy S with the goal of maximizing their utility. The literature of game theoretic analyses of blockchain protocols considers various types of utilities [2]: (i) absolute rewards; (ii) absolute profit, i.e., absolute rewards minus cost; (iii) relative rewards, i.e., the percentage of the party's reward among all allocated rewards; (iv) relative profit, i.e., relative rewards minus (absolute) cost; (v) deposits and penalties, e.g., using a slashing mechanism; (vi) external rewards, e.g., bribes.

A strategy profile σ is a vector of all parties' strategies and it is an ϵ-Nash equilibrium if no party can increase its utility more than ϵ by unilaterally changing its strategy (Definition 1).

▶ **Definition 1** (ϵ-Nash equilibrium). *Let $\mathbb{P}$ be the set of parties, $\mathbb{S}$ the set of available strategies, and $\sigma = \langle S_i, S_{-i} \rangle$ a strategy profile, where S_i is the strategy of $\mathcal{P}_i$ and S_{-i} are the strategies of all other parties. Let $U_i(\sigma)$ denote the utility of $\mathcal{P}_i$ under σ. The profile σ is an ϵ-Nash equilibrium with respect to $\bar{U} = \langle U_1, \ldots, U_{|\mathbb{P}|} \rangle$ if*

$$\forall \mathcal{P}_i \in \mathbb{P}\ \forall S_i' \in \mathbb{S}: \quad U_i(\langle S_i, S_{-i} \rangle) \geq U_i(\langle S_i', S_{-i} \rangle) - \epsilon.$$

3 The FairChain Protocol

FairChain is a two-level transformation on any proof-of-work longest-chain protocol. At the *protocol level*, it adds the recording of low-work objects, called *workshares*, on-chain alongside blocks. At the *reward level*, it replaces the existing allocation rule with proportional splitting: each height's reward is distributed among all competing objects at that height in proportion to their intrinsic work. Neither modification touches the host protocol's fork-choice rule or block-production mechanism; as such, FairChain inherits the host protocol's security guarantees.

FairChain Overview. The intrinsic work of any hash-based object obj is

$$\text{WORK}(\mathsf{obj}) = \lambda - \lfloor \log H(\mathsf{obj}) \rfloor, \tag{1}$$

the number of leading zero bits in its hash: the more zero bits, the higher the work. FairChain uses this measure to estimate each miner's relative power via workshares. A workshare has the same structure as a block and is produced by the same mining loop, but satisfies a lower work threshold. Parties propagate workshares through the network and publish them in blocks, as they do with transactions. Since workshares and blocks are outputs of the same hash query, a miner neither chooses between the two nor knows in advance whether a query yields a workshare, a block, or neither; the hash function models a random oracle. Recording workshares is the only modification FairChain makes to the host protocol. Because workshares affect neither fork choice nor block production, FairChain inherits the host protocol's security properties unchanged.

At a high level, in each round a miner runs the PoW loop; a successful query yields a workshare, a block, or both, depending on the leading zero bits of its hash. The miner broadcasts every new object, and whoever mines the next block records the workshares and uncles it has seen inside that block. Once the relevant blocks are stable, the work recorded for a given height, across its block and the workshares pointing to it, determines how that height's reward is split among miners (Section 3.4). Workshares thus serve one purpose: they sample mining power between blocks, so that rewards track power closely.

The accuracy of this power estimate is not a fixed per-block quantity: it depends jointly on the workshare rate, the block-production rate, and the reward-maturity window. A deployment can improve accuracy by increasing the workshare rate, by extending the maturity window, or both, without requiring all samples to appear in a single block.

In the remainder of this section, we describe the protocol's data objects, its core algorithm, the chain-selection and ledger-extraction mechanisms, the reward allocation rule, and an optimization that removes the on-chain storage overhead.

3.1 Blocks and Workshares

A work object in FairChain has the following format:

$$\mathsf{WorkObject} = \langle (d(\mathsf{TX}); d(\mathsf{WS}); h_{B'}; h_{\hat{B}}), \mathsf{ctr} \rangle$$

where TX is the transaction list, WS is the list of work objects (workshares or uncles) published when WorkObject acts as a block,[3] $h_{B'}$ and $h_{\hat{B}}$ reference two prior blocks via their hashes, and ctr is the PoW nonce. A full block is the work object together with the explicit lists TX and WS. When an object is interpreted as a workshare, $d(\mathsf{TX})$ and $d(\mathsf{WS})$ are ignored; those fields are used only by the chain-selection and ledger-extraction mechanisms. Let R denote the recency window: a workshare or uncle is eligible for inclusion only if the block it references is at most R blocks behind the including block.

In more detail, the validity rules for blocks and workshares are as follows. In addition, we assume the existence of a transaction validity predicate, which specifies the application logic of the ledger, e.g., preventing double spending.

Block validity. $B = (\langle (d(\mathsf{TX}); d(\mathsf{WS}); h_{B'}; h_{\hat{B}}), \mathsf{ctr} \rangle, \mathsf{TX})$ is valid w.r.t. a chain C if the following conditions hold:

- $d(\mathsf{TX})$ is the digest of TX.
- $h_{B'}, h_{\hat{B}}$ are references to $C[-1], C[-k]$ respectively.

[3] An uncle satisfies the block work threshold but is not part of the canonical chain, i.e., it has been forked out.

- Every Workobject (workshare or uncle) in WS is valid w.r.t. C (see below);
- No Workobject is present in C multiple times.
- Every transaction in TX satisfies the protocol's validity predicate.
- $\textsc{work}(B) > T_{\text{block}}$, where T_{block} is a protocol parameter that defines the PoW lower bound for a block to be valid.

Uncle validity. $B = (\langle (d(\mathsf{TX}); d(\mathsf{WS}); h_{B'}; h_{\hat{B}}), \mathsf{ctr} \rangle, \mathsf{TX})$ is a valid uncle w.r.t. chain C, if all of the following hold:

- B is included in WS of exactly one block $\overline{B}$ in C and $B \notin C$;
- $h_{B'}$ is a reference to $C'[-1]$, which in turn is either a valid block in C or a valid uncle w.r.t. C;
- $h_{\hat{B}}$ is a reference to $C'[-k]$, which is a valid block in C;
- Let $\overline{C}$ be the prefix of C that goes up to this block $\overline{B}$. $|C'|$ is between $|\overline{C}| - R$ and $|\overline{C}| - 1$;
- Following the reference $h_{B'}$ to a block B', and then this block's reference and so on at most k times, gives a block that is part of C;
- $\textsc{work}(B) > T_{\text{block}}$.

Workshare validity. $\mathsf{ws} = \langle (d(\mathsf{TX}); d(\mathsf{WS}); h_{B'}; h_{\hat{B}}), \mathsf{ctr} \rangle$ is valid w.r.t. chain C if:

- ws is included in WS of exactly one block $\overline{B}$ in C;
- Let $\overline{C}$ be the prefix of C that goes up to this block $\overline{B}$. $h_{B'}$ is a reference to any valid block or uncle with a height between $|\overline{C}| - R$ and $|\overline{C}| - 1$;
- $\textsc{work}(\mathsf{ws}) > T_{\text{ws}}$, where T_{ws} is a protocol parameter that defines the PoW lower bound for a workshare to be valid.

Intuitively, T_{ws} and T_{block} define the minimum amount of intrinsic work that a workshare or block should have, respectively, and it holds that $T_{\text{ws}} \leq T_{\text{block}}$.

Chain validity. Finally, a chain C is valid w.r.t. a genesis block G (denoted $\textsc{Validate}_G(C)$) if every block in C is valid and they form a chain that starts from G.

3.2 Main Protocol

The main function of $\mathsf{FairChain}$ is defined in Algorithm 1. First, the party is constructed using the $\textsc{Constructor}$ function (Line 3). In every round, each party is executed by the environment using the function $\textsc{Execute}$ (this function is due to the round-based nature of our time model). The algorithm maintains a local chain C, which is initialized to the genesis block G, and proceeds as follows.

Chains. The party first receives from the network the set of available chains via the function $\textsc{receive}$ (Line 8). Out of all the available chains, including the local chain C, the party adopts the chain selected by the host protocol's chain-selection rule, denoted $\textsc{MaxValid}$.

Mempool. The party receives the set of transactions in the mempool TX via the function $\textsc{input}$ (Line 10) and attempts to publish them in the newly-mined block. Specifically, the party orders the transactions in TX in an arbitrary manner in order to form the list TX; for ease of notation, we abstract the ordering process outside of the protocol of Algorithm 1. The list's digest $d(\mathsf{TX})$ is included in the new block's header.

Algorithm 1 The FairChain protocol.

1: $G \leftarrow \emptyset$
2: $C \leftarrow [\,]$
3: **function** CONSTRUCTOR(G')
4: $G \leftarrow G'$ ▷ Select Genesis Block
5: $C \leftarrow [G]$ ▷ Add Genesis Block to start of chain
6: round $\leftarrow 1$
7: **function** EXECUTE$_{d,H}(1^\kappa)$
8: $\mathbb{C} \leftarrow$ RECEIVE() ▷ Receive chains from the network
9: $C \leftarrow$ MAXVALID($\{C\} \cup \mathbb{C}$) ▷ Apply host protocol's chain-selection rule
10: $\langle \mathsf{TX}, \mathsf{WS} \rangle \leftarrow$ INPUT()
11: $\mathbb{B} \leftarrow \emptyset$
12: $\mathsf{ctr} \xleftarrow{\$} \{0,1\}^\lambda$
13: **for** $i \leftarrow 1$ to Q **do**
14: $\mathsf{WorkObject} \leftarrow \langle (d(\mathsf{TX}); d(\mathsf{WS}); H(C[-1]); H(C[-k])), \mathsf{ctr} \rangle$
15: **if** WORK(WorkObject) $> T_{\text{ws}}$ **then**
16: $\mathsf{WS} \leftarrow \mathsf{WS} \,\|\, \mathsf{WorkObject}$
17: **if** WORK(WorkObject) $> T_{\text{block}}$ **then**
18: $B \leftarrow \langle \mathsf{WorkObject}, \mathsf{TX} \rangle$
19: $\mathbb{B} \leftarrow \mathbb{B} \,\|\, B$
20: $C \leftarrow C \,\|\, B$
21: $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
22: DIFFUSE($C, \mathsf{WS} \,\|\, \mathbb{B}$)
23: round $\leftarrow$ round+1

Workshares. The party receives from the network workshares WS, including uncle blocks (Line 10), and attempts to publish them in the new block. As opposed to transactions, the order of workshares does not matter, since they are only used for reward allocation (cf. Section 3.4).

We note that TX and WS are both sanitized to contain valid objects w.r.t. the party's local chain C, so that the transactions and workshares can be published in a new block. For ease of notation we omit this process from Algorithm 1 and assume that TX and WS (obtained from INPUT) are sanitized w.r.t. the local chain C at all rounds. In particular, all workshares fulfill the PoW inequality, there are no duplicate workshares in the chain, and uncle blocks are not part of the chain.

Proof-of-Work loop. The main part of the protocol is the PoW loop, during which the party *potentially* creates workshares and blocks. Specifically, on each round the party can execute Q queries to the hash function. For each query, the work object includes a PoW nonce ctr and, depending on the amount of work that the object has, it may be a valid workshare and/or block. If a workshare is created, then it is immediately inserted into the list WS. This has the implication that a workshare which is created on some round r may be published on a block that is created on the same round r. Similarly, if a block is created, it is immediately appended to the local chain C. Regardless of whether a workshare or block is created, the party exhausts all available hashing queries at their disposal without interrupting the PoW loop. Therefore, a party that follows the protocol may create multiple workshares and/or blocks on a single round.

Finally, the party diffuses to the network their local chain and list of workshares and proceeds to the next round.

3.3 Chain Selection and Ledger Extraction

Chain selection follows the host protocol's MaxValid rule. Ledger extraction is parameterized by the safety parameter k. Given a chain C, the last k blocks are excluded from the ledger extraction process. The remaining blocks in C are parsed sequentially, starting from the genesis block G, and the ledger L is formed by concatenating each transaction in the list TX of each block.

3.4 Proportional Reward Splitting Mechanism

We now specify the reward rule of FairChain. At each block height, PRS distributes a fixed reward among all eligible work objects at that height in proportion to their intrinsic work. This refines Reward Splitting [25, 13, 4], which also distributes a height's reward across competing objects but uses a fixed 50-50 split regardless of the work behind them. PRS replaces this coarse split with the measured power distribution: a miner contributing 30% of the work at a height receives 30% of that height's reward.[4] The estimate is recomputed at every height to track fluctuations in mining power as they occur.

Eligibility windows. Two parameters control which objects are eligible for rewards at a given height (Figure 1). The *recency window* R sets the finalization delay: the reward for height h is settled at height $h + R$, and only work objects published within that window are counted. The *fork eligibility window* k limits which forks contribute: an object is eligible only if the block it references lies at most k blocks into a fork, excluding objects that build on deeply invalid chains. In our analysis, k is set to the common-prefix (consistency) parameter T_0 of the host protocol (cf. [1]).

Reward rule. At height $h + R$, each eligible work object at height h receives a fraction of the height's reward equal to its intrinsic work divided by the total intrinsic work of all eligible objects at that height. The height of a workshare or uncle is the height of the block it references via $h_{B'}$, not the block in which it is published. Maturity delays of this kind are standard: Bitcoin requires 100 blocks before a coinbase transaction can be spent.[5]

3.5 Removing on-chain storage overhead

FairChain does not modify the MaxValid fork-choice rule of the underlying proof-of-work chain, so work objects play no role in chain selection, only in reward allocation. Instead of recording them on-chain within blocks, it suffices that miners share them over the P2P network. Miners will verify reward (coinbase) transactions based on the work objects sent along with the block. Once the rewards for height h_i are finalized at height $h_i + R$ (cf. Figure 1), all work objects older than $R + k$ blocks behind the current tip can be safely discarded. Bootstrapping nodes need only download the longest chain; they can skip work object verification entirely. Thus, the storage overhead of FairChain is bounded by the work objects of the last $R + k$ blocks.

[4] The power distribution is not directly observable; PRS estimates it from workshares, as analyzed in Section 5.

[5] `https://learnmeabitcoin.com/technical/mining/coinbase-transaction/#coinbase-maturity`

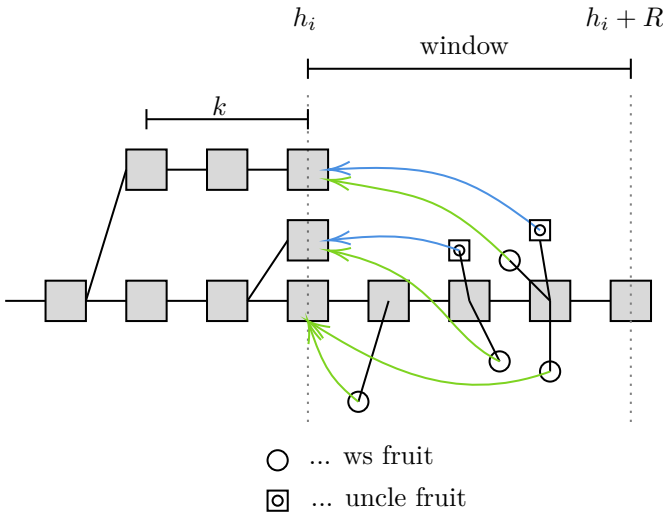


Figure 1 Uncles can be referenced up to R blocks away. Objects pointing to blocks more than k blocks into a fork are ineligible. At height $h_i + R$, the reward at height h_i is distributed proportionally to the intrinsic work of the eligible objects at h_i.

This storage improvement comes with a small tradeoff: a bootstrapping node cannot independently recompute past reward allocations. However, under the honest-majority assumption, honest miners validate reward allocations before extending a block, so a bootstrapping node can still infer their historical validity by following the heaviest PoW chain.

4 Theoretical Analysis

Our analysis closely follows that of FruitChains [16]. The protocol difference mainly affects the analysis of workshare freshness. In the rest of this section, we prove workobject freshness, consistency, growth, and fairness, ultimately leading to incentive compatibility. We denote the probability that any player mines a block in a round as p, and similarly the probability that a player mines a workshare in a round as p_f, such that $q = \frac{p_f}{p}$.

4.1 Main Theorem

Theorem 2 is the main theoretical result of our work, which shows that the workshare-based protocol guarantees the useful properties that we aim to achieve, namely consistency, chain growth, and fairness.

▶ **Theorem 2** (Security of FairChain). *For any constant $0 < \delta < 1$, and any p, p_f, let $R = 16$, $k_f = 2qRk$, and $T_0 = 5\frac{k_f}{\delta}$. Then, in $\Gamma^{p,p_f,R}_{\mathit{FairChain}}$-environments, the Workshare protocol denoted $\Pi_{\mathit{FairChain}}(p, p_f, R)$ satisfies (i) k_f-*consistency*, (ii)* chain growth rate (T_0, g_0, g_1) *with $g_0 = (1-\delta)(1-\rho)np_f$ and $g_1 = (1+\delta)np_f$, and (iii)* fairness (T_0, δ).

We use the standard notions of consistency, chain growth, and fairness (e.g., [16]); their formal definitions, together with the definition of $\Gamma^{p,p_f,R}_{\mathsf{FairChain}}$ and the proofs of Workobject freshness, consistency, and growth, are given in the paper's full version [1].

4.2 Workobject Fairness

Disregard the chain growth, chain quality, Workobject freshness, and Workobject growth failure events, which happen with negligible probability (cf. [1]). Consider some ϕ–fraction player subset selection S, some view view in the support of $EXEC^{\Pi_{\mathsf{FairChain}}(p,p_f,R)}(A,Z,\kappa)$, some round r and some player i that is honest in round r of view. Let $C = b_0,\ldots,b_{|C|}$ be the blocks in the view of i at r, let $f_0,\ldots,f_\ell$ be the Workobjects contained in them, and let $m_0,\ldots,m_\ell$ be the records contained in the Workobjects; let $f_j,\ldots,f_{j+T}$ be the T consecutive Workobjects for some j, where $T \geq \frac{5k_f}{\delta}$.

Let r_0 be the round when the *block* in the view of i at r containing f_{j+k_f} was first added to *some* honest player j_0's chain; let r_1 be the round when the block (again in the view of i at r) containing f_{j+T} was first added to some honest player j_1's chain, and let $t = r_1 - r_0 - 2$ be the number of rounds from r_0+1 to r_1-1. We lower bound the number of S-compatible (honest) Workobjects in the sequence, similar to the proof of Workobject growth *lower bound*:

- By the *Workobject freshness* condition, every Workobject mined by some honest player between (r_0+1) and $(r_1-1)-wait$ will be in the chain of j_1 at some position *pos* that is at least k positions from the end of the chain, before the beginning of round r_1 and will remain so.
- By the Chernoff bound, in the $t-wait$ rounds from r_0+1 to $(r_1-1)-wait$, except with probability $e^{-\Omega((t-wait)\phi np_f)}$, the honest parties in S mine at least $(1-\delta')(t-wait)\phi np_f$ Workobjects (where δ' is some arbitrarily small constant), which thus are all included in the chain of j_1 by r_1-1.
- Since Workobjects are ordered by the block containing them, and since in round r_1 a *new* block is added which contains f_{j+T}, it follows from *blockchain consistency* that all these Workobjects contained in the sequence $f_1,\ldots,f_{j+T}$ (recall that all these Workobjects are found in blocks that are at least k positions from the end of the chain, so by consistency, those blocks cannot change and thus were not added in round r_1 and consequently must come before the block containing f_{j+T}).
- Except with negligible probability (cf. [1]), these Workobjects are also all "new" (i.e., not included in the chains of j_0 at r_0) and different. Since in round r_0, the block containing f_{j+k_f} was added to the chain of j_0, and excluding blockchain collisions (cf. [1]), the chain of j_0 at r_0 up until (and including) the block which contains f_{j+k_f} is a prefix of C, all these Workobjects must in fact be contained in the sequence $f_{j+k_f},\ldots,f_{j+T}$.
- Finally, by Workobject consistency, at r_0 all honest players' Workobject chains contain $f_1,\ldots,f_j$ (since recall that some player added f_{j+k_f} at r_0). Thus, all these Workobjects are S-compatible w.r.t. the prefix $f_1,\ldots,f_{j-1}$ before the T segment we are considering.

We proceed to show that t is sufficiently large. Recall that j_0 is honest at r_0 and j_1 is honest at r_1. We know that at r_1, the Workobject chain contains at least f_{j+T} Workobjects. Additionally, at r_0 the Workobject f_{j+k_f} is added for the first time, so by Workobject consistency, at most $j+2k_f$ Workobjects could have been in the chain of i at this point (since a Workobject at position $j+k_f$ is modified). Thus, the Workobject chain must have grown by at least $T-2k_f$ from r_0 to r_1. By the *upper bound on Workobject growth* we have that: $T-2k_f \leq k_f + (1-\delta')^2 np_f(t+2)$. Thus: $t \geq \frac{1}{(1+\delta')^2 np_f}(T-3k_f)-2$.

We conclude that (except with negligible probability) the number of Workobjects in the sequence is at least:

$$(1-\delta')\phi np_f\Big(\frac{1}{(1+\delta')^2 np_f}(T-3k_f)-2-wait\Big)=$$
$$(1-\delta')\phi\Big(\frac{1}{(1+\delta')^2}(T-3k_f)-np_f(wait+2)\Big)\geq(1-\delta')\phi\Big(\frac{1}{(1+\delta')^2}(T-3k_f)-k_f\Big)\geq$$
$$(1-\delta')\phi\Big(\frac{1}{(1+\delta')^2}(T-4.5k_f)\Big)\geq\phi(T-5k_f)$$

where we use $(wait+2)np_f \leq k_f$, established in the full version of this paper [1], and take δ' sufficiently small. Since $T \geq \frac{5k_f}{\delta}$, we have that $T-5k_f \geq (1-\delta)T$, thus the number of Workobjects in the sequence is at least: $(1-\delta)\phi T$.

4.3 Incentive Compatibility

Our final theoretical result is game-theoretic and concerns utility-maximizing participants.

Any secure blockchain protocol that satisfies δ-approximate fairness (where $\delta < 0.3$) w.r.t $T(\kappa)$ length windows can be used as the ledger underlying a cryptocurrency system while ensuring 3δ-incentive compatibility if players (i.e., miners) only care about how much money they receive.

We say that honest mining is a ρ-coalition-safe ϵ-Nash equilibrium if, with overwhelming probability, no $\rho' < \rho$ fraction coalition can gain more than a multiplicative factor $(1+\epsilon)$ in utility, no matter what transactions are being processed; formally, consider some environment providing transactions into the system. We restrict to a setting where the total rewards and transaction fees during the run of the system are some fixed constant V.

Fairness implies that no matter what deviation the coalition performs, with overwhelming probability, the fraction of adversarial blocks in any $T(\kappa)$-length window of the chain is upper bounded by $(1+\delta)\rho$ and thus the total amount of compensation received by the attacker is bounded by $(1+\delta)\rho \cdot V$; in contrast, by fairness, if the coalition had been following the honest protocol, they are guaranteed to receive at least $(1-\delta)\rho \cdot V$; thus, the multiplicative increase in utility is: $\frac{1+\delta}{1-\delta} \leq 1+3\delta$ when $\delta < 0.3$.

5 Evaluation

We now compare the Proportional Reward Splitting (PRS) mechanism to other reward mechanisms under realistic parameters. In order to perform the comparison, we utilized the Markov Decision Process (MDP) implementation of [25]. MDPs are fitting tools in our case since we want to evaluate the optimal strategies w.r.t. a specific utility. Therefore, when parameterizing our MDPs with realistic values, e.g., in terms of block finality, we can identify the optimal strategy that utility-maximizing rational parties would opt for.

The model does not require knowledge of the identities of parties, or which parties are adversarial, but only assumes that the adversary A holds a certain amount of power. In practice, a protocol's designer could use our analysis to get a lower bound on their protocol's performance, by pinpointing the advantage of the strongest adversary that their protocol can tolerate.

Next, we first outline the construction of the MDP and then compare PRS with other reward mechanisms under three metrics, namely incentive compatibility, subversion gain, and censorship susceptibility. We then evaluate how different values of the PRS's parameters, namely the workshare and fork eligibility windows, affect its performance. Finally, we analyze the accuracy of estimating the adversarial and honest parties' power via workshares and outline the tradeoff between sampling accuracy and storage overhead.

5.1 Description of Markov Decision Process

The MDP implementation of PRS is a direct adaptation of the vanilla Reward Splitting (RS) MDP. Here we give a brief description of the MDP and refer to [25] for more details. Our implementation is available at [10].

We note that our evaluation is not a Monte Carlo simulation with independent runs. For each parameter point, we construct the corresponding finite MDP and solve for the optimal adversarial policy under the selected reward function. Therefore, the reported values correspond to expected long-run rewards induced by the optimal policy, rather than empirical means over repeated executions.

The state of the MDP is a tuple of four elements $\langle l_a, l_c, \text{fork}, \text{history}\rangle$:

- l_a denotes the length of the (private) adversarial chain;
- l_c denotes the length of the honest (public) chain;
- fork takes three values: *active* if there is a tie, i.e., honest miners are split between two chains; *cLast* if the latest block is honest; *aLast* if the latest block is adversarial.
- history is a bitstring as follows: (i) its length represents the number of consecutive attacker blocks in the common (main) chain; (ii) each position corresponds to a height of the main chain, starting from the last block (the tip of the chain); (iii) a 0 bit denotes that no competing honest block exists for the corresponding height, whereas 1 means that an honest block for that height exists.

State transitions occur via the following actions:

- *Adopt*: A adopts the public chain.
- *Wait*: A keeps mining privately and publishes no blocks.
- *Match*: A publishes a chain of private blocks equal to the length of the public chain, causing a tie; this is feasible when $l_a \geq l_c$ and fork = cLast.
- $Override_k$: A publishes a chain of private blocks which is k blocks longer than the public chain.

Finally, the reward allocation is the point of difference between PRS and RS. In RS, when choosing *Adopt*, the honest miners and A each receive half of the rewards for heights where both honest and adversarial blocks exist. Instead, in PRS the rewards are split proportionally to each side's mining power; for instance, if the honest miners (are set to) have 70% of mining power, they receive 70% of the rewards for these heights, instead of 50% as in RS.[6] For the heights where only honest blocks exist, the honest miners get the full reward. Additionally, the adversary receives the full reward for heights where honest objects (blocks and/or workshares) are "pushed out" of the history, i.e., the adversary creates enough consecutive blocks to censor their inclusion.

5.2 Comparison of Reward Mechanisms

We first explore the metrics established in [25], namely: (i) incentive compatibility; (ii) subversion gain; (iii) censorship susceptibility. For each metric, we compare PRS with Bitcoin, FruitChains, and vanilla RS.

In all evaluations, we used the same parameters for the eligibility windows. The first window concerns object eligibility (i.e., the recency window), that is the number of block heights within which an object should be published in order to be eligible for rewards. This

[6] In practice it is impossible to know the exact power distribution between honest parties and the adversary. Instead, an approximation is made using workshares. The accuracy of this approximation is explored in Section 5.4.

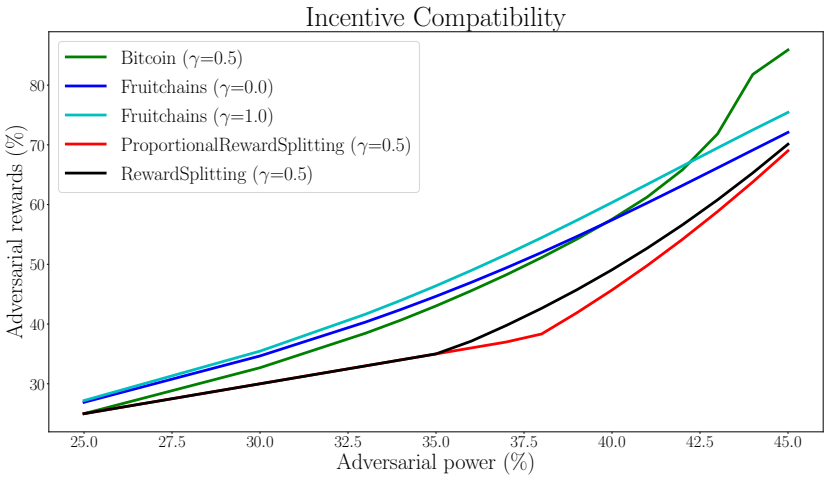


Figure 2 Incentive compatibility for various reward mechanisms. The eligibility window for fruits (in FruitChains) and objects (in Proportional Reward Splitting) is set to $R = 6$. The fork eligibility window for FruitChains, Reward Splitting, and Proportional Reward Splitting is set to $k = 6$. Larger values indicate worse performance.

window was set to $R = 6$. The second window concerns fork eligibility, that is the depth of a fork up to which objects (that point to blocks of that depth) are eligible for rewards. This window was set to $k = 6$.

The parameter γ, which indicates the percentage of honest miners that adopt the adversarial forks, is set to 0.5 for Bitcoin, RS, and PRS, and 0 and 1 for FruitChains.[7] In essence, when two chains of the same height are available, $\gamma = 0.5$ implies that honest miners choose between competing chains of equal height uniformly at random.

Following the baseline configuration of [25], we set the fruit-to-block ratio to 1. While higher fruit rates improve the incentive compatibility of FruitChains, at the cost of increased bandwidth, [25] observes only small improvements for higher ratios and finds RS to outperform FruitChains in its comparison.

Incentive Compatibility. The first metric of comparison is incentive compatibility (IC). Briefly, IC expresses the percentage of rewards that the adversary gains compared to its fair share, that is, its relative mining power. Figure 2 depicts the IC for the reward mechanisms under question. In essence, values above the line $x = y$ indicate that the adversary gains disproportionately more rewards, compared to its mining power, so larger values indicate worse IC performance.

PRS outperforms all other mechanisms on IC. For adversarial power up to 25%, all mechanisms except FruitChains are optimal. RS and PRS both remain optimal through 35%; beyond that, PRS strictly outperforms RS.

Subversion Gain. Subversion gain indicates the profitability of double-spending attacks. In particular, the profit from a double-spending attack is time-averaged and set to 3 times the block reward. In other words, if the adversary manages to orphan k blocks during the

[7] For FruitChains we use both 0 and 1 because the FruitChains MDP implementation of [25] did not enable using 0.5 due to complexity issues.

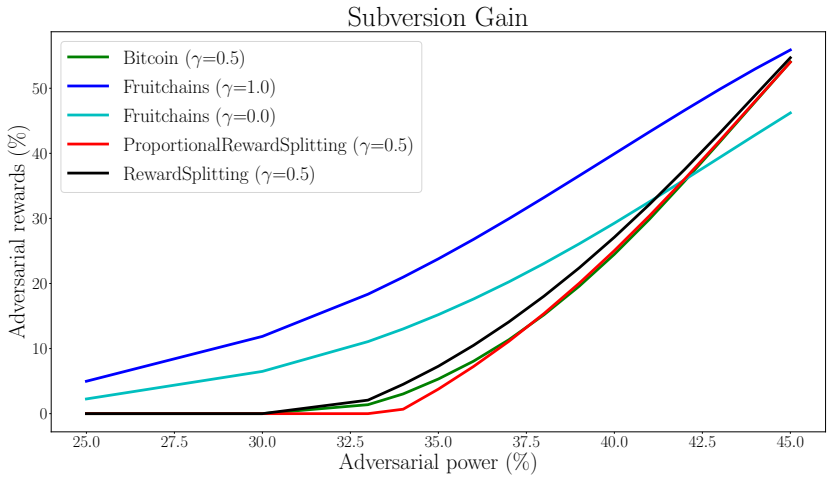


Figure 3 Comparison of subversion gain for various reward mechanisms. The eligibility window for fruits (in FruitChains) and objects (in Proportional Reward Splitting) is set to $R = 6$. The fork eligibility window for FruitChains, Reward Splitting, and Proportional Reward Splitting is set to $k = 6$. Larger values indicate worse performance.

attack, it is granted $3 \cdot k$ rewards. The confirmation time is set to 6 blocks, so the attack is successful if the adversary reverts 6 blocks in order to successfully double-spend the assets. If the attack is unsuccessful, the adversary gets no rewards, but also incurs no penalty.

Figure 3 depicts the comparison of the reward mechanisms w.r.t. subversion gain. For the most part, PRS is again the better option. In particular, up to 30% all mechanisms except FruitChains perform optimally, i.e., the adversary has zero gain. For the range of adversarial power between 30% and 38% PRS is the best option, whereas between 38% and 42% it is in effect equally good to Bitcoin. Above 42%, FruitChains with $\gamma = 0$ performs better, which is a result of the particularly favorable setting (compared to $\gamma = 0.5$ for the other mechanisms); instead, under $\gamma = 1$, FruitChains is strictly worse than all others for all adversarial powers.

In summary, PRS is optimal w.r.t. subversion gain and, in some cases, it is strictly better than all other options.

Censorship Susceptibility. Censorship susceptibility indicates the maximum fraction of income loss that the adversary can incur on honest miners under the threat of a censorship attack. Specifically, the adversary's reward corresponds to the honest miners' loss in the form of orphaned blocks.

Figure 4 depicts the results for censorship susceptibility. These are consistent with [25], wherein FruitChains outperforms the other mechanisms for smaller adversaries but becomes relatively worse for larger values. Interestingly, PRS strictly outperforms vanilla RS for all adversarial powers. As a result, FruitChains is the best option only for adversarial power up to 25%, after which point PRS is the best option. Notably, even for smaller values of adversarial power, the difference between PRS and FruitChains is less than 2%.

5.3 Comparison of Eligibility Parameters

We now evaluate how the different eligibility parameters affect the performance of the reward mechanisms. In particular, we will compare the two types of reward splitting w.r.t. the fork and the workshare eligibility windows. The metric of comparison is incentive compatibility, since all metrics are affected in the same manner when changing the parameters. As before, we set γ to be 0.5.

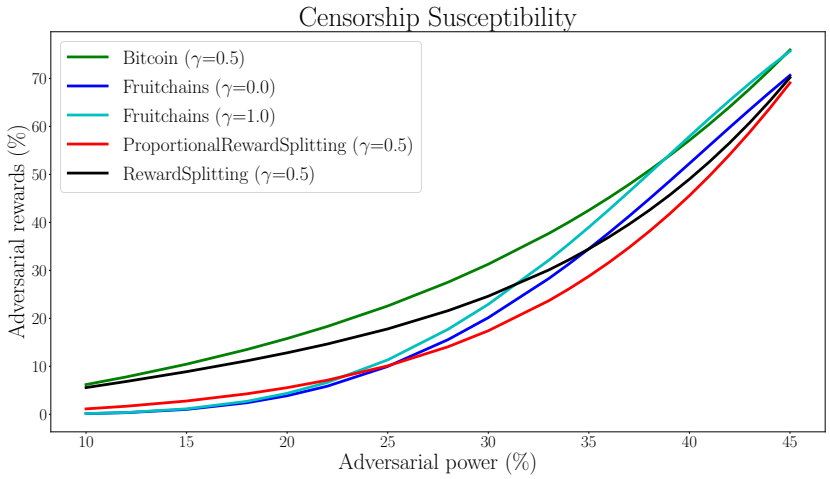


Figure 4 Comparison of censorship susceptibility for various reward mechanisms. The eligibility window for fruits (in FruitChains) and objects (in Proportional Reward Splitting) is set to $R = 6$. The fork eligibility window for FruitChains, Reward Splitting, and Proportional Reward Splitting is set to $k = 6$. Larger values indicate worse performance.

Workshare Eligibility Window. The first parameter is the workshare eligibility window R. This is the distance between the block that a workshare references and the block within which it gets published. Intuitively, larger window values require the adversary to censor the chain for longer periods of time, in order to force an honest workshare to become ineligible, thus making the attack harder. However, they also incur a delay on the finalization of objects and, consequently, the allocation of rewards.

Figure 5 depicts how the two reward splitting mechanisms perform for three workshare eligibility window values, $R \in \{3, 6, 9\}$. Evidently, Proportional Reward Splitting outperforms vanilla Reward Splitting in all cases. As expected, larger values of R result in better performance, that is, lower adversarial rewards. Consequently, PRS is optimal for larger values of adversarial power; when $R = 3$, PRS is optimal for adversarial power up to 33%, whereas for $R = 9$ this goes up to 38%.

Fork Eligibility Window. The second parameter is the fork eligibility window k. This window sets the depth of a fork up to which objects are eligible for rewards. Specifically, if $k = 3$, then objects that point to a block in a fork are eligible for rewards only if this block is among the first 3 of the fork. Note that the height of the workshare, which is considered for the workshare eligibility window, is computed as the height of the referenced block. For example, if a workshare points to the second block of a fork which occurred at height 100, then the workshare's height is 102 and, in order to be eligible for rewards, it should be published in a (main chain) block up to height $102 + R$.

Figure 6 depicts the effect of the fork eligibility window on incentive compatibility. Evidently, lower values of k result in better performance, i.e., lower adversarial rewards. One explanation for this could be that lower values of k penalize forking more, so the adversary is incentivized to abandon forks more often, hence its rewards get closer to its fair share.

Interestingly, we observe that PRS outperforms RS even for different values of k. Specifically, PRS under $k = 6$ performs better (for the most part) than RS under $k = 1$.

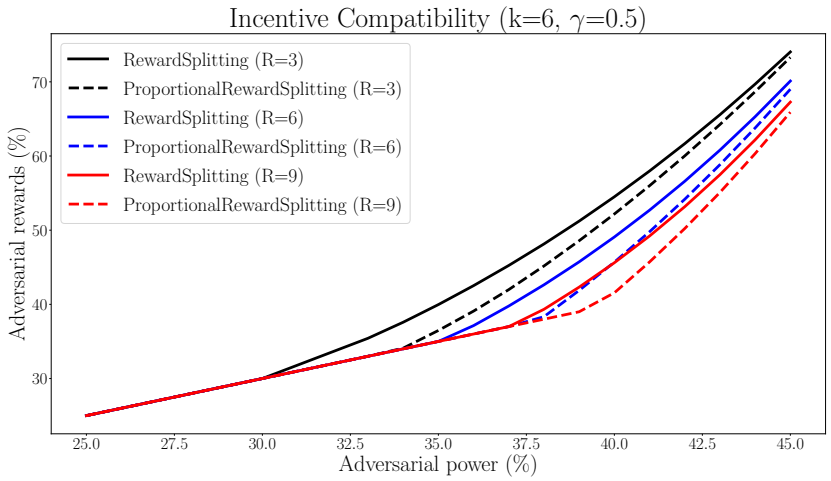


Figure 5 Incentive compatibility w.r.t. the workshare eligibility window between Reward Splitting and Proportional Reward Splitting. The fork window for objects is set to $k = 6$. Larger values indicate worse performance.

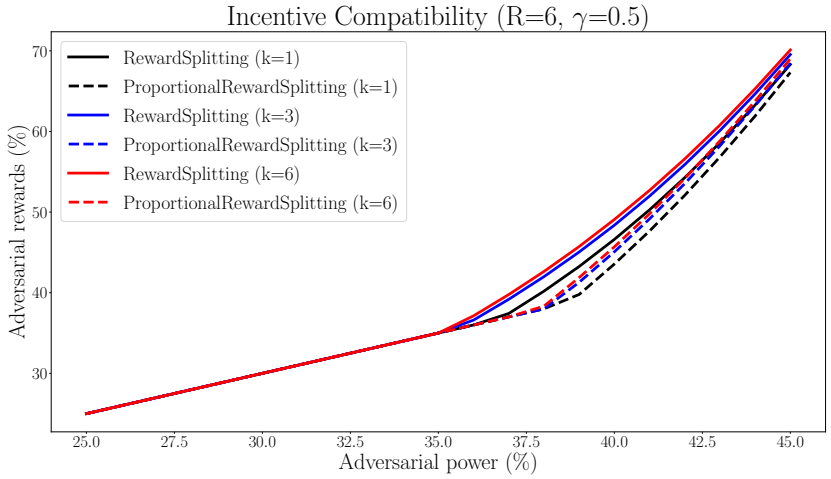


Figure 6 Incentive compatibility for various values of the fork window between Reward Splitting and Proportional Reward Splitting. The eligibility window for objects is set to $R = 6$. Larger values indicate worse performance.

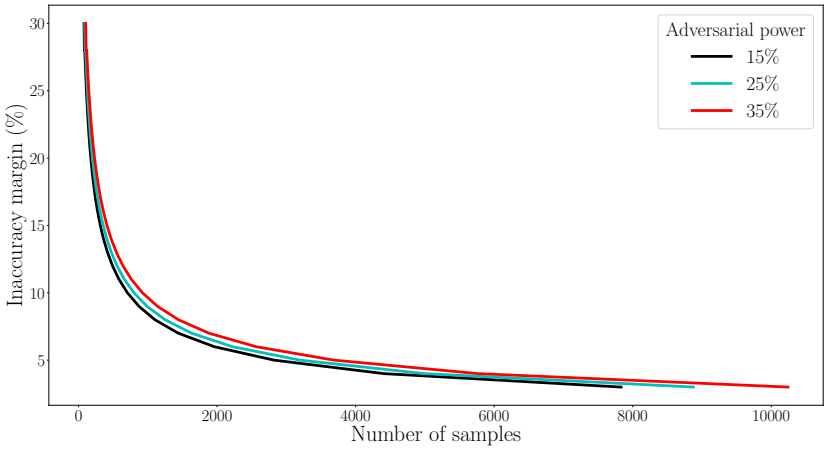


Figure 7 Number of workshares needed to estimate the honest/adversarial power distribution, at error bound $\epsilon = 0.05$. Inaccuracy is the acceptable deviation from the ideal value.

5.4 Sampling Accuracy

So far, our evaluation of the proportional reward splitting mechanism assumes a perfect knowledge of the distribution of power between the honest parties and the adversary. In practice, this is not possible to achieve for proof-of-work (PoW) ledgers, as each party's proportional power is not visible to anyone, not even to the party itself. Although a party knows its *absolute* power, they do not know the exact amount of power of other parties.

Instead, the distribution of power can be inferred from observed events, specifically block creations. Since the probability that each party creates a block on each round is proportional to their power and each such event is independent, a large enough sample of blocks enables inferring the power distribution.

A pertinent question is how many samples are needed to make an accurate enough estimation, with a low margin of error. In our setting, the samples correspond to workshares and blocks. Therefore, the question of interest is how many workshares should be published per height, such that each block's rewards are distributed proportionally to each side's power, accurately and in most cases (low margin of error).

To estimate the number of workshares needed, we model the generation of workshares as a series of independent Bernoulli trials. Let $X_i \in \{0, 1\}$ be the i-th sample of random variable X, where 0 corresponds to an adversarial workshare creation and 1 corresponds to honest workshare. The probability of 1 for each X_i is p_h, so $\{X_i\}_{i\in[n]}$ is independent and identically distributed to a Bernoulli trial $\mathsf{Bern}(p_h)$.

Let $X = \sum_{i=1}^{n} X_i$ be the random variable that represents the sum of the observed values, i.e., the number of honest workshares. It holds that $E[X] = n \cdot p_h$, where n is the number of samples (i.e., Bernoulli trials) and p_h is the probability of creating an honest workshare.

By the Chernoff bound: $\Pr[X \leq (1-\delta) \cdot E[X]] < \epsilon = e^{-\frac{\delta^2 \cdot n \cdot p_h}{2}}$. where: (i) δ denotes the acceptable deviation from the completely accurate (expected) value; (ii) ϵ denotes the error bound of the observation, i.e., the probability that the observed value is less accurate than δ. By solving for n we find, for a given honest-power fraction p_h, the minimum number of workshares needed to get an accurate enough observation with low error probability: $n = -\frac{2 \cdot \log \epsilon}{\delta^2 \cdot p_h}$.

Figure 7 depicts how n is affected given various values of the other parameters. Here, we set the error bound to 0.05; in other words, the probability that the observed distribution between the honest and adversarial parties is less accurate than the acceptable level of inaccuracy is 5%.

The x-axis corresponds to the number of workshares, while the y-axis corresponds to the acceptable level of inaccuracy. We observe that inaccuracy drops exponentially w.r.t. the number of samples used. Additionally, inaccuracy drops at a faster rate when the adversarial power is lower. We assume that each workshare is equivalent to a Bitcoin block's header, which amounts to 80 bytes. Therefore, the overhead *per block* increases linearly with the number of workshares. As we explain in Section 3.5, however, this storage is off-chain and can be discarded after finalizing the rewards.

For example, assume adversarial power is 15%. 7,832 workshares achieve 3% inaccuracy (with error probability 5%), requiring 612 KB per block. Similarly, 1,000 workshares achieve approx. 8.5% inaccuracy at 79 KB. On the other hand, assuming 35% adversarial power, 3% inaccuracy is achieved with 10,242 workshares, which corresponds to 800 KB.

6 Discussion and Deployment

PRS in theory. PRS matches FruitChains in theoretical guarantees but outperforms it in practical settings. This is because the theoretical analysis evaluates the protocols *asymptotically*, for (very) large parameters. In practice, parameters like freshness or the fork eligibility window cannot be too large without delaying transaction and reward finalization.

At realistic parameter values, the theoretical guarantees no longer apply, so we turn to the MDP evaluation. One explanation for why PRS outperforms FruitChains lies in the tradeoff between "Rewarding the Bad" and "Punishing the Good" identified in [25]. FruitChains eliminates the incentive to fork by rewarding the adversary for failed attempts. PRS does the same through workshares, but to a lesser degree: it splits rewards proportionally rather than yielding the full contested reward to the adversary. This finer split strikes a better balance in practice.

PRS in practice. PRS outperforms Reward Splitting (RS) in practical terms. Both mechanisms follow the same punishment strategy, that is, splitting the rewards, but PRS does so in a more fine-grained way, which better approximates the distribution of power between the adversary and the honest parties. In other words, the gain of PRS comes from the difference between 50% and the actual value of adversarial power; hence, when the latter approaches the former, the two lines converge.

Dynamic difficulty. Our theoretical analysis follows the methodology of FruitChains [16], which operates in the static-difficulty setting; we therefore do not establish formal guarantees for variable-difficulty executions. We expect the same intuition to apply in this setting, similarly to variable-difficulty analyses of accumulated-work protocols such as NIPoPoWs: when the target changes, work should be interpreted relative to the target that was active when the object was created. In a deployment, the workshare target would be recalculated using the same difficulty-adjustment rule as the main block target, so that workshares continue to provide a fresh sample of the mining-power distribution. A formal analysis of PRS under variable difficulty is left for future work.

Deployment of FairChain. Our FairChain transformation has been deployed in practice as part of a Proof-of-Work blockchain launched in January 2025 [5] that uses PoEM as its fork-choice rule. The live blockchain records workshares on-chain with a hard cap of 9 workshares per block and a soft target of 3, where each workshare header is approximately 120 bytes, which is comparable to a standard transaction header. Workshares older than 2 blocks

(freshness parameter) are rejected by peers. Rewards are paid proportionally to each miner's workshare-estimated power, with a linear staleness discount applied to shares approaching the freshness timeout, incentivizing timely propagation. At the current parameters, the storage overhead amounts to under 1 KB per block above baseline.

7 Related Work

The work most directly related to FairChain is FruitChains [16]. FruitChains was shown to be a ρ-coalition-safe ϵ-Nash equilibrium, i.e., any coalition of parties with a fraction of power less than ρ cannot gain more than ϵ of their fair share of rewards. As shown in Section 4, our mechanism achieves the same guarantee, as the analysis of FruitChains carries over in our case almost directly. Nonetheless, FairChain outperforms FruitChains in practice, that is, for parameter values smaller than the theoretical, unrealistic, bounds (cf. Section 5).

Recording low-work objects on-chain was explored in StrongChain [21]. StrongChain publishes workshares on-chain, as opposed to keeping them within a mining pool and invisible to the protocol. The paper offers a protocol implementation and a brief discussion of the protocol's performance in the optimistic setting, where all parties are honest. However, StrongChain's analysis is lacking in many aspects. Specifically, StrongChain offers only a high-level description of reward allocation and an intuition of why this allocation is beneficial. In comparison, our mechanism is analyzed both in theory, in a game-theoretic setting where parties are assumed rational instead of simply honest, and in practice via Markov decision processes. Moreover, StrongChain's analysis considers only a specific instance of a selfish mining strategy, and analyzes the protocol's performance under various parameter values, whereas our MDP analysis evaluates the performance of the optimal adversarial strategy.

Another line of work on recording low-work objects on-chain builds on the idea of "weak blocks". These works include Subchains [19] and Flux [23]. Here, the main idea is the formation of "sub-chains", that is, chains of objects with less PoW than blocks. Therefore, between two consecutive (heavy) blocks, a sub-chain of weak blocks may be created, enabling a more fine-grained estimation of the power distribution among participants. Flux offered a simulation-based analysis, showing that it performs better w.r.t. selfish mining compared to Bitcoin. However, as was shown in [25], Subchains performs worse in practice compared to Reward Splitting (RS). In comparison, our mechanism outperforms RS, and thus also Subchains, under the framework of [25].

The uncle reward system used in Ethereum [3] prior to its transition to Proof-of-Stake rewarded non-canonical work via *uncle blocks*. Ritz and Zugenmaier [18] show empirically that such a fixed partial reward is insufficient: it actually *lowers* the mining power threshold at which selfish mining becomes profitable, compared to Bitcoin, rather than eliminating the incentive. The core issue is that a fixed reward subsidizes the adversary regardless of their actual mining power. FairChain avoids this by splitting rewards proportionally to the power distribution, ensuring the adversary cannot benefit beyond their fair share.

Finally, FairChain can be instantiated on any longest-chain protocol that guarantees safety and liveness, including Nakamoto Consensus [14], uniform tie-breaking [6, 7], smallest-hash tie-breaking [13], unpredictable deterministic tie-breaking [11], Publish-or-Perish [24], and PoEM [12]. The security analysis requires only that the host protocol satisfies these two properties; no other assumptions on the fork-choice rule are needed.

References

1 Lukas Aumayr, Zeta Avarikioti, Dimitris Karakostas, Karl Kreder, and Shreekara Shastry. Optimal reward allocation via proportional splitting, 2025. doi:10.48550/arXiv.2503.10185.

2 Sarah Azouvi and Alexander Hicks. Sok: Tools for game theoretic models of security for cryptocurrencies. *CoRR*, abs/1905.08595, 2019. arXiv:1905.08595.

3 Vitalik Buterin. Ethereum: A next-generation smart contract and decentralised application platform. https://ethereum.org/content/whitepaper/whitepaper-pdf/Ethereum_Whitepaper_-_Buterin_2014.pdf, 2014.

4 Philippe Camacho and Sergio Demian Lerner. Decor+ lami: A scalable blockchain protocol. In *Scaling Bitcoin Workshops*, 2016.

5 Dominant Strategies. go-quai: Official Go implementation of the Quai protocol. https://github.com/dominant-strategies/go-quai, 2025. Accessed: 08-2026.

6 Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert van Renesse. Bitcoin-ng: A scalable blockchain protocol. In Katerina J. Argyraki and Rebecca Isaacs, editors, *13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2016, Santa Clara, CA, USA, March 16-18, 2016*, pages 45–59. USENIX Association, 2016. URL: https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/eyal.

7 Ittay Eyal and Emin Gün Sirer. Majority is not enough: bitcoin mining is vulnerable. *Commun. ACM*, 61(7):95–102, 2018. doi:10.1145/3212998.

8 Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 281–310. Springer, 2015. doi:10.1007/978-3-662-46803-6_10.

9 Arthur Gervais, Ghassan O. Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 3–16. ACM, 2016. doi:10.1145/2976749.2978341.

10 Dimitris Karakostas. commonprefix/proportional-reward-splitting-MDP. Software, swhId: swh:1:dir:4a244dd0f68a10c4abb1fa0bf218527ba36dd085 (visited on 2026-08-19). URL: https://github.com/commonprefix/proportional-reward-splitting-MDP.

11 Eleftherios Kokoris-Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, pages 279–296. USENIX Association, 2016. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/kogias.

12 Karl Kreder, Shreekara Shastry, Apostolos Tzinas, Sriram Vishwanath, and Dionysis Zindros. A better proof-of-work fork choice rule. Cryptology ePrint Archive, Paper 2024/200, 2024. URL: https://eprint.iacr.org/2024/200.

13 Sergio Demian Lerner. Decor+ hop: A scalable blockchain protocol. *Semantic Scholar*, 2015.

14 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

15 Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, pages 305–320. IEEE, 2016. doi:10.1109/EUROSP.2016.32.

16 Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In Elad Michael Schiller and Alexander A. Schwarzmann, editors, *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*, pages 315–324. ACM, 2017. doi:10.1145/3087801.3087809.

17 retrodrive. Qubic performs 51% monero network takeover demonstration. https://qubic.org/blog-detail/historic-takeover-complete-qubic-miners-now-secure-monero-network, 2025. Accessed: 08-2026.

18 Fabian Ritz and Alf Zugenmaier. The impact of uncle rewards on selfish mining in ethereum. In *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 50–57, 2018. doi:10.1109/EuroSPW.2018.00013.

19 Peter R. Rizun. Subchains: A technique to scale bitcoin and improve the user experience. *Ledger*, 1:38–52, 2016. URL: https://ledgerjournal.org/ojs/index.php/ledger/article/view/40.

20 Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In Jens Grossklags and Bart Preneel, editors, *Financial Cryptography and Data Security - 20th International Conference, FC 2016, Christ Church, Barbados, February 22-26, 2016, Revised Selected Papers*, volume 9603 of *Lecture Notes in Computer Science*, pages 515–532. Springer, 2016. doi:10.1007/978-3-662-54970-4_30.

21 Pawel Szalachowski, Daniël Reijsbergen, Ivan Homoliak, and Siwei Sun. Strongchain: Transparent and collaborative proof-of-work consensus. In Nadia Heninger and Patrick Traynor, editors, *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, pages 819–836. USENIX Association, 2019. URL: https://www.usenix.org/conference/usenixsecurity19/presentation/szalachowski.

22 Nicolas van Saberhagen. Cryptonote v 2.0. https://github.com/monero-project/research-lab/blob/master/whitepaper/whitepaper.pdf, 2013. Accessed: 08-2026.

23 Alexei Zamyatin, Nicholas Stifter, Philipp Schindler, Edgar R. Weippl, and William J. Knottenbelt. Flux: Revisiting near blocks for proof-of-work blockchains. *IACR Cryptol. ePrint Arch.*, page 415, 2018. URL: https://eprint.iacr.org/2018/415.

24 Ren Zhang and Bart Preneel. Publish or perish: A backward-compatible defense against selfish mining in bitcoin. In Helena Handschuh, editor, *Topics in Cryptology - CT-RSA 2017 - The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14-17, 2017, Proceedings*, volume 10159 of *Lecture Notes in Computer Science*, pages 277–292. Springer, 2017. doi:10.1007/978-3-319-52153-4_16.

25 Ren Zhang and Bart Preneel. Lay down the common metrics: Evaluating proof-of-work consensus protocols' security. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 175–192. IEEE, 2019. doi:10.1109/SP.2019.00086.

Friend or Foe?
Anomalous Peers in Monero's P2P Network

Yannik Kopyciok
Technische Universität Berlin, Germany

Stefan Schmid
Technische Universität Berlin, Germany

Friedhelm Victor
TRM Labs, San Francisco, CA, USA

Abstract

Monero, the leading privacy-focused cryptocurrency, relies on a peer-to-peer (P2P) network to propagate transactions and blocks. Growing evidence suggests that non-standard nodes exist in the network, posing as honest nodes but are perhaps intended for monitoring the network and spying on other nodes. However, our understanding of the detection and analysis of anomalous peer behavior remains limited. This paper presents a systematic methodology for detecting anomalous peer behavior in P2P cryptocurrency networks, which we apply in a forensic case study of Monero's network. We collected and analyzed over 240 hours of network traffic captured from five distinct vantage points worldwide. We further present a formal framework which allows us to analytically define and classify anomalous patterns in P2P cryptocurrency networks. Our detection methodology defines detection rules directly applicable to real-time monitoring. Our analysis reveals the presence of non-standard peers in the network where approximately 14.38% (1,877 of 13,050) of the observed peers exhibit anomalous behavior. Of these, at least 1,582 peers show patterns consistent with control by a single entity. These peers exhibit behavioral patterns consistent with a Sybil attack coupled with self-reinforcing peer list poisoning, which establishes the preconditions for degrading Monero's privacy guarantees and undermining network decentralization. To support reproducibility and enable network operators to protect themselves, we release our examination pipeline to identify and block suspicious peers based on newly captured network traffic.

2012 ACM Subject Classification Networks → Network privacy and anonymity

Keywords and phrases network forensics, security, cryptocurrency networks

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.31

Related Version *Full Version*: `https://doi.org/10.48550/arXiv.2509.10214`

Supplementary Material *Software*: `https://github.com/ykpyck/monero-traffic-analysis` archived at `swh:1:dir:a1913201f79277bf2bd8f1107ef12c594ca5910d`

Funding Supported by German Research Foundation (DFG), SPP 2378/ReNO-2 (grant 511099228), 2025-2029.

1 Introduction

The security and privacy of cryptocurrencies critically depends on the underlying P2P networks as adversaries exploiting network-level vulnerabilities can bypass security and privacy mechanisms applied at the protocol and application layer, risking the network's integrity even for privacy-focused cryptocurrencies. Ten years after its release, Monero continues to be the dominant privacy coin. With a 24-hour trading volume exceeding


© Yannik Kopyciok, Stefan Schmid, and Friedhelm Victor;
licensed under Creative Commons License CC-BY 4.0

100 million USD[1], Monero demonstrates that users increasingly demand financial privacy comparable to physical cash in traditional monetary systems. Simultaneously, empirical evidence has demonstrated a notable migration from Bitcoin to Monero in illicit online marketplaces [2].

While general blockchain analysis platforms exist [11] Monero's privacy techniques have demonstrated robust protection against sophisticated blockchain analysis [9, 14]. Consequently, rather than breaking cryptographic primitives, adversaries appear to be increasingly targeting the underlying P2P network layer [4, 5] to exploit network-level vulnerabilities that undermine claimed privacy guarantees. In 2023, community developers identified a few highly saturated /24 subnets, each hosting close to the full 254-host range of peers, opening many short-lived connections in the Bitcoin network as well as the Monero P2P network, with this pattern continuing into 2024[2]. In September of 2024, an independent community analysis[3] followed, discussing the potential intrusion of non-standard monitoring nodes within the P2P network. Such monitoring nodes would be positioned to observe RPC calls, potentially linking invoked transactions to real-world IP addresses, a capability that directly undermines Monero's privacy guarantees. This capability not only defeats Monero's privacy promise but may also lower the bar for targeted attacks such as focused eclipse attacks against specific users, although eclipse feasibility depends on the details of peer discovery and connection management and several concrete vectors have since been closed. By late 2024, these concerns prompted Monero developers to issue an official warning[4] and recommend node operators make use of a protective ban list. However, no public explanation was provided of how this ban list was compiled or what criteria were used to identify malicious nodes, creating a significant gap in the understanding of this network-level threat.

A variety of previously studied attacks on blockchain P2P networks [1, 4, 5, 10, 18, 21] show that whilst attacks are feasible, an adversary first needs to infiltrate the network with peers frequently diverting from standard configurations. Detection of such non-standard peers remains limited (Sec. 2), and our work fills this gap by systematically characterizing these subliminal network intrusions.

Contributions. We investigated Monero's P2P network for about six months with repeated captures of network traffic and analysis cycles. This preliminary analysis revealed that detection cost differs sharply across anomaly classes. Some deviations are visible in the first message a peer sends and others only in the timing of a single connection over its first minutes. Others emerge only once compared against one another. Our final measurement design accommodated the slowest of these. This paper presents a forensic case study, based on a two-day, five-vantage-point traffic capture from June 2025. To our knowledge, this is the first work to systematically detect passive, in-protocol monitoring nodes in a cryptocurrency P2P network, rather than the disruptive attackers targeted by prior detection systems. Our detection angles are tailored to the cryptocurrency setting and exploit Monero's presumed software homogeneity, under which any systematic deviation becomes a detectable fingerprint. Our systematic examination framework introduces a novel classification: protocol-level anomalies (Sec. 4.1) categorized into syntactic violations, identifying evident deviations from protocol specifications; semantic inconsistencies, going beyond structural validation

[1] https://coinmarketcap.com/currencies/monero/
[2] https://b10c.me/blog/013-one-year-update-on-linkinglion
[3] https://www.digilol.net/blog/chainanalysis-malicious-xmr.html
[4] https://github.com/monero-project/meta/issues/1124

and analyzing logical consistency; and behavioral deviations, including a temporal context separating standard state sequences from anomalous behavior. Furthermore, we advance this peer-based taxonomy by including network structural patterns (Sec. 4.2) and node attributes (Sec. 4.3) commonly found in P2P environments. Despite Monero's P2P network presuming homogeneous node software deployment across all participating peers, our examination (Sec. 5) revealed that approximately 14.38% of peers observed during our measurement period exhibited anomalous behavior, with one entity estimated to control at least 1,582 of the observed nodes. These nodes distort Monero's peer discovery mechanism to gain strategic network positions, enabling an effective first-spy estimator. Our anomaly detection approach complements community efforts by providing a comprehensive framework with empirical evidence for identifying non-standard nodes, raising concerns about Monero's privacy guarantees and network decentralization. To ensure reproducibility and to facilitate follow-up work, we release our GitHub repository and dataset, allowing every Monero node operator to replicate our results and analyze their own network traffic, identify unusual peers, and block them if desired.

2 Related Works

Prior work on Monero and related P2P layers spans topology inference, transaction deanonymization, and eclipse attacks, alongside analogous vulnerabilities in Bitcoin and Ethereum.

Monero's Network Attacks. Cao et al. [5] demonstrated a topology learning attack by exploiting `last_seen` timestamps in Handshake and Timed Sync responses, achieving 97.98% recall for inferring node connections across crawled data. In response, developers removed the timestamp [5] from Peer Lists (PLs), neutralizing the attack while leaving network centralization unaddressed. Shi et al. [20] later correlated transactions with IP addresses by injecting malicious Monero Tor hidden service nodes and watermarking Tor circuits to link onion and real addresses. Shi et al. [21] then demonstrated the first practical eclipse attack against Monero, combining peer-discovery flooding, priority-list population, and double-spend-induced connection drops to eclipse a victim within minutes using 1,020 IPs. A 2024 update addressed the reset vector [6], though new vectors may emerge.

Network-Level Attacks on Bitcoin and Ethereum. Eclipse attacks are well studied here. Heilman et al. [10] monopolized a target's peer list and waited for a restart to achieve isolation; Tran et al. [22] introduced Erebus, a stealthier AS-level attack replacing TCP connections without direct occupation; and Marcus et al. [13] forced restarts and flooded incoming connections in a low-resource variant. These works highlight the persistent vulnerability of P2P networks to network-layer isolation.

Anomaly Detection in P2P Networks. Sybil attacks were formalized by Douceur [6], who showed that without a central identity authority they are unpreventable in open systems, and have since been measured in deployed P2P networks. Wang and Kangasharju [23] identified large-scale Sybil deployments in BitTorrent's Mainline DHT precisely by their partial protocol implementation, and Neudecker and Hartenstein [15] survey the corresponding network-layer design space for permissionless blockchains. These works detect Sybils through address concentration, active crawling, or honeypots. We instead fingerprint per-packet and per-connection protocol behavior observed passively from peers that connect to us.

[5] `https://github.com/monero-project/monero/pull/5682`
[6] `https://github.com/monero-project/monero/pull/9218`

3 Background

Monero differs from generic P2P networks as there is effectively one maintained client. Within its P2P network, Monero aims for homogeneous-looking traffic as core fungibility design goal, any systematic deviation requires deliberate code modification. Throughout this paper, we use three related but distinct terms with increasing evidential burden. We call a node non-standard if it deviates from the Monero specification at any layer, including protocol fields, message timing, or software configuration. We call a node's behavior anomalous when it produces observable patterns that deviate from what a standard node would produce. Finally, we reserve the terms malicious and adversarial for cases where corroborating evidence supports an intentional, hostile interpretation of non-standard behavior.

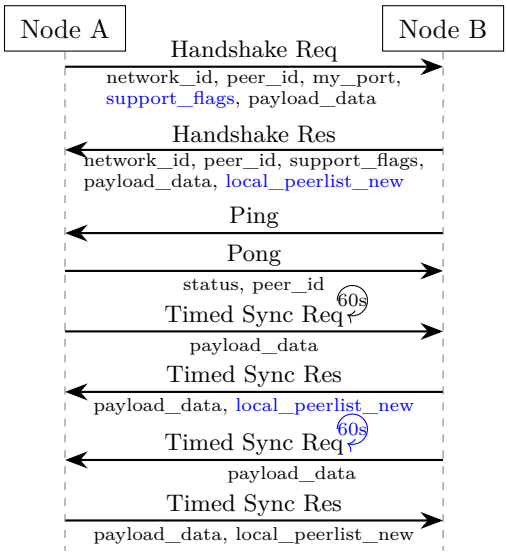
In the following, we will present an overview of relevant functionalities inside Monero's P2P network. Monero's P2P layer uses a custom application protocol known as the Levin protocol [17], most importantly consisting of a Handshake sequence and a continuous peer discovery mechanism using Timed Sync messages as visualized in Figure 1a and seen for a real sample peer in Figure 1b.

Handshake Sequence. To initialize a connection, node A sends a Handshake request to node B, sharing a unique network ID, node ID, and support flags. Since 2018, the support flag is 1 by default[7], indicating the usage of fluffy blocks, where nodes send compact block representations that contain mostly transaction IDs rather than full transaction data. If B has capacity, B responds with its own node and payload data and additionally a PL containing a maximum of 250 peers as part of Monero's peer discovery mechanism.

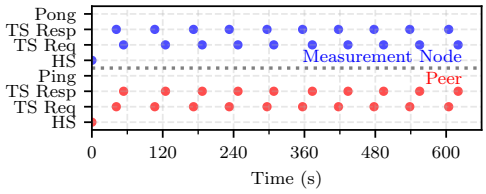
Peer Discovery Mechanism. Monero's peer discovery combines local PL management and PL exchanges between neighbors. An initial PL exchange occurs as part of the Handshake, after which both peers enter a 60-second loop that schedules Timed Sync messages to request and send PLs. Each Timed Sync response contains up to 250 white list entries, avoiding duplicates during a continuous exchange. Peer entries include IP address, port, network ID, and optionally pruning seed, RPC port, and RPC cost in credits per hash. Local PL management employs a white list for up to 1,000 validated peers and a gray list for up to 5,000 unvalidated peers. Received peers are initially added to the gray list. Every 60 seconds, the node randomly selects a gray list peer, tests reachability via Ping, and adds it to the white list upon validation. Similarly, incoming connections advertising an open port are tested via Ping and added to the white list if validated.

Offense Tracker. Monero applies a penalty mechanism to score peer behavior and drop connections when necessary [17]. However, the mechanism is limited and evaluates specific behaviors in the P2P communication like blockchain height discrepancies, transaction validity, or block verification failures. Each peer maintains a score that decreases when problematic behaviors are detected, such as sending invalid blocks, providing inconsistent blockchain data, or attempting double-spend transactions. When a neighboring peer's score falls below a critical threshold, the connection to this peer is terminated and it may be temporarily banned from reconnecting.

[7] `https://github.com/monero-project/monero/pull/3191`



(a) Monero's Default Message Sequence. The figure shows a standard Monero message sequence where node A opens an outgoing connection to node B. The Handshake request initiates the connection. The Ping/Pong sequence verifies that A allows incoming connections and adds A to B's white list. The Timed Sync commands are individual loops to retrieve updated information about the network's topology.



(b) Standard Connection. The graph displays a real sample of a standard incoming connection with each peer sending Timed Sync requests approximately every 60 seconds.

Figure 1 The figures depict how a typical connection in Monero's P2P network should look like, with the default message sequence outlined on the left and a sampled peer connection on the right.

Transaction Propagation. Monero's transaction propagation is based on Dandelion++ [8]. Dandelion++ employs two phases to enhance network-level anonymity. In the stem phase, transactions are forwarded to one of two outgoing relay nodes that are periodically switched. Each intermediate node either continues forwarding as stem or transitions to the fluff phase where transactions are broadcast using standard diffusion. While the protocol provides improved privacy over naive flooding protocols, Sharma et al. [19] demonstrate significant weaknesses against sophisticated adversaries. Using a combination of coordinated graph learning and intersection attacks, even modest adversaries occupying 20% of the network can significantly reduce the set of potential originators [19], revealing a substantial gap between formal guarantees and practical security.

4 Examination Framework

This section introduces our methodology for identifying and categorizing anomalous patterns. The framework provides a comprehensive analytical approach that extends beyond simple protocol compliance checking to encompass content, behavioral, structural, and attribute-based detection methods that can be easily applied to a variety of P2P networks. We establish formal definitions for the core components of our anomaly examination. Let $G = (V, E)$ represent a P2P network where V is the set of peers and $E \subseteq V \times V$ is the set of (logically) bidirectional connections. Each connection $e \in E$ carries a time-ordered sequence of messages $\mathcal{M}_e = \langle m_1, m_2, \ldots, m_k \rangle$, where each message has an associated type from the set $\mathcal{T}$ of all protocol message types.

4.1 Protocol-Level Anomaly Detection

Protocol-level anomalies are observable behavioral deviations that indicate non-standard implementations or configurations adapted from traditional signature-based and specification-based detection methodologies for distributed network environments. We define three distinct categories based on detection complexity and required domain knowledge.

4.1.1 Syntactic Violations

To identify syntactic violations, we systematically extract and validate all available protocol fields against current specifications, diverging from traditional signature-based approaches that target specific known attack patterns. Rather than searching for predetermined malicious signatures, our heuristic approach performs exhaustive field-level validation to identify any deviation from syntactic correctness. This includes:

- **Missing Required Fields**: Absence of mandatory protocol elements as defined in current specifications
- **Unexpected Field Presence**: Inclusion of deprecated or non-standard protocol elements
- **Malformed Message Structures**: Violations of defined message syntax and encoding standards

Through exhaustive field-level compliance checking against canonical protocol definitions, this approach identifies nodes running non-standard protocol implementations. While efficient, this methodology cannot detect more sophisticated anomalies or attacks that operate within syntactic correctness boundaries.

Syntactic Formalization. For each type $t \in \mathcal{T}$, let $\mathcal{F}_{req}(t)$ and $\mathcal{F}_{opt}(t)$ represent message type-specific required and optional fields. For a message m of type t, let $\mathrm{dom}(\phi_m)$ denote the set of fields present in m. The message is syntactically anomalous, if:

$$\mathrm{SynAnom}(m) = \mathcal{F}_{req}(t) \not\subseteq \mathrm{dom}(\phi_m) \vee \mathrm{dom}(\phi_m) \not\subseteq (\mathcal{F}_{req}(t) \cup \mathcal{F}_{opt}(t)) \tag{1}$$

4.1.2 Content Anomalies

Content Anomalies extend beyond structural validation to analyze logical content consistency. These subtle anomalies involve correctly formatted messages containing logically inconsistent or fabricated information that requires domain-specific knowledge for detection. Detection methodologies include:

- **Baseline Analysis**: Comparison of observed pattern with established expected content patterns
- **Cross-Peer Comparison**: Comparing information consistency across multiple sources

This approach effectively identifies content-based attacks that appear as legitimate protocol exchanges, particularly useful for detecting coordinated misinformation campaigns across multiple peers.

Content Formalization. For content anomalies we model validation through baseline comparison and cross-peer comparison. For a message m of type t with content c_m, the message is anomalous if:

$$\mathrm{ContAnom}(m) = \delta_t(\mathrm{c}_m, \mathcal{B}_t) \geq \tau_t \vee \rho_t(\{m\} \cup \mathcal{M}^{(m)}_{cross,t}) \notin [\tau^t_{c,min}, \tau^t_{c,max}] \tag{2}$$

where δ_t measures content deviation from the baseline against threshold τ_t for message type t. ρ_t computes cross-peer comparison scores for message set $\mathcal{M}^{(m)}_{cross,t}$, containing all messages from distinct peers of the same type. $[\tau^t_{c,min}, \tau^t_{c,max}] \subseteq [0, 1]$ defines the expected scoring range for type t.

4.1.3 Behavioral Pattern Deviations

Behavioral pattern deviations extend beyond static protocol validation to examine dynamic communication patterns, capturing anomalies that manifest through temporal and sequential deviations rather than explicit specification violations. It follows stateful protocol analysis principles [12], analyzing communication timing, frequency, and sequence patterns. These anomalies manifest through deviations from expected interaction patterns rather than explicit protocol violations. Key behavioral indicators include:

- **Timing and Frequency Anomalies**: Deviations from expected message timing and frequencies
- **Sequence Violations**: Non-standard state transitions
- **Connection Pattern Anomalies**: Unusual connection establishment and termination behaviors

Our methodology models expected protocol behavior, establishing definitive and statistical baseline patterns that enable detection of subtle deviations indicating reconnaissance activities, resource exhaustion attacks, or protocol manipulation attempts. This approach captures sophisticated attacks that operate within protocol specifications but exhibit anomalous temporal or sequential characteristics.

Behavioral Formalization. For behavioral anomalies we analyze communication patterns across connections and peers. For a connection $e \in E$, behavioral anomalies are detected through:

$$\begin{aligned} &\text{BehavAnom}(e, v) = \\ &\text{sequence}(\mathcal{M}_e) \notin \mathcal{S}_{standard} \vee \delta(E^v, \mathcal{B}_{conn}) > \tau_{conn} \vee |\text{timings}(\mathcal{M}_e^t) - \text{timing}_{standard}^t| > \epsilon_t \end{aligned} \tag{3}$$

where $\text{sequence}(\mathcal{M}_e)$ extracts the sequence of message types from connection e, $\mathcal{S}_{standard}$ represents standard sequences, $\text{timings}(\mathcal{M}_e^t)$ extracts the timing frequency of messages of type t in connection e, $\text{timing}_{standard}^t$ represents standard timing for message type t, ϵ is the deviation tolerance threshold for message type t, $\mathcal{B}_{conn}$ is the baseline for connection patterns, E^v denotes all connections by peer v, and δ measures deviation with threshold τ_{conn}.

4.2 Network Structure

We extend our examination by incorporating classic graph-theoretic concepts, including:

- **Centrality Anomalies**: Detecting nodes with unusual connectivity patterns
- **Community Detection**: Identifying unusual clustering patterns indicating coordinated malicious activity

P2P networks like cryptocurrency networks rely on decentralization and aim to fully decentralize the network. A certain degree of centralization can naturally occur due to geographically more active locations. Nonetheless, prior research by Zabla et al. and Beikverdi et al. reveal concerning trends toward increased centralization in major blockchain networks [3, 25]. In systems with topology obfuscation such as Monero, the true connection graph is not observable. We therefore apply these concepts to an observable proxy, the promotion topology defined in Sec. 5.2.3. We instantiate centrality as promotional influence but leave out community detection, as promotion edges would yield communities of advertisement rather than of connectivity.

4.3 Node Attributes

At last, we introduce a form of attribute fingerprinting based on various available attributes such as IP addresses, unique node identifier, RPC services, or pruning seeds for applicable cryptocurrencies. We include in our analysis:

- **Identity Correlation**: Correlating identities across the network
- **Geographic and ASN Analysis**: Analysis of infrastructure distribution patterns

Unlike protocol-level anomalies, network structure and node attribute assist our analysis in finding potential clusters by adding a network-wide perspective. We thus omit formal mathematical definitions. We now instantiate this framework on data collected from Monero's live network.

5 Case Study: Monero's P2P Network

We implement a comprehensive detection methodology to identify and quantify anomalous behavior within Monero's P2P network, applying the framework established in Sec. 4. The methodology leverages network traffic captured from unmodified P2P nodes participating in the presumably homogeneous network as standard pruned nodes with both incoming and outgoing connections. Our investigation spanned several months with repeated informal snapshots. In these preliminary observations, the same classes of anomalies recurred across capture cycles, which motivated a final measurement design of two consecutive 24-hour snapshots capturing this behavior at scale while keeping storage and computation feasible.

5.1 Data Collection

Our dataset constitutes a snapshot: five unmodified vantage points, two configurations each, captured over two days in June 2025. It is therefore a forensic record of the network state at that time, not a longitudinal characterization. All percentages reported in this section refer to peers observed during this window. Since anomalous infrastructure can be redeployed or withdrawn, current prevalence may differ. However, the released pipeline enables operators to repeat the measurement on newly captured traffic.

- **Locations**: San Francisco, Amsterdam, Bangalore, Singapore, and Sydney
- **Default Configuration**: 12 outgoing connections and no ban list enabled
- **Community Configuration**: 32 outgoing connections and a pre-loaded ban list[8]
- **Timeframe**: Network traffic captured June 2–4, 2025, with each configuration running for 24 hours

Following data collection, network packet analysis was conducted using Wireshark [24] to extract protocol information from identified Monero network traffic. We collected all available data fields, including command identifiers, node attributes, and payload data required for PL analysis. Each PL was systematically processed to extract complete peer entries with all associated metadata fields. This approach yielded a timestamped dataset enabling granular analysis of individual network connections, peer behaviors, and network representations.

5.2 Examination Results

In aggregate, the measurement nodes captured a total of 240 hours of network traffic. The collected dataset statistics are reported in Table 1 across all measurement nodes.

[8] `https://github.com/monero-project/meta/issues/1124`

Table 1 Individual and Aggregated Node Results. The table shows the amount of total Monero packets sent and received. The IPs are represented as IPs we have had a direct connection with and additionally total IPs we have seen inside the collected PLs.

	SF	AMS	BLR	SG	SYD	Total
Monero Packets	3,573,738	4,807,042	1,902,680	3,004,744	3,606,164	16,894,368
PLs received	135,242	171,956	113,753	118,692	133,435	673,078
IPs connected (ASs)	6,573 (1,246)	6,697 (1,229)	6,226 (1,170)	6,396 (1,173)	6,507 (1,194)	13,050 (1,870)
IPs seen (ASs)	14,401 (1,668)	17,784 (1,833)	15,191 (1,694)	14,237 (1,630)	15,935 (1,695)	26,919 (2,480)

The extracted PLs contained multiple IPs that are either stale or didn't have a connection established with any of our measurement nodes. These IPs yield information on subnet and AS affiliation but cannot be classified further. The examination results are aggregated over the five nodes. Unless noted otherwise, all examination results derive from the default-configuration nodes, some community-configuration results are reported in Sec. 6.2. All data[9] and the complete examination pipeline[10] is publicly available to reproduce the results in this paper.

We present the results in order of increasing evidential weight rather than framework order, to build a chain from structural and content indicators toward behavioral and syntactic fingerprinting.

5.2.1 IP Address Affiliation

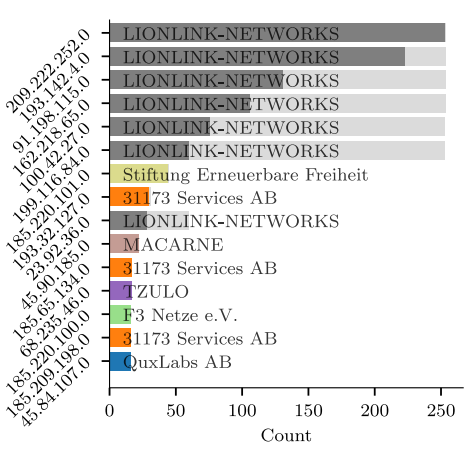
We begin by characterizing the global distribution of IP addresses, as part of the network structure (Sec. 4.2), across both direct connections and PLs, establishing a structural baseline against which anomalies are measured. As noted, the PLs include IP addresses with varying temporal validity, as PLs may contain stale entries from disconnected nodes or outdated IPs. This limits what PL-derived counts can support. We therefore use PL data only for structural claims about subnet and AS concentration, which are robust to individual entries being outdated, and never as a count of concurrently online peers. Our analysis reveals two concerning findings.

Subnet Saturation. We observed six /24 subnets fully saturated with 254 promoted IP addresses each. Additionally, Figure 2a shows that multiple subnets exhibit a population of ten or more peers.

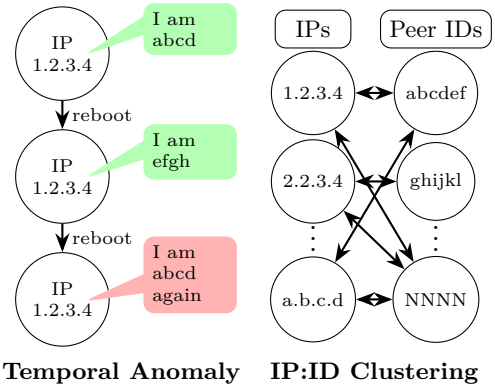
AS Concentration. All six saturated subnets are assigned to a single AS, which contains an additional densely populated subnet, totaling 1,582 peers within AS LIONLINK-NETWORKS (ASLN), occupying a large fraction of the peers observed in our measurement.

A high number of peers inside a few ASs can be expected and follows an ongoing centralization trend in decentralized systems [5, 16, 18]. However, a high number of peers inside the same subnet is a common indicator that these peers share a single operator. Figure 2a shows the most represented subnets along with their respective AS organization. Whereas the median saturation of any subnet is at 1, the top ones are fully saturated, and the most saturated ones share the ASLN.

[9] https://zenodo.org/records/16947083
[10] https://anonymous.4open.science/r/monero-traffic-analysis-2528



(a) AS Subnet Distribution Of Established Connections. Each bar represents a /24 subnet with colored ASs. The light gray bars show IPs retrieved from PLs without an established connection, resulting in a few fully saturated /24 subnets and multiple saturated subnets are affiliated under the same AS organization.



(b) Peer ID Anomalies. Monero's unique peer identifier is randomly generated during the start of a node, keeps static during the runtime, and newly generated again between every reboot. As the ID is not stored or loaded, after a node rebooted and identifies with a new ID, it should not have access to the older ones. Simultaneously, IPs cluster themselves along shared IDs, building not fully interconnected clusters of IP addresses aggregating behind multiple distinct identifiers.

Figure 2 The figures show anomaly patterns in Monero's IP distribution (left side) and when matched to the peer identifier (right side).

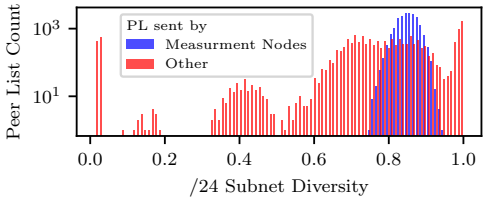
5.2.2 Monero's Peer Identifier

We examined the nodes' peer identifier as one of Monero's node attributes (Sec. 4.3). Peer identifiers are unique IDs randomly created during the start of the node. If a node is rebooted a fresh ID is created and if it reboots once or multiple times while the IP persists, an IP address naturally announces two or more identifiers. Similarly, if the peer's network IP changes due to ISP mapping or temporarily connecting to a VPN, it will keep its identifier while being associated with multiple IPs. However, by adding a temporal constraint, we compared the identifiers to the complete set of peers to identify patterns of back and forth changing peer IDs and clusters of IP addresses aggregating behind multiple distinct identifiers. This creates an anomaly in the network's node attributes as visualized in Figure 2b. We employed two analytical approaches to identify the respective patterns.

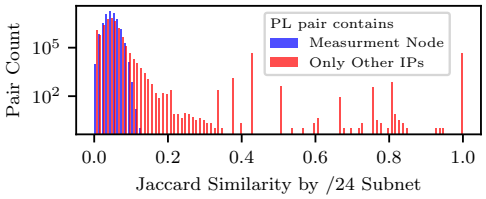
Temporal ID Anomaly. First, we sorted all collected packets that contain an identifier and checked the temporal constraint for ID announcements. An anomaly of back and forth changing IDs was observed for: 635 unique IPs, across 150 unique ASs.

ID:IP Clustering. As a second approach, we extracted IP addresses that announce multiple distinct IDs and reviewed if such IDs are also used by other IPs to identify clusters where distinct IDs are used by distinct IPs and vice versa as visualized at the right side of Figure 2b. We observed: 265 unique IPs clustered in 5 distinct clusters, across 33 unique ASs.

We observed a variety of ASs showing this anomaly and we acknowledge that peers might potentially aggregate behind a shared VPN endpoint. However, comparing it to the rest of the network, it does create an anomaly as 88% of the connected IP addresses announce only a single identifier. Moreover, the Monero Research Lab uses an active probing technique to



(a) PL Diversity. Distribution of PL diversity scores of individual PLs. Blue bars show PLs sent by our default measurement nodes, red bars represent PLs sent by all other IPs. Diversity is the ratio of unique /24 subnets to total peers. A group of lists accumulates at the left side of the figure, representing less than seven subnets among 250 peers.



(b) PL Similarity. Distribution of pairwise similarities between all distinct PL pairs excluding same peer comparisons. IPs in the PLs are reduced to its /24 subnet. Blue bars represent comparisons where at least one PL is sent by a measurement node. The distribution shows noticeable exceptions with high similarity scores and hundreds of pairs reaching a perfect overlap are on the right side of the figure.

Figure 3 PL Inconsistencies. The figures display PL inconsistencies. The extreme outliers on the left side for the PL diversities and on the right side for PL similarities represent nodes that are part of the ASLN.

identify peers switching IDs[11], and adding affected IPs to their published ban list. Finally, one of the five identified clusters contains 109 unique IPs exclusively within the densely populated ASLN.

5.2.3 PL Anomalies

The peer discovery mechanism is central to Monero's network health, independent of any blockchain or consensus logic, it governs how nodes locate and connect to peers. Here, we applied ContAnom (Sec. 4.1.2) to examine PL content with two approaches. First, we analyzed raw PL data at the individual IP address level. Second, we aggregated IP addresses by /24 CIDR subnet to identify potential clustering patterns. We selected /24 subnets as our aggregation unit because nodes within the same /24 subnet are likely to be controlled by a single entity, making subnet-level analysis a meaningful clustering indicator. Based on the different aggregation methods, we evaluated PLs individually by measuring their diversity and also performed cross-peer comparisons between distinct peers to identify cluster of peers sharing similarities. For each approach, we included only lists containing at least 225 entries (allowing a 10% variation) to establish a comparable baseline of similarly sized lists.

PL Diversity. We define diversity as the ratio of unique /24 subnets to total peers in a list. Under standard operation, random selection from the local white list is likely to produce PLs with high subnet diversity. A perfectly diverse list achieves a diversity index of 1.0, while lower values indicate subnet clustering. Modest clustering may occur naturally due to random selection from popular cloud providers, but extreme clustering represents a content anomaly that deviates from expectations of the protocol's random selection specification.

Figure 3a shows the distribution of diversity scores for PLs sent by our measurement nodes (blue) and all other IPs (red). All of our measurement nodes consistently achieve diversity scores above 0.7, which aligns with 90% of all PLs. However, two notable outliers (left side) exhibit diversity scores below 0.028. These lists advertise up to 250 peers distributed across just six or fewer subnets. Empirically, we observed 554 unique IPs show this anomalous PL diversity, all residing within the ASLN.

[11] `https://github.com/Rucknium/xmrnetscan/tree/main`

PL Similarity. We continued our analysis by measuring a pairwise similarity between PLs with IPs aggregated to its subnets. We computed standard Jaccard similarity coefficients, which capture the proportion of shared elements between two sets, to quantify the overlap between peer lists from independent IP sources. For a decentralized P2P network, PLs from independent peers should exhibit low similarity due to the randomized peer selection from distinct local white lists. However, our analysis revealed node pairs with suspiciously high similarity coefficients, including instances of over 90% and perfect overlap in subnet reduced representations. Figure 3b shows the distribution of pairwise similarities. With several peaks, especially, the peak showing a complete overlap in the subnet representation appears critical. Contrary, for similarity pairs where at least one of the PLs belongs to one of our measurement nodes, the similarity score never exceeds a score above 0.2.

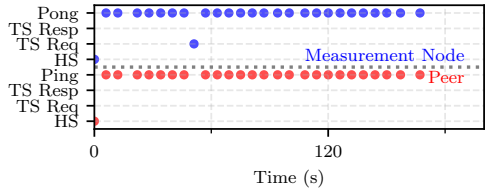
Empirically, 553 unique IPs achieve perfect subnet-level overlap in their peer lists, all residing within the ASLN. These peers exclusively advertise subnets in which they themselves reside, forming a promotional network spanning seven subnets. A peer that had itself been poisoned would predominantly advertise a foreign subnet. Self-referential advertisement, where a peer's list is dominated by the subnet it resides in, is therefore more consistent with being the source of the promotion than its recipient.

Monero's Promotion Topology. The PL anomalies identified above characterize individual peer behavior. We now examine how these behaviors manifest at the network level. Since Monero implements topology obfuscation to protect routing privacy, we cannot reconstruct the true connection graph. Rather than developing novel topology learning attacks, we examine Monero's "promotion topology", revealing structural anomalies in Monero's peer discovery mechanism. We define the promotion topology as a directed graph with IP addresses as nodes. For each received PL, we construct directed edges from the source of the PL to every peer it has promoted within this PL. While this graph does not reflect the network's actual topology at any given time, it captures promotional behaviors. Therefore, multiple edges from one node to another are allowed as multiple promotions naturally occur for extended timeframes. This view has limits: promotion edges reflect advertisement behavior, not established connections, and are biased toward peers that our vantage points happened to observe. It therefore supports claims about promotional prominence, not about actual link-level topology or global node degree.

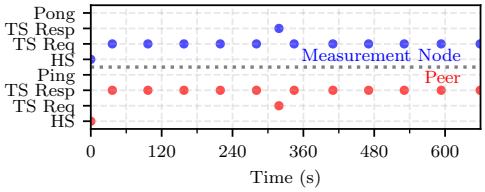
We observed, that the in-degree distribution of the promotion topology is highly skewed, with a small number of peers receiving a disproportionate share of promotions, consistent with scale-free degree distributions observed in Internet topology [7]. At the IP level, individual ASLN addresses do not rank among the highest in-degree nodes. However, when promotions are aggregated by AS and within our captured PLs, ASLN dominates, receiving nearly three times as many promotions as the next largest AS (Hetzner Online GmbH) and more than hundred times as many as the top individual IP.

5.2.4 Behavioral Anomalies

Structural and content anomalies established the presence of non-standard peers. We now apply *BehavAnom* (Sec. 4.1.3) to examine behavioral deviations that further characterize these nodes. We measured timing, frequency, and sequential deviations across the P2P protocol. Monero's message set comprises base commands that follow deterministic patterns and event commands that exhibit unpredictable timing characteristics. A standard connection sequence consists of two mandatory message exchanges, bidirectional Handshake and periodic Timed Sync commands, each in request-response format. Additionally, when a peer advertises



(a) Ping Flooding. A peer spamming Ping messages to our measurement node. After a successful Handshake, the peer spams Ping messages. The measurement node does not crash, treats the connection as usual, and starts at around 60 seconds with the first Timed Sync request. The connection is dropped as the peer does not respond to the request within 120 seconds.



(b) Throttled Timed Sync. Sample peer throttling its Timed Sync request messages. Contrasting to the measurement node using the fixed 60-second frequency for sending requests, the peer follows a non-standard frequency. With a single request within ten minutes while responding timely, the peer extremely deviates from the protocol standard.

Figure 4 Behavioral Anomalies. Both figures show connection-level deviations from Monero's protocol standard, as observed on our measurement nodes.

an accessible port for incoming connections, Ping messages are transmitted by the connecting peer to verify the advertised inbound connectivity capability. The Timed Sync exchanges occur at a defined frequency of 60 seconds, initiated by both peers in parallel. Figure 1 demonstrates the default and baseline behavior. The patterns represent the expected behavior between two standard Monero clients. To identify anomalous behavior, we focus on the three predictable command patterns. We adopt a connection grouping methodology, filtering our dataset by individual connections before conducting detailed analysis. Connections exhibiting TCP-level anomalies, likely attributable to Wireshark dissector limitations, were excluded from analysis due to their negligible frequency and potential for introducing analytical artifacts. We located:

- 181,091 total TCP connections.
- Connections last on average 364.61 (median: 0.54) seconds, and
- connections contain on average 93.14 (median: 4) commands.

The discrepancy between average and median is the result of many short-lived connections usually lasting less than one second.

Short-lived Connections. While brief connections may result from legitimate network conditions or operational factors, it becomes critical to determine whether specific peer populations exhibit disproportionate representation in short-lived connection occurrences. We therefore analyzed the distribution of IPs across short-lived connection anomalies, as repeating IPs indicate systematic scanning or probing activities. Empirically, we observed 85,288 connections lasting less than 1 second. Of these, 10 unique IPs have more than ten short-lived connections, across 7 unique ASs.

All of the observed IP connections completed the Handshake sequence and, in some cases, sent a single Ping or Timed Sync message before terminating. Notably, a single IP address accounts for 65,340 connections terminating in under one second. This pattern indicates systematic probing rather than transient network churn. We note that research and community crawlers produce a similar signature and this evidence alone does not distinguish scanning conducted for measurement from scanning conducted for reconnaissance.

Ping Flooding. We observed excessive Ping message transmissions beyond what normal peer verification requires. Figure 4a visualizes a sample connection exhibiting this bursty behavior. Within 24 hours, our measurement nodes received 2,900 Ping flooding incidents,

each delivering a burst of around 23 pings, from 501 distinct IPs, all residing in the ASLN. Notably, incoming connections should only respond to locally initiated Ping messages to prove their capability to as well accept incoming connections, showing a clear deviation from defined protocol sequences. These connections serve no other apparent protocol function, as evidenced by their uniform disregard for Timed Sync requests. They consistently ignore the first Timed Sync request, causing the connection to drop after the 120-second inactivity timeout, indicating that these nodes neither seek nor sustain normal protocol participation.

Throttled Timed Sync. Given the fixed protocol specification requiring Timed Sync requests every 60 seconds, we measured the frequency distribution of these messages across all connections. Figure 4b demonstrates a peer transmitting Timed Sync requests far off the specification. We initially developed a baseline by analyzing the network's overall distribution of Timed Sync frequencies as some timing variations may be attributed to network latency or node computational load. We average the Timed Sync frequency on each connection and the median is at around 61.02 seconds, matching with the protocol specifications. However, we identified connections that consistently deviate from the defined 60-second interval. We set our threshold at 120 seconds, what corresponds to monerod's timeout period after which a node connection drops due to peer unresponsiveness. Any peer with a Timed Sync frequency above 120 seconds must be responsive in general communication. In all cases where the Timed Sync frequency exceeds 120 seconds, peers respond to our requests without notable delay. While we cannot attribute throttled Timed Sync to a single specific attack vector, the behavior requires deliberate source code modification. Empirically, we observed 121 connections have a frequency above 120 and 27 IPs maintain connections with throttled Timed Sync frequencies, across 21 unique ASs.

5.2.5 Syntactic Anomalies

Beyond behavioral deviations, we applied *SynAnom* (Sec. 4.1.1) and identified three syntactic violations of Monero's protocol standard, indicating outdated or incorrectly implemented protocol standards. While not immediately critical in isolation, they are consistent with intentional customization and correlate with other anomalies identified above.

Support Flags Omission. We identified nodes omitting the support flags field during Handshake exchanges. Support flags are enabled by default. These peers must have explicitly set this flag to 0 or falsely implemented a custom client. Beyond syntactic identification, this violation creates a unique command sequence pattern where support flags are explicitly requested through Support Flags request messages, followed by response messages, deviating from standard command sequence flows. Empirically, we observed 1,054 unique IPs omit the support flags, across 80 unique ASs.

These missing flags resulted in 177,853 additional support flags request sequences, deviating standard protocol flow and creating increased network workload.

Deprecated `last_seen` timestamps. During PL exchanges in Handshake responses or Timed Sync responses, we observed peers still including the deprecated `last_seen` timestamp field. As of our observation, the timestamp that these peers transmit appears to be placeholders fixed to a value, indicating an outdated custom implementation of a Monero node. Empirically, we observed that 288 unique IPs inside the ASLN continue to transmit the `last_seen` timestamp.

While these two anomalies represent syntactic violations, they have historically correct syntax and are still accepted in Monero's current connection handling. Nonetheless, these outdated definitions date back longer than the last network upgrade, which should have enforced every node to update their software, raising questions on the legitimacy of node software running on servers showing these violations.

TCP Fragmentation. Messages in P2P traffic can get bigger than the Maximum Transmission Unit (MTU) size. Those messages naturally get fragmented and reassembled and the size of the fragmentation can differ depending on the network path's MTU size. However, we identified IP addresses that consistently fragment packets in an unusual manner. For each reassembled message, the first packet only contains Monero's 8 byte signature. This persists even for messages that could be transmitted in a single packet. We observed 1,123 unique IPs sending signature only packets, across 5 unique ASs.

Such a behavior is unlikely to occur naturally, as legitimate MTU-based fragmentation would create more varied fragment sizes based on network conditions. The concentration within a small number of ASNs, including the ASLN, is consistent with coordinated deployment.

Individually, several of these anomalies admit benign explanations. Their joint co-occurrence within a shared AS, however, is not explained by hosting, VPN aggregation, or churn, and is most parsimoniously attributed to a single coordinated actor. We make this interpretation explicit in Sec. 6.

6 From Anomalies to Adversary

In this section, we first characterize the coordinated attack chain (Sec. 6.1), evaluate existing defenses (Sec. 6.2), and then propose mitigations (Sec. 6.3). Throughout this section, we use the term "adversary" to refer to the entity or entities responsible for the anomalies characterized in Sec. 5. We acknowledge in Sec. 7 that anomalous behavior does not constitute proof of malicious intent. However, the scale, coordination, and network positioning of the observed anomalies support an adversarial interpretation.

6.1 A Coordinated Attack Chain

The adversary's operational footprint likely extends beyond ASLN. As shown in Figure 6a, the anomaly overlap reveals that many identified anomalies extend beyond ASLN into additional ASs. This suggests the adversary operates nodes with distinct IP addresses outside their primary infrastructure, either as auxiliary capacity or to reduce detectability. We focus our threat analysis on the ASLN cluster (1,582 nodes), where anomalies are most concentrated and corroborated, but the true operational footprint is likely broader. We break down the threat chain shown in Figure 5 by evaluating the identified anomalies and explaining the effect each enables.

(1) Sybil attack. At the center of our analysis lies a set of seven highly saturated subnets. The concentration of 254 peers within a single /24 subnet is strongly indicative of a Sybil attack, a conclusion further supported by several corroborating anomalies. First, these peers belong to large peer ID clusters characterized by shared and frequently rotated identifiers. Second, they exhibit a distinctive communication pattern consisting of signature-only packets. This uniform behavior suggests these IPs function as proxy or VPN endpoints load-balancing traffic for a limited number of actual Monero nodes.

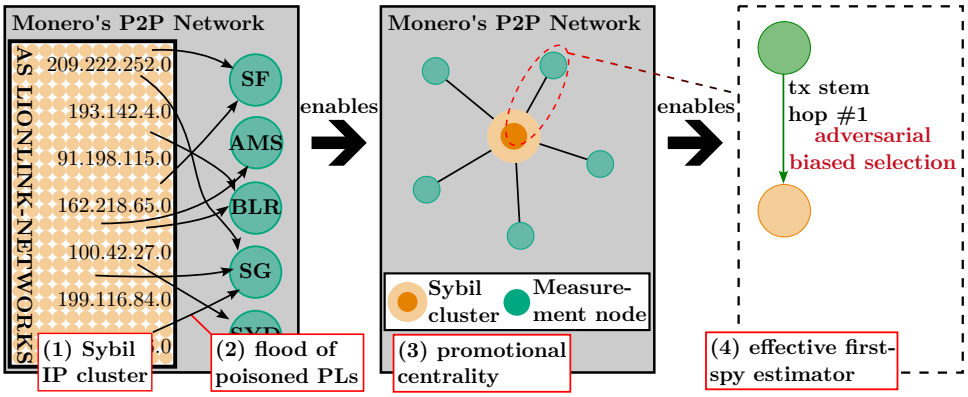
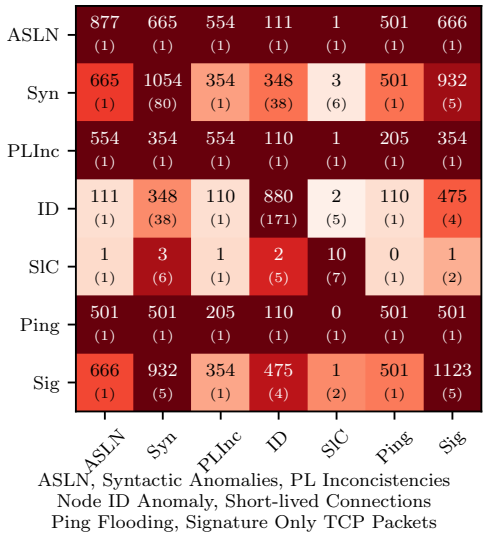


Figure 5 ASLN Threat Chain. AS LIONLINK operates seven densely populated subnets forming a Sybil cluster within Monero's P2P network. (1) The cluster's scale enables a Sybil attack, with the full saturation of /24 subnets suggesting that a small number of actual nodes operate behind proxy endpoints. (2) Peers across all seven subnets flood the network with poisoned PLs exclusively containing IP addresses from within these same subnets, as observed by each of our five monitoring nodes. (3) The combined effect of Sybil presence and PL poisoning translates into outsized promotional centrality, with adversarial peers consistently occupying a fraction of outgoing connections. (4) This central positioning enables an effective first-spy estimator: with high probability of being selected as an outbound relay during Dandelion++'s stem phase, the adversary can link transactions to their originating node.

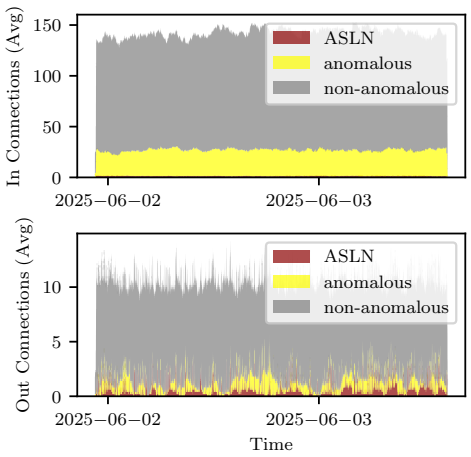
(2) Peer List Poisoning. The semantic PL inconsistencies imply poisoning of the discovery mechanism to increase visibility and gain strategic positioning. Although nodes will prefer to choose outgoing connections to distinct /16 subnets, high amounts of nodes in the local white and gray lists will naturally increase the chance of picking a node in the ASLN. This visible PL manipulation occurs exclusively for the ASLN but appears to be very effective. The Sybil attack combined with the PL poisoning results in widespread exposure across the network. We examined the inferred direct exposure of each node by analyzing all collected PLs along their promoted peers. Our measurements reveal that the majority of PLs we received contain ASLN peers, and a peer's PL is on average occupied by 16.22% ASLN peers.

(3) Central Network Positioning. The promotion topology analysis in Sec. 5.2.3 establishes that ASLN achieves outsized promotional prominence in our captured PLs. To quantify how this translates into direct exposure on a node-level, we measured how many ASLN and anomalous peers occupy our measurements' nodes active connection pools over the observation period, averaged across all five nodes. As shown in Figure 6b, ASLN peers occupy 0.94% of incoming and 6.69% of outgoing connections on average. Additionally, 18.05% and 7.78% of incoming and outgoing connections are occupied by anomalous peers. This degree of saturation is not coincidental but a direct operational consequence of the Sybil attack and PL poisoning described in steps (1) and (2). By flooding the network with nodes and ensuring those nodes dominate peer lists, the adversary translates promotional influence into persistent connection-level presence. This central positioning is the precondition for the privacy attacks described next.

(4) Transaction Privacy Degradation. The presence and network positioning of ASLN peers creates the preconditions necessary to compromise Dandelion++. The main capability to undermine privacy guarantees is based on the central position its nodes achieve through



(a) Anomaly Overlap Matrix. The matrix presents the overlap of IPs (ASs) showing anomalous behavior in the respective anomaly. Whereas many IPs are often affiliated with just a few ASs, some anomalies show higher AS overlap than IP overlap, suggesting coordinated deployment of peers with distinct functionalities.



(b) Proportion of Active Connection Saturation. The graph visualizes the proportion of active connections. Outgoing and incoming connections show on average around 7.78% and 18.05% are occupied by non-standard peers and 6.69% and 0.94% are occupied by adversarial peers.

Figure 6 Adversarial Reach. The overlap matrix (left) characterizes which anomalies share IP and AS affiliation. The connection saturation graph (right) quantifies how this translates into direct presence within an individual node's active connection pool.

effective PL poisoning and Sybil attacks. Dandelion++ applies an initial stem phase where the transaction is only forwarded to one of the node's chosen outbound relays. With the ASLN occupying on average 6.69% of each node's outgoing connection pool, the adversary consistently has a chance of being picked as an outbound relay, thereby being positioned to learn about new transactions and apply a first-spy estimator (a deanonymization technique linking the first node to relay a transaction as its likely originator). Assuming the ASLN successfully learns the network's topology, it can further limit possible originators beyond relying solely on the first-spy-estimator. According to Sharma et al. [19], an adversary that controls around 20% of the network can intercept around 70% of transactions and effectively reduce the anonymity set to 32 possible originators per transaction. However, we observed a node's connection pool to be occupied by 7.78% of anomalous peers. This scenario potentially constitutes a more critical threat model than conventional network distribution approaches. Widespread connection saturation may enable direct node identification or, as approached by Sharma et al., allow adversaries to systematically eliminate candidate nodes, constraining the anonymity set to a limited subset of potential originators.

Finally, in certain cases, the adversary might confidently identify a transaction's originator. If the adversary knows whether a node is reachable (i.e., whether it allows inbound connections) and receives a stem transaction from such an unreachable node, it can confidently identify this as the originator since the node cannot have relayed the transaction from elsewhere.

Eclipse Attacks. Another final attack mapping follows a combined approach of Handshake and Ping flooding as reported in two distinct vulnerability reports [12]. The observed patterns also align closely with the eclipse attack methodology described by Shi et al. [21]. Many distinct peers align around each measurement node, able to quickly fill a victim's gray and white lists once needed. The next step requires a connection reset attack. While the double-spend attack vector has been closed, we observed many attempts to perform Ping flooding attacks, which have a similar outcome but have also been disclosed and fixed. We did not observe a successful eclipse attack against our measurement nodes. Our observations are most consistent with a broad monitoring and Sybil deployment rather than a targeted eclipse campaign. A targeted eclipse attacker would concentrate resources on selected victims and would largely avoid simultaneous exposure at five unrelated vantage points. Our data therefore speaks to eclipse capability rather than to an attempt in progress.

6.2 Defense Mechanisms

As a potential countermeasure, Monero contributors advertise the usage of a community crafted ban list. We evaluate the effectiveness of this defense against the identified threats.

Ban List Effectiveness. To compare the ban list with our identified anomalous nodes, we merge all anomalous peers identified across measurement nodes, arriving at 1,877 anomalous IP addresses. Of these, 734 are not in the community proposed ban list. With an active ban list and an increased pool of outgoing connections, we observed no Ping flooding attacks and outgoing connections occupied by anomalous peers dropped to an average of 3.18%. The ASLN is fully blocked on the ban list.

Ban List Limitations. Although a ban list seems to increase privacy from an individual node's perspective, we observed ban-listed peers being promoted in the majority of PLs. This suggests that many nodes in the network have not enabled the recommended ban list. As such nodes will be neighbors of nodes having a ban list enabled, the adversary will most likely be able to map these neighboring nodes despite their active ban list. Extending this, even if a majority is reached that have enabled an effective ban list, updates must be maintained and monitored on an ongoing basis. The static nature of a ban list cannot adapt to a rapidly evolving adversary.

Furthermore, the results showed that even with an active ban list many distinct anomalies persisted. We continued to observe several distinct anomalous behaviors for each measurement node with an active ban list throughout the observation period. Extending ban list coverage to entire ASs would reduce exposure further, but an adversary with sufficient resources to deploy a diverse, multi-AS infrastructure can circumvent AS-level filtering by design. The fundamental limitation is architectural, as a static, externally-maintained list reacts to observed behavior rather than preventing it.

6.3 Mitigation Strategies

Protocol changes such as the removal of the `last_seen` timestamp or support flags date back to 2018, but the network does not yet punish false usage of either. The already existing offense tracker could be updated to account for syntactic incorrectness and act accordingly.

[12] `https://hackerone.com/reports/3185083, https://hackerone.com/reports/2858802`

The effectiveness of PL poisoning could be reduced through periodic pruning of a node's local white and gray lists, removing duplicated subnets or even ASs. In addition, a set of anchor nodes, potentially the seed nodes, could act as a regular peer, providing a healthy PL to help counteract the self-reinforcing nature of PL contamination.

However, these mitigations address symptoms rather than root causes, and each defense carries its own tradeoffs. Validating PL entries and pruning duplicated subnets raises the cost of PL poisoning, but favoring older or more stable peers risks introducing centralization and ossification, and constraining peer selection too far erodes the randomness of outgoing connections on which eclipse resistance depends. Reputation systems for peer behavior can themselves be accumulated and gamed by Sybil identities, shifting the attack surface rather than eliminating it. Any effective mechanism to detect and penalize coordinated Sybil networks must therefore be designed with these manipulation risks in mind. The sophistication of the observed adversary suggests that incremental improvements to existing defenses may prove insufficient without fundamental architectural changes to the peer discovery and connection management mechanisms, yet such changes must balance Sybil resistance against decentralization and openness. The community ban list remains a useful short-term measure but provides no durable security guarantee against the adversary characterized in this paper.

Complementing these protocol-level changes, our released pipeline lets operators act today: it takes a PCAP captured by their own node and outputs a per-anomaly breakdown, reporting each flagged IP alongside the specific anomaly classes it triggered. Operators can inspect *why* a peer was flagged rather than trusting an opaque list, and derive a local ban list from the output. As it is regenerated from current traffic, the breakdown adapts to redeployed infrastructure, addressing the static ban list's central limitation (Sec. 6.2). Flags remain indicators rather than proof of malice, so operators tune sensitivity against their review capacity.

7 Discussion

Our approach provides us with a verified behavioral baseline from five geographically distributed locations. Running the default and unmodified Monero software allows us to distinguish standard from anomalous behavior, and to infer non-standard implementations where the observed behavior cannot be produced by an unmodified Monero client. This consistent baseline behavior, combined with Monero's network architecture presuming homogeneous node software deployment, ultimately allows us to distinguish between standard and non-standard behavior.

Thresholds, Anomalies, and Malicious Intent. We identified many IPs showing anomalous signs, associated with either few or many distinct ASs. While syntactic anomalies are clear indicators, behavioral inconsistencies are more ambiguous. Node ID anomalies, throttled Timed Syncs, or PL similarities might indicate anomalous peers without malicious intent. Several benign explanations exist for individual anomalies in isolation. Shared VPN endpoints may cause multiple IPs to announce the same peer ID. Network congestion may account for throttled Timed Sync frequencies. Peering agreements between hosting providers could produce correlated peer lists over time. Short-lived connection bursts and elevated Ping rates are likewise characteristic of active measurement infrastructure. However, these explanations do not account for the joint co-occurrence of anomalies within a single AS, nor for the scale and uniformity of the observed behavior. While our node's data serves as a verified baseline for identifying anomalies, direct classification as malicious requires verified labels. However,

consistent deviation from network baseline remains significant for security contexts, regardless of ultimate intent. A sensitive threshold can produce many false positives requiring manual review but reducing missed threats. An insensitive threshold can prioritize precision at the cost of potentially overlooking subtle anomalies. In our examination, we focused on clear outliers in our dataset rather than defining specific thresholds, as optimal values depend on operational capacity and use case requirements.

Community Awareness. Monero's community has been increasingly aware of potential surveillance and malicious activities within the network with an ongoing discussion and monitoring[13] that partially overlap with our findings. Most prominently, repeated ID switching is used to fingerprint spy nodes, and affected IPs are added to a continuously updated ban list. Our anomaly detection approach complements these community efforts by providing a comprehensive framework with empirical evidence for identifying non-standard nodes. The Monero Research Lab is aware of our analysis and already is testing our examination pipeline on newly captured data, which supports the practical applicability of our approach despite the absence of formal ground truth data.

8 Conclusion & Future Work

In this work, we have proposed methods to identify anomalous nodes in Monero's P2P network. We have found that approximately 14.38% of the observed peers in Monero's P2P network exhibit anomalous behavior. At the time of measurement, this indicated that a well-resourced actor was active in the network, performing what appears to be PL poisoning, topology mapping, and connection saturation. Despite community mitigation efforts, including when adopting ban lists, anomalous peers still occupied 3.18% of outgoing connections, potentially undermining Monero's privacy guarantees and creating a critical gap between theoretical protections and practical vulnerabilities.

First, we envision further study of the promotion topology as we have identified interesting patterns that might yield further insights. Second, another direction concerns the impact on Dandelion++ considering adversarial structures which go beyond random distributions in the network.

References

1 Maria Apostolaki, Aviv Zohar, and Laurent Vanbever. Hijacking bitcoin: Routing attacks on cryptocurrencies. In *2017 IEEE symposium on security and privacy (SP)*, pages 375–392. IEEE, 2017. doi:10.1109/SP.2017.29.

2 Katsiaryna Bahamazava and Rohan Nanda. The shift of darknet illegal drug trade preferences in cryptocurrency: The question of traceability and deterrence. *Forensic Science International: Digital Investigation*, 40:301377, 2022. doi:10.1016/j.fsidi.2022.301377.

3 Alireza Beikverdi and JooSeok Song. Trend of centralization in bitcoin's distributed network. In *2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 1–6, 2015. doi:10.1109/SNPD.2015.7176229.

4 Alex Biryukov and Sergei Tikhomirov. Deanonymization and linkability of cryptocurrency transactions based on network analysis. In *2019 IEEE European symposium on security and privacy (EuroS&P)*, pages 172–184. IEEE, 2019. doi:10.1109/EuroSP.2019.00022.

[13] https://moneronet.info

5 Tong Cao, Jiangshan Yu, Jérémie Decouchant, Xiapu Luo, and Paulo Verissimo. Exploring the monero peer-to-peer network. In *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24*, pages 578–594. Springer, 2020. doi:10.1007/978-3-030-51280-4_31.

6 John R. Douceur. The sybil attack. In Peter Druschel, Frans Kaashoek, and Antony Rowstron, editors, *Peer-to-Peer Systems*, pages 251–260, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. doi:10.1007/3-540-45748-8_24.

7 Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. *ACM SIGCOMM computer communication review*, 29(4):251–262, 1999. doi:10.1145/316188.316229.

8 Giulia Fanti, Shaileshh Bojja Venkatakrishnan, Surya Bakshi, Bradley Denby, Shruti Bhargava, Andrew Miller, and Pramod Viswanath. Dandelion++: Lightweight cryptocurrency networking with formal anonymity guarantees. *Proc. ACM Meas. Anal. Comput. Syst.*, 2(2), June 2018. doi:10.1145/3224424.

9 Nada Hammad and Friedhelm Victor. Monero traceability heuristics: Wallet application bugs and the mordinal-p2pool perspective, 2024. doi:10.48550/arXiv.2408.05332.

10 Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on {Bitcoin's}{peer-to-peer} network. In *24th USENIX security symposium (USENIX security 15)*, pages 129–144, 2015. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/heilman.

11 Harry Kalodner, Malte Möser, Kevin Lee, Steven Goldfeder, Martin Plattner, Alishah Chator, and Arvind Narayanan. BlockSci: Design and applications of a blockchain analysis platform. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2721–2738. USENIX Association, August 2020. URL: https://www.usenix.org/conference/usenixsecurity20/presentation/kalodner.

12 Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24, 2013. doi:10.1016/j.jnca.2012.09.004.

13 Yuval Marcus, Ethan Heilman, and Sharon Goldberg. Low-resource eclipse attacks on ethereum's peer-to-peer network. Cryptology ePrint Archive, Paper 2018/236, 2018. URL: https://eprint.iacr.org/2018/236.

14 Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, and Nicolas Christin. An empirical analysis of traceability in the monero blockchain, 2018. doi:10.48550/arXiv.1704.04299.

15 Till Neudecker and Hannes Hartenstein. Network layer aspects of permissionless blockchains. *IEEE Communications Surveys & Tutorials*, 21(1):838–857, 2019. doi:10.1109/COMST.2018.2852480.

16 Michael Polinski, Richard Jo, Kevin McAfee, and Fabián E. Bustamante. The centralization of a decentralized video platform - a first characterization of peertube. *SIGCOMM Comput. Commun. Rev.*, 54(4):25–35, February 2025. doi:10.1145/3717512.3717516.

17 Monero Project. Monero. https://github.com/monero-project/monero, 2024. offense tracker: src/cryptonote_protocol/ cryptonote_protocol_handler.inl:2792, levin protocol: docs/LEVIN_PROTOCOL.md., peer_id: src/p2p/net_node.inl:143 Accessed: May 06, 2025.

18 Muhammad Saad, Victor Cook, Lan Nguyen, My T. Thai, and David Mohaisen. Exploring partitioning attacks on the bitcoin network. *IEEE/ACM Transactions on Networking*, 30(1): 202–214, 2022. doi:10.1109/TNET.2021.3105604.

19 Piyush Kumar Sharma, Devashish Gosain, and Claudia Diaz. On the anonymity of peer-to-peer network anonymity schemes used by cryptocurrencies. *arXiv preprint arXiv:2201.11860*, 2022. doi:10.48550/arXiv.2201.11860.

20 Ruisheng Shi, Yulian Ge, Lina Lan, Zhiyuan Peng, Shenwen Lin, and Lin Li. Deanonymizing transactions originating from monero tor hidden service nodes. In *Companion Proceedings of the ACM Web Conference 2024*, pages 678–681, 2024. doi:10.1145/3589335.3651487.

21 Ruisheng Shi, Zhiyuan Peng, Lina Lan, Yulian Ge, Peng Liu, Qin Wang, and Juan Wang. Eclipse attacks on monero's peer-to-peer network. In *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society, 2025. URL: https://www.ndss-symposium.org/ndss-paper/eclipse-attacks-on-moneros-peer-to-peer-network/.

22 Muoi Tran, Inho Choi, Gi Jun Moon, Anh V. Vu, and Min Suk Kang. A stealthier partitioning attack against bitcoin peer-to-peer network. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 894–909, 2020. doi:10.1109/SP40000.2020.00027.

23 Liang Wang and Jussi Kangasharju. Real-world sybil attacks in bittorrent mainline dht. In *2012 IEEE Global Communications Conference (GLOBECOM)*, pages 826–832. IEEE, 2012. doi:10.1109/GLOCOM.2012.6503215.

24 Wireshark Foundation. Wireshark 4.4.7. Network protocol analyzer. URL: https://www.wireshark.org/.

25 Philipp Zabka, Klaus-T. Förster, Christian Decker, and Stefan Schmid. A centrality analysis of the lightning network. *Telecommunications Policy*, 48(2):102696, 2024. doi:10.1016/j.telpol.2023.102696.

Who Plays to Earn? Detecting Bots in Web3 Games via On-Chain Transactions Analysis

Woncheol Jeong
Korea University, Seoul, Republic of Korea

Sohee Choi
Korea University, Seoul, Republic of Korea

Jiwon Yoon[1]
Korea University, Seoul, Republic of Korea

Abstract

Bots in Web3 Play-to-Earn (P2E) games extract on-chain rewards at scale, accelerating token inflation and eroding the incentives that sustain legitimate participation. Existing defenses transfer poorly to this setting. Traditional game-bot detectors require proprietary server-side telemetry that is unavailable on permissionless chains, and blockchain-native detectors target DeFi behaviors whose footprints differ fundamentally from those of game automation; research specific to Web3 game bots remains scarce, and no public labeled dataset exists. We argue that the very economic asymmetry that makes P2E automation profitable also exposes it, because the temporal, topological, financial, and behavioral regularities that bots cannot disguise without forfeiting their efficiency advantage remain observable from public on-chain transactions alone. Building on this insight, we propose a cross-game, cross-chain detection framework that models player activity as a heterogeneous temporal transaction graph over a unified five-type edge vocabulary, extracts 32 chain-agnostic behavioral features, and classifies accounts through a dual-branch encoder fusing a typed-edge graph attention network (GAT) with a gated recurrent unit (GRU)-attention sequence model, bootstrapped by graph-contrastive pre-training. To support evaluation, we further release the first multi-game labeled bot dataset, spanning three P2E titles on three heterogeneous chains with 57,879 accounts and 3,551 confirmed bots. On the joint test split, the framework attains F1 = 94.39%, AUC = 99.92%, and AP = 98.90%, with a sharply bimodal score distribution and stable performance across a wide range of decision thresholds.

2012 ACM Subject Classification Security and privacy → Intrusion/anomaly detection and malware mitigation; Computing methodologies → Neural networks; Information systems → Data mining

Keywords and phrases Bot detection, Web3 gaming, Play-to-Earn, on-chain behavioral analysis, transaction graph analysis, heterogeneous graph neural network

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.32

Supplementary Material *Dataset*: `https://github.com/WONCHEOL-JEONG/Dataset-for-Detecting-Bots-in-Web3-P2E-Games.git` [8]
archived at `swh:1:dir:2eef651a0c2d25b011e0754a2f62e4784d985072`

Funding This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. RS-2024-00460321, Development of Digital Asset Transaction Tracking Technology to Prevent Malicious Financial Conduct in the Digital Asset Market). This work was also supported by Korea Internet & Security Agency(KISA) grant funded by the Korea government(PIPC) (No. 2780000036, Masters and Doctoral Program for Advanced Talent Development in Personal Data Protection and Responsible Use).

[1] Corresponding author

© Woncheol Jeong, Sohee Choi, and Jiwon Yoon;
licensed under Creative Commons License CC-BY 4.0

1 Introduction

Web3 Play-to-Earn (P2E) games allow players to earn cryptocurrencies and NFTs by performing gameplay actions recorded on public blockchains [26]. This direct coupling of gameplay and financial return has made P2E ecosystems a lucrative target for automated exploitation: bots farm rewards around the clock, accelerating token inflation, depressing per-player yields, and eroding the trust of legitimate participants [1]. Existing detection approaches are ill-suited to this setting: traditional game-bot detectors rely on proprietary server-side signals such as self-similarity measures and behavior sequences [10, 23] that are unavailable in permissionless blockchain environments; blockchain-native anomaly detectors target DeFi-specific behaviors [2, 14] that differ fundamentally from those of game bots; and all prior work examines a single game on a single blockchain with no public labeled dataset [6].

We observe that every economically meaningful action in a P2E game settles as a public, immutable on-chain transaction, producing behavioral traces that bots cannot disguise without sacrificing the efficiency that makes automation profitable; we substantiate this claim in Section 3.1. This economic asymmetry suggests that on-chain data alone may carry sufficient signal for effective detection.

Building on this observation, we propose a cross-game bot detection framework that operates exclusively on public on-chain data, and evaluate it across three P2E titles on three heterogeneous blockchains.

Contributions

This paper makes the following contributions:

- **Problem formulation.** We formalize Web3 game-bot detection as node classification on a heterogeneous temporal transaction graph with a unified five-type edge vocabulary abstracting across games and chains.
- **Feature design.** We propose 32 chain-agnostic behavioral features in four complementary groups (temporal regularity, graph topology, financial flow, action diversity) that capture game-bot signatures missed by DeFi-oriented feature sets.
- **Model architecture.** We design a dual-branch encoder combining typed-edge GAT with GRU-attention, integrated through gated fusion and bootstrapped by graph-contrastive pre-training.
- **Dataset and evaluation.** We construct, to our knowledge, the first multi-game labeled bot dataset spanning three P2E titles on three blockchains (57,879 accounts, 3,551 confirmed bots), and demonstrate that the framework achieves F1 = 94.39%, AUC = 99.92%, and AP = 98.90% on the joint test set with a sharp bimodal score separation between bots and humans.

2 Background

2.1 Play-to-Earn Game Architecture

Despite differing gameplay mechanics, P2E games share a common economic architecture. Players perform in-game actions such as mining, battling, or questing, each of which is recorded as a transaction on a public blockchain. Successful actions yield reward tokens (e.g., TLM in Alien Worlds, JEWEL in DeFi Kingdoms, DEC in Splinterlands) or NFT assets such as characters, tools, and virtual land. Players can then trade these tokens and NFTs

on in-game marketplaces or decentralized exchanges, converting gameplay outcomes into fiat-equivalent value [26]. This closed economic loop creates a direct incentive for automation, as any action that produces on-chain rewards can, in principle, be scripted.

2.2 Bot Behaviors and Their Detectability

To ground the detection problem, we describe three categories of P2E game bots. We distinguish them by their primary objective, but our main concern is the on-chain traces that make each category detectable.

Reward-farming bots repetitively execute the core reward-yielding action (e.g., mining in Alien Worlds, questing in DeFi Kingdoms) with minimal variation. Their automation is exposed on-chain through mechanical regularity in action timing, narrow action-type diversity, and rapid withdrawal of earned tokens.

Market-manipulation bots exploit marketplace mechanics through automated sniping, wash trading [19], or price manipulation. Their footprint extends beyond dense intra-cluster transaction patterns and repetitive trading sequences confined to a narrow set of counterparties. Automated market engagement produces abnormal timing regularity in listing and purchase actions; wash trading generates artificial volume, visible as high gross turnover with near-zero net change in inventory or funds; inventory cycling repeatedly re-lists and re-trades the same small set of NFTs; and coordinated activity across controlled accounts leaves reciprocal, circular value flows within a closed group, visible from each participating account's own history. Such trades are often economically irrational once marketplace fees and gas are accounted for, and systematic price-distortion strategies, for example self-dealt sales that step an asset's price upward before an outside sale, produce price trajectories decoupled from organic demand.

Sybil bots operate coordinated clusters of accounts to circumvent per-account reward limits or to farm airdrops across many controlled accounts. They leave convergent, funnel-shaped fund flows toward a small number of collection addresses, together with homogeneous behavioral profiles across the controlled accounts.

These categories are not mutually exclusive, and a single operator may combine them; we use them not as a strict classification but as a guide to the behavioral signals a detector must capture, all of which are observable from on-chain data alone. Reward-farming bots, which we expect to constitute the majority of automated accounts in P2E ecosystems, and market-manipulation bots are detectable at the level of an individual account, whereas Sybil bots are only partially so: their convergent structure and profile homogeneity are visible locally, but explicit cross-account synchronization requires cluster-level analysis (Section 3.3). The framework proposed in this work therefore targets per-account detection, capturing the first two categories in full and the locally observable component of the third.

2.3 On-Chain Observability

Although some games execute gameplay logic off-chain (e.g., Splinterlands resolves card battles on a centralized server), the economic outcomes that bots are designed to exploit, such as reward claims, NFT transfers, and marketplace trades, always settle as public, immutable transactions. Detection is therefore possible from on-chain data alone, without cooperation from game operators.

2.4 Limitations of Existing Approaches

Despite a rich body of work on game-bot detection and blockchain anomaly detection, none of the existing approaches is directly applicable to Web3 P2E games; a detailed review with citations appears in Section 7. In brief, mature game-bot detectors operate on proprietary server-side telemetry that Web3 game operators do not necessarily expose and community-side stakeholders never see, while blockchain-native anomaly detectors rely on feature sets engineered for DeFi-specific behaviors that do not arise in P2E game contexts. Moreover, existing studies evaluate within one game on one blockchain, and no public ground-truth dataset exists: the closest prior work on game-bot behavior in Web3 [6] is a model-free clustering analysis on a single title without supervised labels. These four gaps motivate a framework that is (a) purely on-chain, (b) game-agnostic in its feature design, (c) evaluated across multiple games and chains, and (d) accompanied by a labeled dataset.

3 Problem Motivation and Threat Model

This section motivates the proposed framework and defines its threat model.

3.1 Detection Feasibility from On-Chain Behavior

The central conjecture of this work is that bot accounts and human accounts in P2E games occupy distinguishable regions of a behavioral feature space defined purely over public on-chain data. We support this conjecture from two empirical angles, prior to any model training.

Raw-feature bimodality

Applying four simple, independent threshold-based indicators of known bot characteristics to raw on-chain behavior – (H1) transaction count > 100 within the collection window; (H2) action-type entropy < 0.5 bit; (H3) 24/7 activity ratio > 0.5; and (H4) coefficient of variation (CV) of daily volume < 0.2 – we find that 86.8% of accounts in our 57,879-account dataset satisfy none of the four rules and 1.4% satisfy at least three simultaneously, leaving only 11.8% in the ambiguous 1–2-rule zone. Even without learned representations, the bot and human populations thus exhibit strong bimodality on independently chosen behavioral axes.

Economic infeasibility of full camouflage

Even a sophisticated bot operator cannot simultaneously erase all four signals. Smoothing inter-action timing requires inserting idle delays that reduce reward throughput proportionally; diversifying actions forces engagement with unprofitable game mechanics; randomizing withdrawal cadence increases exposure to reward-token volatility; obfuscating graph structure requires routing through additional accounts, each of which incurs gas. Each evasion lever reduces the very economic advantage that motivated automation in the first place. This asymmetry implies that the behavioral subspace bots occupy is structurally constrained, not merely empirically common.

Together, these observations motivate classifying accounts from public on-chain histories alone.

3.2 Detection Challenges

A practical detector for P2E bots must overcome the following challenges:

C1: Cross-chain heterogeneity. The transaction representations, action vocabularies, and account models of EVM chains (DeFi Kingdoms on DFK Chain), EOSIO chains (Alien Worlds on WAX), and DPoS chains (Splinterlands on Hive) differ substantially. A detector must abstract over these differences without losing behavioral resolution.

C2: Cross-game semantic heterogeneity. Different titles employ different reward mechanics (idle mining vs. RPG questing vs. card battling) and different action vocabularies, yet share the same underlying bot objective. A detector must learn signatures that transfer across game mechanics.

C3: Severe class imbalance. Bot accounts represent only 6.1% of the labeled dataset, varying from 1.2% (Splinterlands) to 6.3% (Alien Worlds). Standard cross-entropy training fails to recover minority-class structure under this skew.

C4: Absence of public labeled datasets. No prior labeled dataset exists. Ground truth must be constructed from scratch using a labeling pipeline that is both scalable across tens of thousands of accounts and traceable enough to audit for label noise.

C5: Label noise from heuristic priors. Annotating ground truth at the scale of tens of thousands of accounts is aided by heuristic indicators that summarize known bot characteristics. If annotators lean on such indicators and the indicators overlap with the features the model later learns to use, the evaluation risks circularity: the model may merely recover the heuristic rule rather than learn a generalized bot signature.

3.3 Threat Model and Scope

Adversary capabilities

The adversary controls an arbitrary number of EOA-style accounts on one or more supported chains and submits transactions through the same public interfaces as legitimate players. The adversary may (i) automate execution of any in-game action permitted by the protocol, (ii) adjust transaction timing within bounds compatible with profitability, (iii) route funds through intermediate addresses, and (iv) vary action mix to partially mimic human play.

Adversary limitations

The adversary cannot forge on-chain timestamps, conceal completed transactions, or transact without paying associated gas or network fees. Any behavioral camouflage reduces the bot's effective reward-per-time. The adversary has no access to defender-side training pipelines and cannot tamper with the labeled dataset post hoc.

Defender goals

The defender is any stakeholder with read-only access to the chain (game developer, community watchdog, exchange compliance team). Given a target account a and its on-chain transaction history within an observation window, the defender outputs a binary label $\hat{y}_a \in \{0, 1\}$ indicating bot status. Detection latency on the order of hours is acceptable; sub-block-time inference is not required.

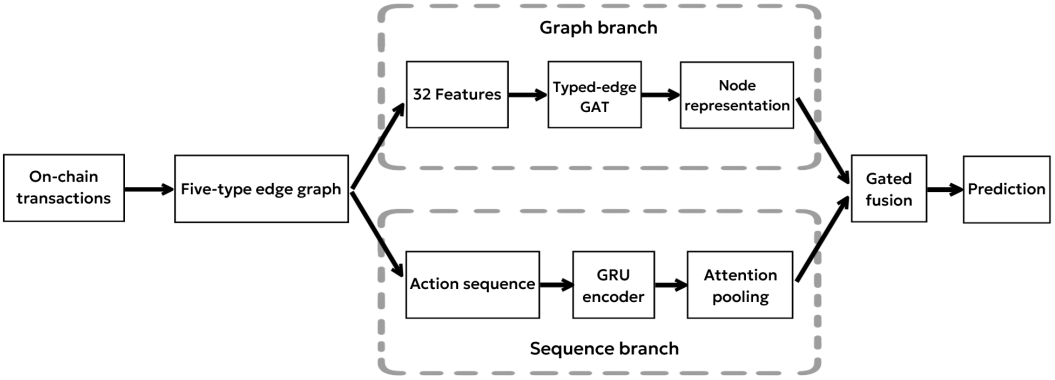


Figure 1 Overview of the proposed framework. On-chain transactions from three P2E games are abstracted into a unified five-type edge graph, which feeds two parallel branches: a graph branch that encodes 32 node features with a typed-edge GAT, and a sequence branch that encodes the per-account action sequence with a GRU and attention pooling. A gated fusion combines the two embeddings, and a classifier predicts whether the account is a bot or a human.

Scope

We focus on per-account classification using a single observation window. The following are out of scope: (i) explicit detection of cross-account Sybil coordination (which requires pairwise or cluster-level analysis); (ii) adversarial robustness against bots that adapt their behavior in response to a public detector model; (iii) prevention or rollback of bot transactions at execution time. Sybil cluster detection and continual-learning extensions are discussed as future work in Section 6.

4 Proposed Framework

We first formalize the detection problem and then describe each component of the framework; Figure 1 gives an overview.

Problem Definition

Let $\mathcal{A} = \{a_1, \ldots, a_n\}$ be the set of player accounts observed during the collection period $[t_s, t_e]$ with at least $m = 10$ on-chain actions. Each account a_i is associated with a set of on-chain transactions. The goal is to learn a function $f\colon \mathcal{A} \to \{0, 1\}$ that classifies each account as human (0) or bot (1), using only publicly available on-chain data.

4.1 Heterogeneous Temporal Transaction Graph

We model on-chain activity as a heterogeneous temporal graph $G = (V, E, \tau, \varphi, w)$, where V is the node set (player accounts), $E \subseteq V \times V$ is the directed edge set (transactions), $\tau\colon E \to \mathbb{R}$ assigns a Unix timestamp, $\varphi\colon E \to \mathcal{T}$ assigns a type from a predefined vocabulary, and $w\colon E \to \mathbb{R}$ assigns a normalized transaction value.

To address challenges C1 and C2 (cross-chain and cross-game heterogeneity), we define a unified type vocabulary

$$\mathcal{T} = \{\mathit{reward},\ \mathit{token},\ \mathit{nft},\ \mathit{action},\ \mathit{market}\}$$

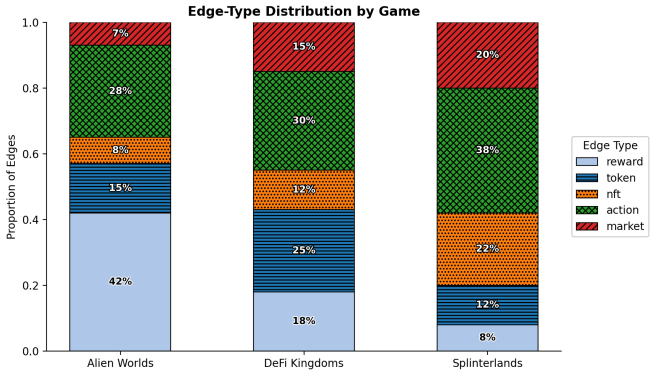


Figure 2 Edge-type composition across the three games. Distributions differ substantially in dominance pattern, motivating the typed-edge GAT design.

Table 1 Edge-type mapping per game.

Type	Alien Worlds	DeFi Kingdoms	Splinterlands
reward	`logmine`	Quest reward	`sm_claim_rewards`
token	TLM transfer	JEWEL transfer	`sm_token_transfer`
nft	`atomicassets`	Hero ERC-721	`sm_sell_cards`
action	`setbag`/`setland`	Hero quest TX	`sm_find_match`
market	AtomicMarket	DEX (UniswapV2)	`sm_market_*`

and map each game's native action set onto it. Table 1 details this mapping. The five types were derived by analyzing the on-chain action vocabularies of all three games and identifying the minimal set of semantically distinct categories that (a) every game instantiates and (b) separates behaviors with different economic implications.

Each edge carries a scalar weight representing the transaction value, normalized per-game via log-transform followed by min-max scaling:

$$w'(e) = \frac{\log(w(e)+1) - w^g_{\min}}{w^g_{\max} - w^g_{\min}} \tag{1}$$

where $w^g_{\min}$ and $w^g_{\max}$ are the per-game log-value extremes.

Figure 2 illustrates the empirical edge-type composition of each game's transaction graph. The three games exhibit substantially different distributions: Alien Worlds is dominated by reward edges (42%) reflecting its high-frequency mining mechanics, DeFi Kingdoms is balanced across action (30%) and token (25%) edges reflecting its RPG-quest economy, and Splinterlands is dominated by action (38%) and nft (22%) edges reflecting its card-trading marketplace. This heterogeneity motivates the typed-edge GAT design in Section 4.3: a uniform-edge GNN would lose the type-conditional aggregation signal that differentiates, for example, a reward claim from a marketplace trade.

4.2 Feature Engineering

For each node $v \in V$, we extract a 32-dimensional feature vector $\mathbf{x}_v \in \mathbb{R}^{32}$ organized into four groups. Features are computable from the five universal transaction fields (sender, receiver, timestamp, value, action type) available on all three chains.

4.2.1 Temporal Regularity ($\mathbf{F_T}$, 8 features)

The defining characteristic of bots is mechanical timing. Let $\Delta_1, \ldots, \Delta_{n-1}$ be the inter-action intervals. We compute: (i) coefficient of variation $\mathrm{CV}(\Delta) = \sigma(\Delta)/\mu(\Delta)$; (ii) hourly entropy $H = -\sum_{h=1}^{24} p_h \log_2 p_h$; (iii) 24/7 activity ratio; (iv) burst count (clusters separated by >1h gaps); (v) lag-1 autocorrelation; (vi) longest continuous streak; (vii) weekend/weekday ratio; and (viii) session duration CV.

4.2.2 Graph Topology ($\mathbf{F_G}$, 8 features)

Transaction graph structure encodes interaction scope: (i) in-degree; (ii) out-degree; (iii) degree ratio; (iv) local clustering coefficient; (v) unique counterparty count; (vi) reciprocity; (vii) PageRank centrality; and (viii) game-contract interaction ratio.

4.2.3 Financial Flow ($\mathbf{F_F}$, 8 features)

Bots optimize for rapid value extraction: (i) reward-to-withdrawal latency; (ii) earned-to-withdrawn ratio; (iii) value Gini coefficient; (iv) median transaction value; (v) withdrawal target count; (vi) max-to-total value ratio; (vii) net flow; and (viii) primary reward token fraction.

4.2.4 Behavioral Diversity ($\mathbf{F_D}$, 8 features)

Human players engage in varied activities; bots repetitively execute a narrow action set: (i) action type entropy; (ii) transition entropy over consecutive edge-type pairs; (iii) action-only ratio; (iv) NFT ID diversity; (v) daily volume CV; (vi) trigram uniqueness ratio; (vii) marketplace participation frequency; and (viii) active-day ratio.

Table 2 summarizes all 32 features together with their expected bot-side signature, i.e., the direction in which each feature is predicted to deviate for automated accounts relative to humans. The signatures are mutually constraining in the sense discussed in Section 3.1: a bot can soften any single feature only by accepting a proportional loss of throughput on another.

4.3 Model Architecture

Our framework employs a dual-branch encoder that separately models spatial and temporal dimensions of on-chain behavior, combined through gated fusion; the encoder is bootstrapped by a contrastive pre-training stage described later in this section. The dual-branch design is motivated by the observation that bot signatures manifest in two complementary modalities: (a) the local graph neighborhood (counterparty topology, financial flow shape) and (b) the chronological action sequence (timing regularity, action-type transitions). Either modality alone is insufficient to capture all three bot types described in Section 2.

4.3.1 Spatial Branch (Typed-Edge GAT)

The 32-dimensional feature vector serves as the initial node representation $\mathbf{h}_v^{(0)} = \mathbf{W}_0 \mathbf{x}_v$. Multi-edges between each node pair are aggregated into composite edge features by concatenating value statistics, edge count, type histogram, time span, and sinusoidal positional

Table 2 The 32 chain-agnostic behavioral features grouped into four complementary categories. The third column states the qualitative direction in which the feature is expected to deviate for automated accounts.

Group	Feature	Bot-side signature
$\mathbf{F_T}$ Temporal	`ft1_interval_cv`	Low (mechanical timing)
	`ft2_hourly_entropy`	Low or very high
	`ft3_247_ratio`	High (round-the-clock activity)
	`ft4_burst_count`	Low (no rest periods)
	`ft5_lag1_autocorr`	High (periodic)
	`ft6_max_streak`	High (long continuous runs)
	`ft7_weekend_ratio`	≈1 (no weekday/weekend distinction)
	`ft8_session_cv`	Low (uniform sessions)
$\mathbf{F_G}$ Topology	`fg1_in_degree`	Low (few incoming sources)
	`fg2_out_degree`	Low (few outgoing targets)
	`fg3_degree_ratio`	Skewed (one-way flow)
	`fg4_clustering`	Low (isolated, star-shaped)
	`fg5_counterparty`	Low (few unique counterparties)
	`fg6_reciprocity`	Low (one-way relationships)
	`fg7_pagerank`	Low (peripheral)
	`fg8_gc_ratio`	High (concentrated on game contracts)
$\mathbf{F_F}$ Financial	`ff1_r2w_latency`	Low (immediate withdrawal)
	`ff2_e2w_ratio`	≈1 (drain everything earned)
	`ff3_gini`	Low (uniform-sized transactions)
	`ff4_median_value`	Low (repeated small amounts)
	`ff5_withdrawal_targets`	Low (1–2 collection addresses)
	`ff6_max_value_ratio`	Low (no large one-off transfers)
	`ff7_net_flow`	Strong net outflow
	`ff8_reward_fraction`	High (reward-only revenue)
$\mathbf{F_D}$ Diversity	`fd1_action_entropy`	Low (single action repeated)
	`fd2_transition_entropy`	Low (same transition repeated)
	`fd3_action_only_ratio`	Low (no genuine gameplay)
	`fd4_nft_id_diversity`	Low (same item reused)
	`fd5_daily_volume_cv`	Low (constant daily volume)
	`fd6_trigram_unique`	Low (same 3-action pattern)
	`fd7_market_freq`	Low or extreme
	`fd8_active_day_ratio`	High (active every day)

encoding. We apply a two-layer Graph Attention Network [18] with $K = 4$ attention heads, extended with edge-type-dependent parameters in the spirit of heterogeneous graph attention networks [20, 7]. The attention coefficient incorporates edge features:

$$\alpha_{ij}^{(l)} = \mathrm{softmax}_j\Big(\mathrm{LeakyReLU}\Big(\mathbf{a}^{(l)\top}\Big[\mathbf{W}^{(l)}\mathbf{h}_i \,\|\, \mathbf{W}^{(l)}\mathbf{h}_j \,\|\, \mathbf{W}_e^{(l)}\mathbf{e}_{ij}\Big]\Big)\Big) \tag{2}$$

In Eq. (2), the edge-feature term $\mathbf{W}_e^{(l)}\mathbf{e}_{ij}$ lets the attention weight depend on the edge type, so that a reward edge and a market edge between the same pair of accounts contribute differently. The two layers are stacked with residual connections and layer normalization, and the final-layer node embedding is taken as the spatial representation $\mathbf{h}_{\text{spatial}}$.

4.3.2 Temporal Branch (GRU-Attention)

For each account v, we extract the chronologically ordered action sequence $S_v = [(c_1, a_1, \Delta t_1, w_1), \ldots]$ where c_i is the contract identifier, a_i the action type, Δt_i the log-transformed inter-action time, and w_i the transaction value. Each tuple is embedded and fed into a single-layer GRU. An attention mechanism pools over hidden states:

$$\alpha_t = \mathrm{softmax}\big(\mathbf{w}^\top \tanh(\mathbf{W}_a \mathbf{h}_t + \mathbf{b}_a)\big) \tag{3}$$
$$\mathbf{h}_{\text{temporal}} = \textstyle\sum_t \alpha_t \mathbf{h}_t$$

The attention weights α_t in Eq. (3) let the model emphasize the time steps most indicative of automation, and the weighted sum yields the temporal representation $\mathbf{h}_{\text{temporal}}$.

4.3.3 Gated Fusion

We combine the two branch outputs through a learned element-wise gate that allows the model to dynamically weight spatial versus temporal evidence on a per-account basis:

$$\mathbf{g} = \sigma(\mathbf{W}_g[\mathbf{h}_{\text{spatial}} \,\|\, \mathbf{h}_{\text{temporal}}] + \mathbf{b}_g) \tag{4}$$
$$\mathbf{h}_{\text{fused}} = \mathbf{g} \odot \mathbf{h}_{\text{spatial}} + (1 - \mathbf{g}) \odot \mathbf{h}_{\text{temporal}} \tag{5}$$

The gate $\mathbf{g}$ in Eq. (4) is computed per account, so an account whose bot signature is mainly structural and one whose signature is mainly temporal can be weighted differently in the fused embedding of Eq. (5).

4.3.4 Contrastive Pre-training

To address C4 (label scarcity), we pre-train the encoder using NT-Xent loss [4] on unlabeled data, following the graph augmentation paradigm established by GraphCL [25]. Two augmented views of the graph are generated by independently applying edge dropout ($p_e = 0.2$) and feature masking ($p_m = 0.2$). For a batch of N accounts, the two views yield $2N$ embeddings $\{\mathbf{z}_k\}$; for each anchor i with positive counterpart i^+ (its other view), the loss is

$$\mathcal{L}_{\text{con}} = -\frac{1}{2N} \sum_{i=1}^{2N} \log \frac{\exp(\mathrm{sim}(\mathbf{z}_i, \mathbf{z}_{i^+})/\tau)}{\sum_{k \neq i} \exp(\mathrm{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)} \tag{6}$$

where $\mathrm{sim}(\cdot, \cdot)$ is cosine similarity, $\tau = 0.5$, and the denominator sums over all $2N - 1$ non-anchor embeddings.

4.3.5 Classification Head

The fused embedding is passed through a two-layer MLP with ReLU, dropout ($p = 0.3$), and batch normalization. To address C3 (class imbalance), we use Focal Loss [13] with $\gamma = 2$ and $\alpha_c = 0.75$ for the bot class:

$$\mathcal{L}_{\text{cls}} = -\alpha_c (1 - p_t)^\gamma \log(p_t) \tag{7}$$

In Eq. (7), p_t denotes the predicted probability of the account's ground-truth class. The two factors counteract the 6.1% class imbalance in complementary ways: the class weight $\alpha_c = 0.75$ scales up the loss of the rare bot class relative to the human majority, while the modulating factor $(1 - p_t)^\gamma$ decays the contribution of well-classified examples – with $\gamma = 2$,

an example classified with $p_t = 0.9$ contributes 100× less than under standard cross-entropy – so the abundant, mostly easy human accounts do not dominate the gradient and training concentrates on the hard examples near the decision boundary. The total supervised loss combines the focal classification loss of Eq. (7) with the contrastive term of Eq. (6) as a regularizer, $\mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda \mathcal{L}_{\text{con}}$ with $\lambda = 0.01$.

4.3.6 Training Procedure

Training proceeds in two phases. Phase 1 pre-trains the entire dual-branch encoder with the contrastive objective alone, using two independently augmented views of the graph so that the encoder learns bot-discriminative structure before any label is consumed. Phase 2 fine-tunes the whole model with the supervised focal loss while retaining the contrastive term as a regularizer; at each epoch the decision threshold $\theta^\star$ is chosen to maximize validation F1, and the model with the best validation F1 is retained for test-time inference.

4.4 Ground Truth Creation

To address C4 (no public labeled dataset), we adopt a two-stage labeling process designed for auditability. The procedure follows the ground-truth creation methodology of Niedermayer et al. [14], adapting it from individual Ethereum addresses to P2E game accounts.

Stage 1: Heuristic Reference Scoring

We reuse the four threshold-based indicators H1–H4 introduced in Section 3.1 (transaction volume, action-type entropy, 24/7 activity, and daily-volume variability). Each account receives a heuristic score equal to the number of rules it satisfies. This score is provided to annotators purely as a reference signal; it does not select, filter, or pre-label accounts, and every account in the dataset proceeds to annotation in Stage 2 regardless of its score.

Stage 2: Annotator Validation

Two independent annotators label each account by examining its raw on-chain activity (its transaction history and behavioral feature profile) together with the Stage 1 heuristic score as a supporting reference, and assign a binary label, with disagreements resolved through discussion until consensus. Because the decision is grounded in the raw activity, annotators may reach a conclusion that diverges from what the heuristic score alone would suggest. The resulting dataset comprises 3,551 bot and 54,328 human accounts across the three games; its per-game composition is reported later in Table 4.

Addressing Potential Circularity (C5)

Because the heuristic score is only a reference signal and never assigns or filters labels, the ground truth is determined entirely by annotator judgment over the full feature profile of every account. The reported performance therefore reflects agreement with human judgment rather than recovery of the heuristic rules.

4.5 Data Collection

We collect on-chain transaction data from three operational P2E games over a 31-day period (March 1–April 1, 2026) using publicly available blockchain APIs. The three games were selected to maximize diversity along three dimensions: blockchain architecture (EVM, EOSIO, DPoS), gameplay mechanics (idle mining, RPG questing, card battling), and economic scale. Table 3 summarizes data sources.

Table 3 Data sources and collection methods.

Game	Blockchain	API
Alien Worlds	WAX	Hyperion History
DeFi Kingdoms	DFK Chain	Routescan + RPC
Splinterlands	Hive	Condenser API

Table 4 Dataset and transaction graph statistics per game.

Game	Accounts (Nodes)	Bots	Bot %	Edges	Unique Pairs	Edges/Nodes
Alien Worlds	54,298	3,440	6.3%	2,421,307	132,854	44.6
DeFi Kingdoms	3,257	107	3.3%	318,426	21,493	97.8
Splinterlands	324	4	1.2%	34,275	2,144	105.8
Total	57,879	3,551	6.1%	2,774,008	156,491	47.9

Ethical Considerations

All data consists of publicly broadcast blockchain transactions with no personally identifiable information – addresses are pseudonymous identifiers not linked to real-world identities – and the research involves no human participants and no private or off-chain data, so it does not fall under institutional review board (IRB) oversight. Given the dual-use nature of bot detection, where a false positive could unfairly penalize a legitimate player, we frame the framework as a decision-support tool for human review rather than a basis for automated enforcement.

5 Evaluation

We design experiments to answer the following research questions:

- **RQ1 (Detection Performance):** How effective is the proposed framework for bot detection on the joint dataset?
- **RQ2 (Feature Contribution):** Which feature groups contribute most to detection performance, and how does the model behave during training?
- **RQ3 (Model Analysis):** What do the learned representations reveal about the decision process?

5.1 Experimental Setup

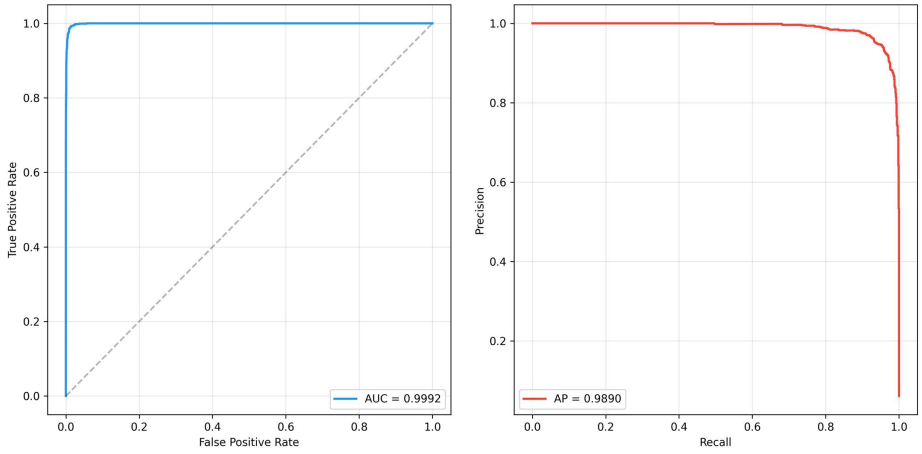
5.1.1 Dataset and Graph Statistics

Table 4 summarizes the labeled dataset and the per-game transaction graphs. The data is split via stratified sampling (70% train, 30% test) per game, ensuring bot representation in both splits. Validation is drawn as an internal 85/15 split from the training set for early stopping.

The graphs vary substantially in scale and density: Alien Worlds produces by far the largest transaction graph (over 2.4M edges) due to its high-frequency mining mechanics, whereas the smaller games are denser per node, with Splinterlands reaching the highest edges-per-node ratio as its card-trading marketplace concentrates activity among fewer accounts.

Table 5 Detection performance on the joint test set.

Metric	Value
Accuracy	99.30%
Precision	92.52%
Recall	96.34%
F1-Score	94.39%
AUC-ROC	99.92%
AP	98.90%
Optimal Threshold	0.480



(a) (b)

Figure 3 (a) ROC curve (AUC = 0.9992). (b) Precision-recall curve (AP = 0.9890). Both indicate near-perfect ranking ability despite the strong class imbalance.

5.1.2 Implementation Details

The dual-branch encoder uses $d = 128$, $K = 4$ attention heads, 2 GAT layers, and GRU hidden dimension 128. We use Adam optimizer ($\text{lr} = 10^{-3}$, weight decay 10^{-5}) with cosine annealing ($\eta_{\min} = 10^{-5}$). Contrastive pre-training runs for 80 epochs; fine-tuning for up to 500 epochs with early stopping (patience 80) on validation F1. The decision threshold is optimized on the validation set via grid search over $[0.05, 0.95]$ with step 0.01.

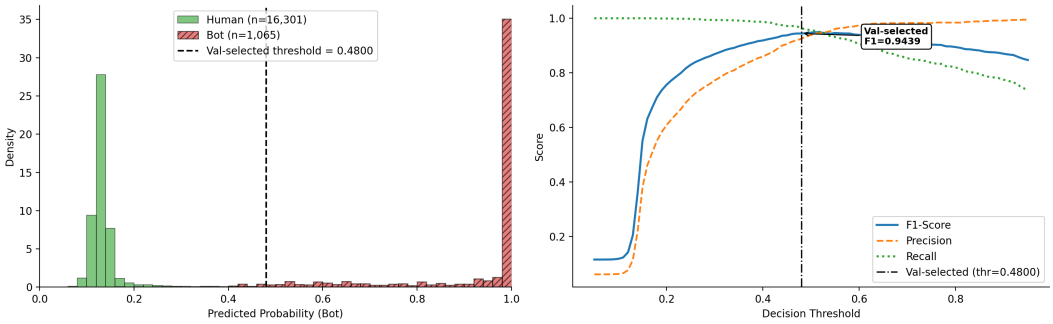
5.2 RQ1: Detection Performance

Table 5 reports the detection performance of the proposed framework on the joint test set.

The model achieves an F1-score of 94.39% and AUC of 99.92% on the joint test set, demonstrating strong discriminative ability despite the 6.1% class imbalance.

Score Separation

Figure 4(a) shows the predicted probability distributions for human and bot accounts. The distributions exhibit a sharp bimodal separation: human accounts concentrate near probability 0.13 and bot accounts near 0.99, with only about 5% of accounts falling in the



(a) (b)

Figure 4 (a) Predicted probability distributions for human (green) and bot (red, hatched) accounts; the dashed line marks the validation-selected decision threshold (0.48). (b) Threshold sensitivity for F1 (blue, solid), precision (orange, dashed), and recall (green, dotted); F1 remains above 0.90 across $[0.36, 0.77]$, and the validation-selected threshold of 0.48 yields F1 = 0.9439.

intermediate range $[0.2, 0.9]$. This clean separation indicates that the model makes highly confident predictions for the vast majority of accounts, rather than relying on a narrow margin around the threshold. The bimodality observed at the raw-feature level in Section 3.1 thus survives, and is in fact sharpened by, the learned representation.

Threshold Robustness

Figure 4(b) shows the relationship between decision threshold and detection metrics. The F1-score remains above 0.90 across the wide range $[0.36, 0.77]$, indicating that the model's performance is not brittle with respect to threshold selection. The threshold of 0.48, selected on the validation split, achieves F1 = 0.9439 with precision 92.52% and recall 96.34%.

Ranking Quality

The ROC and precision-recall curves (Figure 3) achieve AUC = 0.9992 and AP = 0.9890 respectively. The latter is particularly informative given the class imbalance: the model maintains precision above 0.95 up to a recall of roughly 0.9, only dropping more steeply as recall approaches 1.0.

5.3 RQ2: Feature Contribution

Feature Group Ablation

We quantify the contribution of each of the four feature groups ($\mathbf{F_T}$, $\mathbf{F_G}$, $\mathbf{F_F}$, $\mathbf{F_D}$) by masking the corresponding columns at inference time (setting them to zero) and re-evaluating the trained model on the same test set. Figure 5 reports the resulting F1 scores. The full model reaches F1 = 0.9439 on the ablation run; removing any one group degrades performance, but the degradation is highly uneven. The Diversity group $\mathbf{F_D}$ is by far the most critical: masking it drops F1 to 0.2600 (-0.6839), confirming that behavioral diversity, particularly action-type entropy, transition entropy, and trigram uniqueness, carries the dominant signal distinguishing bots from humans. The Temporal (-0.1778) and Topology (-0.1497) groups contribute moderately, while the Financial group is the least sensitive (-0.0405), consistent with financial outflow patterns being easier for bots to randomize than action-mix patterns.

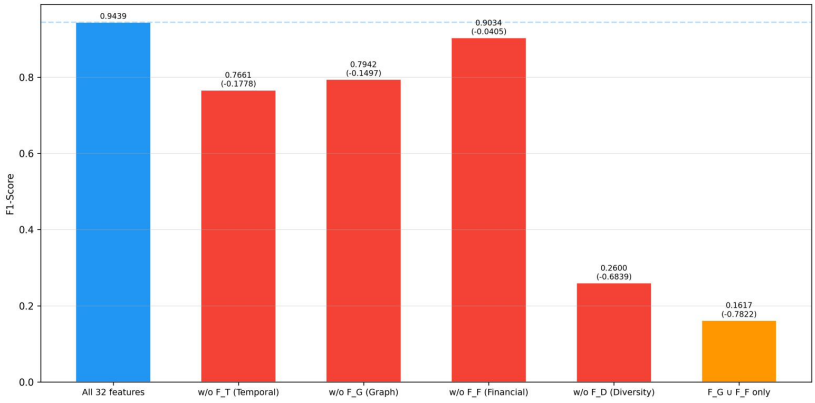


Figure 5 Feature-group ablation by masked inference on the joint test set. The full model attains F1 = 0.9439; masking the Diversity group $\mathbf{F_D}$ causes the largest drop. The rightmost bar reports a model trained on $\mathbf{F_G} \cup \mathbf{F_F}$ alone.

As a further reference point, a model trained on the topology and financial groups alone ($\mathbf{F_G} \cup \mathbf{F_F}$) attains only F1 = 0.1617 (rightmost bar in Figure 5), reinforcing that the temporal and diversity groups carry most of the discriminative signal.

Training Convergence

Figure 8(a) shows the fine-tuning loss and validation metrics over 246 epochs (terminated by early stopping). Validation AUC crosses 0.99 within the first 9 epochs and saturates above 0.999, indicating that the model learns to rank accounts correctly very early, while the threshold-dependent F1 metric requires additional epochs to stabilize, peaking at 0.966 with progressively decreasing variance.

Human versus Bot Feature Separation

The ablation above quantifies *which* feature groups matter; Figure 6 shows *why* they matter, by contrasting the class-conditional distributions of the eight most discriminative features. For every feature, the bot distribution is sharply concentrated while the human distribution is broad. The three behavioral-diversity features (action entropy, transition entropy, and trigram uniqueness) exhibit the clearest separation: bot accounts collapse onto a narrow low-entropy spike, whereas human accounts spread across the full range, reflecting the repetitive action loops that automation cannot avoid without sacrificing throughput. The pattern recurs in the remaining groups. Bots cluster near a reward fraction of 1.0 (almost every transaction is a reward claim), near a giant-component ratio of 1.0 and a clustering coefficient of 0 (star-shaped ego-networks rather than genuine peer interaction), and at near-zero daily-volume variation (mechanically constant activity). This direct distributional evidence corroborates the ablation ranking: the features whose class distributions separate most are exactly those whose removal most degrades F1, and five of the eight belong to the dominant $\mathbf{F_D}$ group.

Cross-Game Feature Distributions

Figure 7 visualizes the distributions of the top four features across the three games. The distributions differ substantially across games (e.g., DeFi Kingdoms shows a narrow spike in `fd1_action_entropy` due to its limited action vocabulary), which motivates the per-game

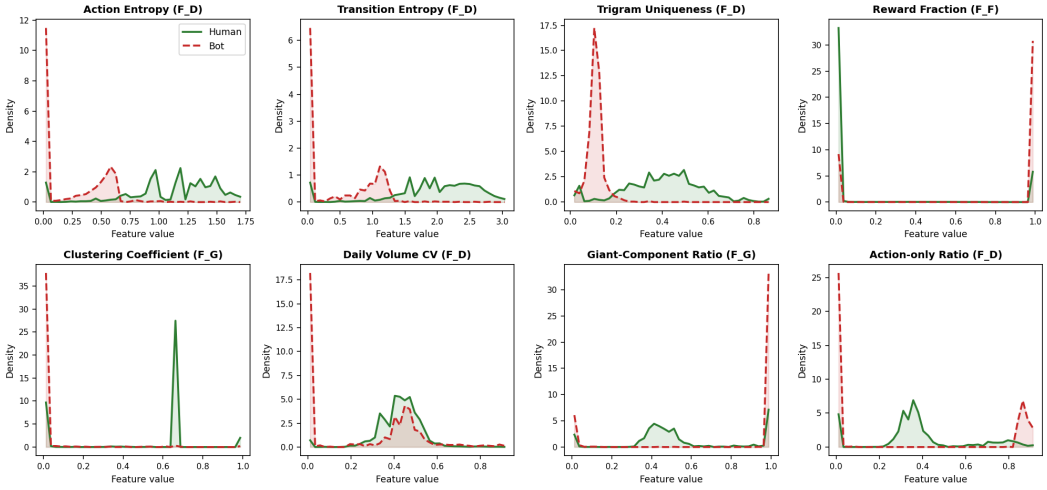


Figure 6 Class-conditional distributions of the eight most discriminative features, ranked by Wasserstein distance between the human and bot distributions. For each feature the x-axis is the (raw) feature value and the y-axis is probability density; the green solid curve is human accounts and the red dashed curve is bot accounts. Bot distributions are consistently concentrated while human distributions are dispersed.

normalization adopted in Section 4.2. Despite these distributional differences, the joint model achieves strong performance, suggesting that the learned representations capture game-invariant bot signatures.

5.4 RQ3: Model Analysis

Learned Representations

Figure 8(b) visualizes the fused embeddings ($\mathbf{h}_{\text{fused}} \in \mathbb{R}^{128}$) of 3,000 randomly sampled test accounts via t-SNE. Bot accounts (red triangles) form several tight, well-separated clusters, while human accounts (green circles) form a more diffuse distribution. The presence of multiple distinct bot clusters suggests that the model learns to distinguish different bot operational profiles (e.g., reward-farming bots vs. market-manipulation bots), even though the training labels are binary, pointing to potential for unsupervised bot sub-type discovery in future work. The clear spatial separation between bot and human clusters corroborates the bimodal score distribution observed in Figure 4.

6 Discussion

Practical Implications

Because the framework relies only on public on-chain data, it can be deployed by any stakeholder (game developers, community watchdogs, or platform operators) without privileged access to game telemetry, and the unified five-type edge vocabulary allows porting to a new P2E title by specifying only a single mapping table.

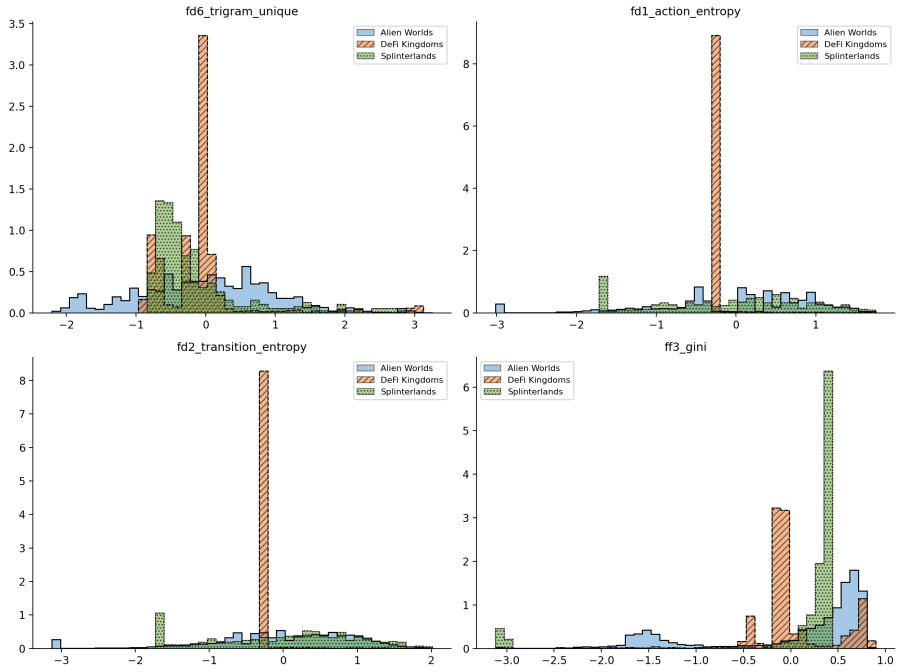
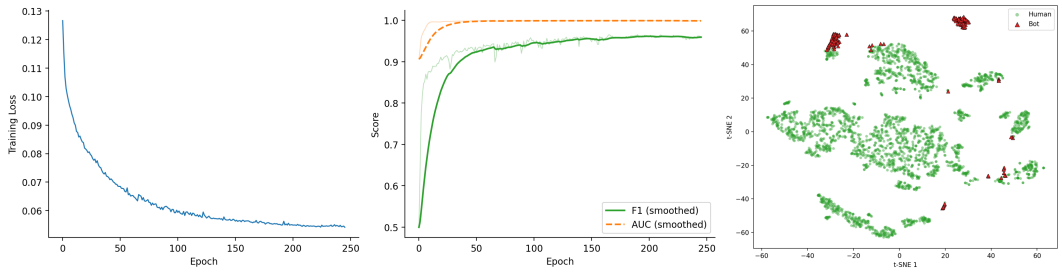


Figure 7 Distributions of the top four features across the three games. The substantial cross-game distributional shifts motivate per-game normalization.



(a) (b)

Figure 8 (a) Training curves, showing fine-tuning loss and validation F1 (green, solid) and AUC (orange, dashed), smoothed with $\beta = 0.9$. (b) t-SNE of fused embeddings, where bot accounts (red triangles) form tight, well-separated clusters and human accounts (green circles) are more dispersed.

Post-Detection Mitigation

Detection is only the first step of an operational response, and the appropriate follow-up depends on who runs the detector and on the cost of a false positive. We envision a graduated, score-driven pipeline rather than a single hard ban decision. Accounts in the high-confidence bot mode of the score distribution (Figure 4) can face automated friction that is cheap for legitimate players but costly for automation, such as rate limits on reward-claim actions, short escrow delays before reward withdrawal, or step-up verification challenges; because such measures throttle rather than ban, they directly erode the reward-per-time advantage that motivates automation (Section 3.1) while remaining reversible if a flag proves incorrect. The roughly 5% of accounts in the intermediate score range should instead be routed to human review before any enforcement, consistent with the decision-support framing of our ethical discussion (Section 4.5); for irreversible actions such as account suspension or confiscation of farmed rewards, operators can shift the decision threshold toward higher precision at a quantified recall cost (Figure 4). Stakeholders without in-game enforcement power, such as community watchdogs and exchange compliance teams, can use flagged-address lists for market-level responses, for example enhanced screening of withdrawal targets. Finally, because bot operators may adapt once a mitigation policy becomes visible, deployment should pair the detector with periodic re-training over rolling observation windows, as discussed under future work.

Limitations

We acknowledge several limitations. First, the framework performs per-node classification and cannot directly detect coordinated Sybil clusters that require cross-account analysis. Second, the unified five-type edge vocabulary necessarily loses game-specific semantic granularity. Third, annotators may still be influenced by the heuristic reference score (Section 4.4), so the ground truth could under-represent sophisticated bots whose raw behavior closely resembles human play. Fourth, our evaluation covers three games, but the Web3 gaming landscape includes hundreds of titles with diverse mechanics; generalization to genres not represented in our dataset remains to be established. Fifth, Splinterlands contributes only 4 bot accounts, which limits the statistical reliability of per-game analysis for this title.

Future Work

We see six promising directions for extending this work. The most immediate is a systematic comparison against alternative classifiers – classical machine-learning baselines on the 32-dimensional feature vector, single-branch ablations, and alternative graph architectures in place of the typed-edge encoder – to more precisely isolate the contribution of the dual-branch design. Pairing this comparison with a time-based train/test split, rather than the random split used here, would further rule out operator-level leakage. A second direction concerns the feature-group ablation of Section 5.3, where we adopted inference-time masking for rapid experimental turnaround; we plan to complement it with ablations that retrain the model from scratch without each feature group. A third is an observation-window sensitivity study: detection currently consumes a 31-day observation window (Section 4.5), and repeating it with one week, one day, and even one hour of history would test whether performance is maintained with far less data. A fourth direction is the detection of Sybil clusters through cross-account synchronization analysis, which would address the per-account scope limitation noted in Section 3.3. A fifth is adaptation to dynamic settings in which bot behavior evolves over time, which we expect to approach through continual learning over rolling observation

windows. Finally, broadening the evaluation to additional Web3 game genres, including fully on-chain games, would test how well the framework generalizes beyond the three titles studied here.

7 Related Work

7.1 Bot Detection in Traditional Online Games

Server-side bot detection in online games has evolved through three methodological generations. Early statistical methods relied on handcrafted behavioral indicators: Yeung et al. [24] modeled aiming-accuracy anomalies with Bayesian inference, Chen et al. [3] identified bots from avatar movement trajectories, and Kang et al. [9] and Thawonmas et al. [17] classified bots from party-play and behavioral-frequency logs. Although effective within their respective games, these approaches require game-specific feature engineering and do not transfer across titles.

Deep sequential models reduced the need for manual feature design. Xu et al. [23] proposed NGUARD+, which captures player behavior patterns through attention-based BiLSTM and hierarchical self-attention networks. Such models achieve strong in-game performance but depend on detailed server-side event logs such as item pickups, skill usage, and movement coordinates, none of which are available in Web3 environments.

More recently, graph, contrastive, and pre-training approaches have addressed structural and label-scarce settings. Lee et al. [10] proposed a self-similarity framework at NDSS showing that behavioral repetitiveness is a game-agnostic bot signature; our $\mathbf{F_T}$ and $\mathbf{F_D}$ groups operationalize this insight through information-theoretic measures. Zhao et al. [27] introduced T-Detector, a trajectory-based pre-trained model achieving cross-game generalization; our contrastive pre-training follows the same paradigm but substitutes spatial trajectories with on-chain transaction sequences. Li et al. [12] applied contrastive learning on finger-operation graphs, and Wang et al. [21] and Qi et al. [16] addressed label scarcity and social signals respectively. A common limitation spans all three generations: each depends on proprietary server-side data unavailable in permissionless settings, and none model the on-chain financial behaviors that constitute the primary objective of P2E game bots.

7.2 Anomaly Detection on Blockchain Graphs

GNN-based methods for blockchain anomaly detection have progressed along two axes. On the temporal modeling axis, Weber et al. [22] established the Elliptic benchmark for anti-money laundering on Bitcoin, and EvolveGCN [15] introduced RNN-coupled weight evolution for dynamic graphs. On the label efficiency axis, GraphCL [25] proposed a graph contrastive learning framework whose augmentation design (edge perturbation, attribute masking) our pre-training directly adapts, and CARE-GNN [5] targeted camouflaged fraudsters; while we do not explicitly model adversarial camouflage, our contrastive objective implicitly encourages representations robust to local perturbations.

Our architecture draws on both lines but applies them to a different problem. TTAGN [11] models temporal transaction relations for Ethereum phishing detection; our temporal branch shares this intuition but extends it to heterogeneous multi-game graphs with typed edges. More broadly, blockchain fraud detection models complex fund-routing topologies (layering, mixing, fan-out/fan-in), whereas game-bot signatures are simple, repetitive action sequences with regular timing and narrow interaction scope.

7.3 Bot Detection on Blockchains

Niedermayer et al. [14] represent the closest methodological precedent to our work. They systematize bot types on Ethereum into seven categories and extract 83 address-level features spanning transaction volume, gas usage, temporal patterns, and smart contract interaction profiles. A Random Forest classifier achieves the strongest supervised performance on a balanced subset of 270 labeled addresses. BotDetect [2] extends this approach into a federated learning framework for EVM chains but inherits the same DeFi-centric feature set.

Two limitations restrict the applicability of these methods to Web3 games. First, the feature set is DeFi-oriented: beyond generic transaction statistics, its contract-level features model only Uniswap-style DEX swaps and ERC-20 transfers, capturing trading rather than gameplay automation. Second, both studies are confined to Ethereum mainnet, whereas P2E games span diverse chain architectures (EVM, EOSIO, Hive DPoS).

Guidi et al. [6] offer a complementary, game-focused perspective through a model-free analysis of Gods Unchained, identifying accounts with abnormally intense activity and locating them within communities of the player battle graph. While the study provides valuable empirical observations, it does not advance to predictive modeling.

Positioned against this literature, our work is, to our knowledge, the first to combine on-chain behavioral analysis with deep graph-temporal modeling in a cross-game, cross-chain framework, extending prior on-chain bot detection [14] and the observations of Guidi et al. [6] from descriptive analysis to predictive, game-specific classification.

8 Conclusion

We presented a cross-game, cross-chain framework that detects Web3 Play-to-Earn game bots from public on-chain transactions alone, using a unified five-type edge vocabulary, 32 chain-agnostic behavioral features, and a dual-branch GAT–GRU encoder. Across three games and three blockchains (57,879 accounts), it attains F1 = 94.39%, AUC = 99.92%, and AP = 98.90%, showing that the economic asymmetry of P2E automation leaves on-chain traces sufficient for detection without per-game retraining.

References

1 Jordan Ali and Gili Vidan. Playing, earning, crashing, and grinding: Axie Infinity and growth crises in the Web3 economy. *Big Data & Society*, 12, 2025. doi:10.1177/20539517251357296.

2 Ahmed Mounsf Rafik Bendada, Abdelaziz Amara Korba, Mouhamed Amine Bouchiha, and Yacine Ghamri-Doudane. BotDetect: A decentralized federated learning framework for detecting financial bots on the EVM blockchains. In *Proc. IEEE Int. Conf. Communications (ICC)*, pages 1103–1108, 2025. doi:10.1109/ICC52391.2025.11161090.

3 Kuan-Ta Chen, Hsing-Kuo Kenneth Pao, and Hong-Chung Chang. Game bot identification based on manifold learning. In *Proc. ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*, pages 21–26, 2008. doi:10.1145/1517494.1517498.

4 Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proc. Int. Conf. Machine Learning (ICML)*, pages 1597–1607, 2020. URL: http://proceedings.mlr.press/v119/chen20j.html.

5 Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S. Yu. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proc. ACM Int. Conf. Information and Knowledge Management (CIKM)*, pages 315–324, 2020. doi:10.1145/3340531.3411903.

6 Barbara Guidi, Andrea Michienzi, and Laura Ricci. Detecting suspicious player behavior in Web3 games: A data-driven analysis of bot accounts. In *Proc. Int. Conf. Information Technology for Social Good (GoodIT)*, pages 364–372, 2024. doi:10.1145/3677525.3678683.

7 Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proc. ACM Web Conf. (WWW)*, pages 2704–2710, 2020. doi:10.1145/3366423.3380027.

8 Woncheol Jeong, Sohee Choi, and Jiwon Yoon. Bots-in-Web3-P2E-Games/Dataset-for-Detecting-Bots-in-Web3-P2E-Games. Dataset, swhId: swh:1:dir:2eef651a0c2d25b011e0754a2f62e4784d985072 (visited on 2026-09-21). URL: https://github.com/WONCHEOL-JEONG/Dataset-for-Detecting-Bots-in-Web3-P2E-Games.git, doi:10.4230/artifacts.28075.

9 Ah Reum Kang, Jiyoung Woo, Juyong Park, and Huy Kang Kim. Online game bot detection based on party-play log analysis. *Computers & Mathematics with Applications*, 65(9):1384–1395, 2013. doi:10.1016/j.camwa.2012.01.034.

10 Eunjo Lee, Jiyoung Woo, Hyoungshick Kim, Aziz Mohaisen, and Huy Kang Kim. You are a game bot!: Uncovering game bots in MMORPGs via self-similarity in the wild. In *Proc. ISOC Network and Distributed System Security Symp. (NDSS)*, pages 1–15, 2016. doi:10.14722/ndss.2016.23436.

11 Sijia Li, Gaopeng Gou, Chang Liu, Chengshang Hou, Zhenzhen Li, and Gang Xiong. TTAGN: Temporal transaction aggregation graph network for Ethereum phishing scams detection. In *Proc. ACM Web Conf. (WWW)*, pages 661–669, 2022. doi:10.1145/3485447.3512226.

12 Wenbin Li, Xiaokai Chu, Yueyang Su, Di Yao, Shiwei Zhao, Runze Wu, Shize Zhang, Jianrong Tao, Hao Deng, and Jingping Bi. FingFormer: Contrastive graph-based finger operation transformer for unsupervised mobile game bot detection. In *Proc. ACM Web Conf. (WWW)*, pages 3367–3375, 2022. doi:10.1145/3485447.3512272.

13 Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, pages 2980–2988, 2017. doi:10.1109/ICCV.2017.324.

14 Thomas Niedermayer, Pietro Saggese, and Bernhard Haslhofer. Detecting financial bots on the Ethereum blockchain. In *Companion Proc. ACM Web Conf. (WWW)*, pages 1742–1751, 2024. doi:10.1145/3589335.3651959.

15 Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. EvolveGCN: Evolving graph convolutional networks for dynamic graphs. In *Proc. AAAI Conf. Artificial Intelligence*, pages 5363–5370, 2020. doi:10.1609/aaai.v34i04.5984.

16 Xianyang Qi, Jiashu Pu, Shiwei Zhao, Runze Wu, and Jianrong Tao. A GNN-enhanced game bot detection model for MMORPGs. In *Proc. Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD)*, pages 316–327, 2022. doi:10.1007/978-3-031-05936-0_25.

17 Ruck Thawonmas, Yoshitaka Kashifuji, and Kuan-Ta Chen. Detection of MMORPG bots based on behavior analysis. In *Proc. Int. Conf. Advances in Computer Entertainment Technology (ACE)*, pages 91–94, 2008. doi:10.1145/1501750.1501770.

18 Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *Proc. Int. Conf. Learning Representations (ICLR)*, 2018.

19 Friedhelm Victor and Andrea Marie Weintraud. Detecting and quantifying wash trading on decentralized cryptocurrency exchanges. In *Proc. ACM Web Conf. (WWW)*, pages 23–32, 2021. doi:10.1145/3442381.3449824.

20 Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. Heterogeneous graph attention network. In *Proc. ACM Web Conf. (WWW)*, pages 2022–2032, 2019. doi:10.1145/3308558.3313562.

21 Yilin Wang, Sha Zhao, Shiwei Zhao, Runze Wu, Yuhong Xu, Jianrong Tao, Tangjie Lv, Shijian Li, Zhipeng Hu, and Gang Pan. PU-Detector: A PU learning-based framework for

real money trading detection in MMORPG. *ACM Trans. Knowl. Discov. Data*, 18(4):1–26, 2024. doi:10.1145/3638561.

22 Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I. Weidele, Claudio Bellei, Tom Robinson, and Charles E. Leiserson. Anti-money laundering in Bitcoin: Experimenting with graph convolutional networks for financial forensics. In *Proc. KDD Workshop on Anomaly Detection in Finance*, 2019. doi:10.48550/arXiv.1908.02591.

23 Jiarong Xu, Yifan Luo, Jianrong Tao, Changjie Fan, Zhou Zhao, and Jiangang Lu. NGUARD+: An attention-based game bot detection framework via player behavior sequences. *ACM Trans. Knowl. Discov. Data*, 14(6):1–24, 2020. doi:10.1145/3399711.

24 Siu Fung Yeung, John C. S. Lui, Jiangchuan Liu, and Jeff Yan. Detecting cheaters for multi-player games: Theory, design and implementation. In *Proc. IEEE Consumer Communications and Networking Conf. (CCNC)*, pages 1178–1182, 2006. doi:10.1109/CCNC.2006.1593224.

25 Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pages 5812–5823, 2020.

26 Jingfan Yu, Mengqian Zhang, Xi Chen, and Zhixuan Fang. SoK: Play-to-earn projects. In *Proc. Int. Conf. Financial Cryptography and Data Security (FC)*, pages 389–407, 2023. doi:10.1007/978-3-031-47751-5_22.

27 Sha Zhao, Junwei Fang, Shiwei Zhao, Runze Wu, Jianrong Tao, Shijian Li, and Gang Pan. T-Detector: A trajectory based pre-trained model for game bot detection in MMORPGs. In *Proc. IEEE Int. Conf. Data Engineering (ICDE)*, pages 992–1003, 2022. doi:10.1109/ICDE53745.2022.00079.

Pre-Model Representation Failures in GNN-Based Smart Contract Vulnerability Detection

Birindwa Prisca Hondi
Carnegie Mellon University Africa, Kigali, Rwanda

Chinonso Philip Nwishienyi
Carnegie Mellon University Africa, Kigali, Rwanda

Charity Wanja Mwaura
Carnegie Mellon University Africa, Kigali, Rwanda

Alia Teto
Carnegie Mellon University Africa, Kigali, Rwanda

Jema David Ndibwile
Carnegie Mellon University Africa, Kigali, Rwanda

Abstract

This paper is a failure analysis of the representation layer underlying GNN-based smart contract vulnerability detectors. These systems convert source code into graphs before any learning takes place; if the graph fails to capture the code's semantics, no model improvement can compensate.

We investigate GNNSCVulDetector and identify four failures. First, structurally different contracts produce byte-for-byte identical graphs, constituting a concrete evasion attack. Second, graph construction is governed by a hardcoded 47-entry variable whitelist (including one duplicate entry), which constrains what the extractor can recognise. As a consequence, identical vulnerabilities with different variable names produce inconsistent graphs, graph quality degrades as naming diverges from the whitelist, and when no entry matches the pipeline produces structural output not grounded in source variables. Third, the `C` node (the graph element representing the external caller that triggers a reentrancy attack) is absent from even the most canonical vulnerable contract in the literature. Fourth, a controlled experiment confirms this as a direct misclassification: a fully exploitable contract is labelled *safe* because the $C \to W$ edge is never constructed.

All four failures are demonstrated experimentally. Current accuracy figures in the literature are measured under conditions that do not expose these failures. We demonstrate one confirmed case of misclassification caused directly by a representation-layer failure; the prevalence of such failures in real-world contract populations remains an open empirical question.

2012 ACM Subject Classification Security and privacy → Systems security; Security and privacy → Software and application security; Computing methodologies → Machine learning

Keywords and phrases Smart contract security, graph neural networks, vulnerability detection, DeFi, reentrancy, feature extraction

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.33

1 Introduction

This paper is a failure analysis. Every experiment targets the representation layer, the point at which source code is converted into a graph, to identify failures that occur *before* the model runs and that model improvement alone cannot resolve.

Decentralized Finance (DeFi) has grown into a complex financial ecosystem built on smart contracts. These contracts handle real assets and when they contain vulnerabilities, the losses are immediate and irreversible. This has motivated the development of automated detection systems, with Graph Neural Networks (GNNs) emerging as a popular approach. Systems such as DeFiGuard [5], DeFiTail [6], GANS-MARL [1] and related systems [7, 4] report detection accuracies between 88.9% and 98.39%.



© Birindwa Prisca Hondi, Chinonso Philip Nwishienyi, Charity Wanja Mwaura, Alia Teto, and Jema David Ndibwile;
licensed under Creative Commons License CC-BY 4.0

These numbers look promising. But they all depend on something that has not been adequately tested: the graph representation itself. Every GNN-based detector works in two steps. First, source code is converted into a graph. Second, the neural network analyses that graph. A GNN trained on millions of contracts cannot detect a vulnerability whose defining features were silently discarded during graph construction. No architectural improvement, no additional training data and no fine-tuning can compensate for information that was never extracted. This paper identifies a class of failures that model improvement cannot address.

We investigated this through four experiments on GNNSCVulDetector [8]. Our findings are:

1. **Structural invariance as an evasion attack.** Contracts with different names, different functions, and added dead code produce byte-for-byte identical graphs. This is a concrete evasion attack: an attacker who renames state variables bypasses detection with no knowledge of the model's training data or architecture.
2. **A hardcoded 47-entry whitelist.** We designed four contracts with identical reentrancy vulnerabilities differing only in variable naming. Identical vulnerabilities produce inconsistent graph encodings, graph quality degrades progressively as naming deviates from the whitelist, and when no entry matches the pipeline produces structural output not grounded in source variables.
3. **Missing caller context: an extraction inconsistency.** The `C` node (the graph element representing the external caller that triggers a reentrancy attack) is generated inconsistently: present for some contracts in the training data but absent for the most well-known vulnerable contract in the literature, with no signal from the pipeline that anything is missing.
4. **Confirmed pre-model failure.** A controlled experiment on a purpose-built minimal contract shows the model outputs label 0 (*safe*) for a fully exploitable contract due to the absent $C \to W$ edge.

These findings show that the pipeline fails at the representation layer before the model runs. The accuracy figures reported in the literature reflect performance under conditions that do not expose these failures.

2 Threat Model

We consider an attacker whose goal is to deploy a reentrancy-vulnerable smart contract that evades GNN-based detection. The attacker has black-box access to the detection pipeline: they know the system uses a GNN-based classifier but have no knowledge of the model's weights, training data, or internal architecture. The attacker's only capability is the ability to modify the source code before deployment, specifically by renaming state variables, functions, or the contract itself, and by inserting semantically inert dead code. The reentrancy vulnerability itself, the external-call-before-state-update pattern, is preserved intact.

Under this threat model, our experiments show that evasion is trivially achievable: Finding 1 demonstrates that surface-level renaming alone produces byte-for-byte identical graphs, and Finding 2 shows that using any state variable name outside the 47-entry whitelist produces a degenerate graph regardless of the vulnerability present. Neither capability requires knowledge of the model. The attacker needs only a text editor.

3 Background and Related Work

3.1 GNN-Based Smart Contract Detection

GNN-based detectors convert smart contract source code into a graph and train a neural network to recognise vulnerability patterns. Systems such as DeFiGuard [5], DeFiTail [6], and GANS-MARL [1] report accuracies between 88.9% and 98.39% on reentrancy and related exploits. GNNSCVulDetector [8], introduced at IJCAI 2020, converts Solidity source code into a graph and classifies contracts using a GNN. This is the system we test.

3.2 What Has Been Tested and What Has Not

Existing evaluations test model accuracy on held-out contract sets. GANS-MARL [1] also tests robustness against obfuscated inputs, finding accuracy drops from 93.8% to 88.3%. However, all of these evaluations assume the graph representation is correct. None examines whether the representation preserves semantic information under modification. This paper fills that gap.

3.3 Adversarial Evasion Research

Prior adversarial work focuses on bypassing the model through crafted inputs, achieving evasion rates of 94–97% [3, 2]. This paper examines a more fundamental question: whether the representation given to the model is faithful in the first place.

4 Experimental Setup

4.1 Environment

All experiments ran on a Kali Linux machine (x86_64). GNNSCVulDetector was originally built against an older TensorFlow/Keras stack, so we reproduced that environment exactly, Python 3.7.17, TensorFlow 1.14.0, Keras 2.2.4, scikit-learn 0.20.2 and protobuf 3.20.3, inside a dedicated virtual environment to avoid dependency conflicts with newer libraries. Because GNNSCVulDetector does not execute correctly under modern TensorFlow releases, reproducing the original software environment was necessary to ensure that observed failures originated from the published implementation rather than from dependency incompatibilities introduced by version drift.

4.2 Model and Pipeline

GNNSCVulDetector [8] works in three steps:

1. `AutoExtractGraph.py` reads the Solidity source code and creates a graph of nodes and edges.
2. `graph2vec.py` converts the graph into numerical vectors.
3. `GNNSCModel.py` trains a GNN classifier on these vectors.

Concretely, a graph here is a structured summary of the contract's logic: each node represents a code element such as a state variable, a function, or an external call, and each edge represents a relationship between them, for example, "this external call happens before this variable is updated." This is conceptually similar to a flowchart of the contract's behaviour, except built automatically from the source code rather than drawn by hand. The GNN in step 3 never sees the original Solidity code; it only ever sees this graph. If the graph omits or distorts a relationship, the model has no way to recover the missing information, regardless of how well it is trained.

Table 1 Baseline performance on reentrancy detection (threshold = 0.45). Validation $F_1 = 0.848$ and AUC = 0.896 represent the performance ceiling when the pipeline operates as designed. False Positive Rate (FPR) is the share of safe contracts incorrectly flagged as vulnerable; AUC (area under ROC curve) summarises the model's ability to distinguish vulnerable from safe contracts across all thresholds, with 1.0 being perfect.

Metric	Training Set	Validation Set
F1-Score	0.933	0.848
Recall	0.926	0.865
Precision	0.940	0.805
False Positive Rate	0.028	0.109
Accuracy	0.955	0.882
AUC	–	0.896

For Experiments 1 and 2, extracted graph files were compared directly at the node, edge, node-feature, and edge-feature levels. Byte-for-byte identity refers to exact equality of the serialised graph representations produced by `AutoExtractGraph.py`. The representation is created in step 1. This is where our investigation focuses.

4.3 Dataset

The dataset contains 6,200 labelled reentrancy contract graphs: 4,948 for training and 1,252 for validation. Source contracts and pre-extracted graph data are provided with the GNNSCVulDetector repository [8].

5 Baseline Performance

Table 1 reports model performance when test contracts come from the same distribution as the training data, that is, contracts using variable names the extractor recognises and vulnerability patterns the graph schema can represent. These figures represent the upper bound of performance when the pipeline operates as designed; Sections 6–9 test what happens when contracts fall outside these conditions. The threshold of 0.45 means the model classifies a contract as vulnerable whenever its output score exceeds 0.45; this value was set by the original authors and is used unchanged throughout our evaluation.

Each experiment that follows tests what happens when contracts fall outside this distribution.

6 Experiment 1: Structural Invariance as an Evasion Attack

6.1 Setup

We took `simple_dao.sol`, a well-known reentrancy-vulnerable contract, and created a modified version with the following changes:

- Contract renamed: `SimpleDAO` → `VaultManager`
- Functions renamed: `donate` → `deposit`, `withdraw` → `requestFunds`, `queryCredit` → `checkBalance`
- State variable renamed: `credit` → `balances`
- Dead code inserted: internal function `_processRequest` that performs no meaningful computation

The reentrancy vulnerability was kept intact. The external call still happens before the state update.

6.2 Original Contract

Listing 1 shows the original vulnerable contract.

Listing 1 simple_dao.sol: original vulnerable contract.

```
pragma solidity ^0.4.24;
contract SimpleDAO {
  mapping (address => uint) public credit;

  function donate(address to) payable public {
    credit[to] += msg.value;
  }

  function withdraw(uint amount) public {
    if (credit[msg.sender] < amount) { throw; }
    require(msg.sender.call.value(amount)()); // call FIRST
    credit[msg.sender] -= amount;             // state update AFTER
  }

  function queryCredit(address to) view public
      returns (uint) {
    return credit[to];
  }
}
```

6.3 Modified Contract

Listing 2 shows the modified version.

Listing 2 simple_dao_modified.sol: renamed and restructured, vulnerability preserved.

```
pragma solidity ^0.4.24;
contract VaultManager {
  mapping (address => uint) public balances;

  function deposit(address to) payable public {
    balances[to] += msg.value;
  }

  function _processRequest(uint amount) internal
      returns (bool) {
    return amount > 0; // dead code
  }

  function requestFunds(uint amount) public {
    if (balances[msg.sender] < amount) { throw; }
    bool ok = _processRequest(amount);
    require(msg.sender.call.value(amount)()); // preserved
    balances[msg.sender] -= amount;
  }

  function checkBalance(address to) view public
      returns (uint) {
    return balances[to];
  }
}
```

6.4 Results

Both contracts were passed through `AutoExtractGraph.py`. Table 2 shows the output.

Table 2 Graph extraction output: original vs. modified. All features are identical across both contracts.

Feature	Original	Modified	Match?
Node count	5	5	✓
Edge count	3	3	✓
Node features	Identical	Identical	✓
Edge features	Identical	Identical	✓
C node present	No	No	✓
Graphs are byte-for-byte identical			

▶ Finding 1 (Evasion Attack). The graph extractor is invariant to contract renaming, variable renaming, function renaming, and dead code insertion. The two contracts, clearly distinct programs, are assigned byte-for-byte identical graph representations. This is not a theoretical edge case: it is a concrete evasion attack. An attacker who renames state variables bypasses detection with no knowledge of the model's training data or architecture. The rename alone is sufficient.

7 Experiment 2: The Variable Name Whitelist

7.1 How the Extractor Constructs the Graph

To understand why Experiment 1 produced identical graphs, we examined the source code of `AutoExtractGraph.py`. The tool does not perform general analysis of the Solidity code. Instead, it looks for specific variable names using a hardcoded list, shown in Listing 3:

Listing 3 Partial whitelist from `AutoExtractGraph.py` (47 entries total, including one duplicate).

```
var_list = [
  'balances[msg.sender]',
  'credit[msg.sender]',
  'credit[to]',
  'accountBalances[msg.sender]',
  'userBalance[msg.sender]',
  'Bal[msg.sender]',
  'payments[msg.sender]',
  'rewardsForA[recipient]',
  # ... 47 entries total
  'msg.sender'
]
```

If a match is found, a node is created in the graph. If no match is found, no node is created. The duplicate entry is `participated[msg.sender]`, which appears at positions 2 and 23 of the list; its presence does not extend coverage but does confirm the list was assembled manually rather than generated systematically.

Table 3 Graph output across four identical vulnerabilities with different variable naming. Three distinct failure modes are evident: inconsistency (A vs. B), degradation (A vs. C), and fabrication (D).

Contract	Variable	Nodes	Edges	Observation
A	`balances[msg.sender]` (exact match)	5	3	Normal graph
B	`userDeposits[msg.sender]` (substring)	5	3	Different edge type
C	`ledger[owner]` (no msg.sender)	3	1	Degraded graph
D	`ledger[depositor]` (zero matches)	3	1	Fabricated S node

7.2 Why Experiment 1 Produced Identical Graphs

Both `credit[msg.sender]` and `balances[msg.sender]` appear on the whitelist. Renaming the variable from `credit` to `balances` moved it to another recognised entry. Both contracts produced the same graph not because the tool understood they share the same vulnerability, but because both names happen to be on the list.

7.3 Experimental Proof of Whitelist Failure

To empirically demonstrate the consequences of the whitelist mechanism, we designed four contracts implementing identical reentrancy vulnerabilities that differ only in the naming of the state variable. All four share the external-call-before-state-update pattern. The four names were chosen to span the whitelist boundary: Contract A uses a name that matches the whitelist exactly, Contract B uses a name that partially overlaps a whitelist entry, Contract C uses a name with no `msg.sender` component and Contract D uses a name that matches no whitelist entry at all.

These results establish three empirical claims. Contracts A and B implement the same vulnerability yet their graphs carry different edge types, so the model receives structurally contradictory encodings for the same attack pattern based solely on variable naming. Moving further from the whitelist, node count drops from 5 to 3 and edge count from 3 to 1 between Contract A and Contract C, with no indication from the pipeline that information has been lost. At the extreme, Contract D has zero whitelist matches yet the extractor still produces 3 nodes and 1 edge: the `S` node it emits does not correspond to any state variable in the source; it is generated by a separate internal code path triggered by the `.call.value()` pattern, producing structural output not grounded in source variables.

▶ Finding 2 (Whitelist Failure). The graph extractor relies on a hardcoded 47-entry variable name whitelist (including one duplicate). Any contract whose state variable name falls outside this list produces a degraded or fabricated graph with no warning from the pipeline. Three failure modes are demonstrated: identical vulnerabilities produce structurally inconsistent graphs when variable names differ (Contracts A and B); graph quality degrades progressively as naming deviates from the whitelist (Contract C); and when no whitelist entry matches, the pipeline fabricates structural output not grounded in any source variable (Contract D). This is a pre-model failure – evasion occurs at the feature extraction stage before any machine learning inference takes place.

8 Experiment 3: The Missing Caller Node – An Extraction Inconsistency

8.1 What the C Node Represents

In GNNSCVulDetector's graph schema, the `C` node represents the caller function context: the external contract that initiates the call into the vulnerable function. For reentrancy attacks, this is not optional context; it is the attack's active agent. A reentrancy attack is defined by an external contract calling the vulnerable function, which then makes an external call back before updating its state. Without a `C` node, the graph cannot represent the attack mechanism at all.

The absence of the `C` node is therefore not a minor limitation. Table 4 shows that `EtherStore.sol` in the training data does receive a $C \to W$ edge, which confirms the schema is capable of encoding the caller relationship under some conditions. The failure is one of consistency: the extractor generates this edge for some contracts and silently omits it for others implementing the same vulnerability, with no indication from the pipeline that the representation is incomplete. This is a conceptual failure in the extraction logic, not a strict impossibility in the schema.

8.2 Results

During graph extraction on both `simple_dao.sol` and its modified variant, the extractor emits:

```
"There is no C node"
```

This occurs for the most well-known reentrancy-vulnerable contract in the literature.

▶ Finding 3 (Extraction Inconsistency). Earlier versions of this work described this issue as a structural inability to encode caller context. Our experiments instead show that the schema can encode this relationship, but the extraction logic fails to do so consistently. The $C \to W$ edge is generated inconsistently across contracts implementing the same vulnerability. `EtherStore.sol` in the training data carries this edge; `simple_dao.sol` and its modified variant do not, despite sharing the same reentrancy pattern. The extractor is therefore not structurally incapable of encoding the caller relationship, but it fails to do so reliably. The consequence is the same: the model receives graphs that omit the cross-contract call dynamic that defines reentrancy for some contracts, with no signal that anything is missing. The detector classifies based on incomplete and inconsistently constructed representations of the attack it is trained to find.

9 Experiment 4: Controlled Confirmation via userFunds.sol

This is the central empirical result of this paper.

9.1 Motivation

Finding 3 establishes that the `C` node is absent for `simple_dao.sol`. To confirm that this absence directly causes a misclassification, we constructed a minimal purpose-built contract (`userFunds.sol`). We used `balances[msg.sender]` – a variable name that is on the whitelist – so that any misclassification could not be attributed to the whitelist failure described in Finding 2. This isolates a single question: whether the missing $C \to W$ edge alone is

Table 4 Graph structure comparison: `EtherStore.sol` vs. `userFunds.sol`. The absent $C \to W$ edge in `userFunds.sol` directly produces the misclassification.

Property	EtherStore.sol	userFunds.sol
Ground truth label	1 (vulnerable)	1 (vulnerable)
C node present	Yes	Yes (isolated)
$C \to W$ edge present	Yes	No
Graph encodes reentrancy	Yes	No
Model prediction	Vulnerable ✓	Safe ×

sufficient to cause the model to mislabel an exploitable contract as safe. `userFunds.sol` was intentionally designed to isolate the caller-edge failure while avoiding whitelist effects, by using `balances[msg.sender]`, a variable name confirmed to be accepted by the extractor. Listing 4 shows the contract.

Listing 4 userFunds.sol: minimal controlled reentrancy contract.

```
pragma solidity ^0.4.18;
contract UserFunds {
  mapping (address => uint256) public balances;

  function deposit() public payable {
    balances[msg.sender] += msg.value;
  }

  function withdrawFunds(uint256 _weiToWithdraw) public {
    require(balances[msg.sender] >= _weiToWithdraw);
    // VULNERABILITY: external call before state update
    msg.sender.call.value(_weiToWithdraw)();
    balances[msg.sender] -= _weiToWithdraw;
  }
}
```

9.2 Graph Output and Comparison

The `C0` node was created but carries a `NULL` connection field and appears in no edge; it is completely isolated. No $C \to W$ edge was generated. An isolated node contributes nothing to the GNN's message-passing computation: because graph neural networks learn by aggregating information along edges, a node with no edges receives no neighbourhood signal and passes none. The `C0` node exists in the graph data structure but is invisible to the model's inference process.

`EtherStore.sol` from the training data carries edge `[1,7,2]` encoding the $C \to W$ relationship. `userFunds.sol`, despite implementing the same vulnerability, produces no such edge (Table 4).

9.3 Model Inference Results

▶ Finding 4. The model achieves a best validation F_1 of 0.8511 on the training distribution yet outputs label 0 (*safe*) for `userFunds.sol`, a contract that is fully exploitable. The failure is located entirely in the graph construction layer, prior to any model inference. The

Table 5 `GNNSCModel.py` inference results, full 250-epoch run.

Metric	Value	Notes
Total epochs run	250	Full run completed
Best training F1 epoch	242	Train F1 = 0.9340
Best validation F1	0.8511	Epoch 218
Final validation F1	0.8292	Epoch 250
Final validation accuracy	0.8786	Epoch 250
Final validation recall	0.8601	Epoch 250
Final validation FPR	0.1118	Epoch 250
Predicted label (userFunds)	0 (safe)	Pre-model failure confirmed

absent $C \to W$ edge means the GNN receives no structural signal encoding the reentrancy relationship. The pipeline produces a graph, the model produces a label, and the label is wrong, not because the model erred, but because the information required to make the correct prediction was never provided to it.

10 Discussion

10.1 Summary of Findings

Our four experiments confirm that representation-layer failures are systematic and occur upstream of the model. Because failures occur before the model runs, downstream improvements cannot compensate.

- Different contracts produce identical graphs, enabling evasion attacks (Finding 1).
- The representation vocabulary is limited to 47 hardcoded entries (including one duplicate), producing inconsistent, degraded, and fabricated output (Finding 2).
- The most critical structural element of the targeted attack, the external caller, is absent from the graph schema. The $C \to W$ edge is generated inconsistently across contracts implementing the same vulnerability, meaning the model receives incomplete representations with no signal that anything is missing (Finding 3).
- A fully exploitable contract is predicted as safe, directly caused by the absent $C \to W$ edge (Finding 4).

These are representation problems, not model problems. The neural network never receives the information needed to make the correct decision.

10.2 What This Means for Reported Accuracy

Our results establish the existence of representation-layer failures and provide one confirmed example in which such a failure leads directly to a misclassification. In particular, Finding 4 shows that a fully exploitable contract can be labelled safe when the graph extraction process omits information required to encode the reentrancy relationship.

At the same time, our study does not measure how frequently these failures occur in real-world contract populations, nor does it estimate their aggregate effect on detector accuracy. Quantifying that impact would require a large-scale corpus study in which contracts are re-extracted and re-evaluated under conditions designed to expose these failure modes.

The extent to which real-world contract populations fall outside the extractor's 47-entry whitelist, and the resulting effect on detection performance, therefore remains an open empirical question. Existing evaluation protocols primarily measure model performance on supplied graph representations; they do not explicitly test the representation-layer failure modes documented in this paper.

10.3 Scope: Why GNNSCVulDetector

This study focuses on GNNSCVulDetector [8] because it is the foundational system in this line of work, widely cited as the basis for subsequent GNN-based detectors, and provides full open-source access to both the graph extraction pipeline and the trained model, making all findings independently reproducible. The failures documented in this paper are experimentally verified only for GNNSCVulDetector. Whether similar vulnerabilities exist in other graph extraction pipelines remains an open empirical question. We hypothesize that systems relying on comparable surface-level matching mechanisms may exhibit related failure modes, but evaluating this requires direct testing of those systems and is left for future work.

10.4 Generalizability as Future Work

We therefore do not claim that the vulnerabilities identified here are widespread. Rather, we present a reproducible methodology that can be applied to additional GNN-based detectors to determine whether similar representation-layer failures occur elsewhere. Applying the renaming-based evaluation protocol used here to other systems would establish whether the failure modes documented in this paper are specific to GNNSCVulDetector's implementation or reflect a broader pattern in graph extraction design.

10.5 Representation Before Model

Most research in smart contract security focuses on improving the model: better architectures, better training, more data. Our findings show that the more urgent problem is upstream, since if the representation is broken, the model cannot compensate. Future work should build representations that capture what code does rather than what code looks like: graphs built from Abstract Syntax Trees instead of name matching, control-flow analysis to detect the true order of external calls and state updates, data-flow analysis to trace values independent of naming, and explicit modelling of cross-contract calling relationships so the external caller is represented rather than omitted. Learned code embeddings such as CodeBERT are a plausible complement, replacing a fixed vocabulary with representations that generalise across naming conventions. Evaluation protocols must be extended accordingly, to cover out-of-whitelist naming, structural transformation, and cross-contract relationships.

These techniques share a common goal: representing program behaviour rather than lexical appearance. An AST-based representation captures syntactic structure independent of identifier naming, control-flow analysis captures execution ordering, and data-flow analysis tracks value dependencies across variables and functions. Together, these approaches reduce sensitivity to superficial code transformations while preserving the semantic relationships relevant to vulnerability detection.

11 Conclusion

This paper demonstrates that the graph representations used by GNN-based smart contract vulnerability detectors do not faithfully capture semantic differences between contracts. We show experimentally that structurally different contracts produce identical graphs, that graph construction is governed by a 47-entry whitelist that produces inconsistent, degraded, and fabricated output, that the $C \rightarrow W$ edge encoding the external caller is generated inconsistently across contracts meaning the graph schema fails to reliably represent the attack's active agent, and that a fully exploitable contract is predicted as safe as a direct consequence.

These failures are silent: the pipeline produces a graph, the model produces a label, and no existing evaluation protocol checks whether the graph faithfully represents the source contract. Existing evaluation protocols do not explicitly evaluate the failure modes documented in this paper.

Closing this gap requires changes at the representation layer: moving from surface-level pattern matching toward semantic analysis of what contracts do, and evaluation protocols that expose the failure modes documented here. Model-level improvements alone are insufficient.

References

1 P. K. Adjei et al. A graph attention network-based multi-agent reinforcement learning framework for robust detection of smart contract vulnerabilities. *Scientific Reports*, 15:29810, 2025.

2 H. Al-Harbi. Detecting anomalies in blockchain transactions using spatial-temporal graph neural networks. *Advances in Management and Intelligent Technology*, 1(1), 2025.

3 J. J. de Leon et al. Smart contract security in decentralized finance: Enhancing vulnerability detection with reinforcement learning. *Applied Sciences*, 15(11):5924, 2025.

4 M. Paramesha, N. L. Rane, and J. Rane. Artificial intelligence, machine learning, deep learning, and blockchain in financial and banking services. *PUMRJ*, 1(2), 2024.

5 S. Xu et al. DeFiGuard: A price manipulation detection service in DeFi using graph neural networks. *IEEE Transactions on Services Computing*, 17(6), 2024. doi:10.1109/TSC.2024.3489439.

6 Y. Zhang et al. Penetrating the hostile: Detecting DeFi protocol exploits through cross-contract analysis. *IEEE Xplore*, 2025.

7 Z. Zhang et al. Scalable blockchain fraud detection using spatial-temporal graph neural networks, 2024.

8 Yuan Zhuang, Zhenguang Liu, Peng Qian, Qi Liu, Xiang Wang, and Qinming He. Smart contract vulnerability detection using graph neural networks. In *IJCAI*, pages 3283–3290, 2020. doi:10.24963/IJCAI.2020/454.

EVM Workloads in the Wild: Evidence for Multi-Dimensional Gas Metering, State Growth, Delayed Execution, and Parallelism

Lioba Heimbach
Category Labs, Zurich, Switzerland

Kushal Babel
Category Labs, New York, NY, USA

Jason Milionis
Category Labs, New York, NY, USA

Abstract

Gas metering on EVM-compatible blockchains assumes that execution conditions are stable: that the resource mix is constant enough to justify collapsing execution costs into a single scalar with fixed relative prices, and that state drift between submission and execution time does not materially alter a transaction's outcome. We measure the extent to which this assumption fails.

We present a trace-level measurement study of EVM workloads on Ethereum (L1) and Base (L2) throughout 2025, sampling 3,000 blocks per day per chain. We decompose each transaction into opcode-level execution gas, intrinsic gas, refunds, and persistent state deltas including storage slots, contract bytecode, and account state. To measure state sensitivity, we re-execute transactions sampled during September 2025 on progressively older blockchain states and record how gas usage, execution outcomes, and storage access patterns change.

We find the resource mix to be far from stable: on Base, storage reads and compute account for 29.2% and 24.3% of execution gas, while Ethereum devotes 34.9% to storage writes. The mix is not stable on the same chain either: Ethereum's gas limit doubling during 2025 shifted its resource profile measurably toward more compute-heavy, Base-like patterns. Base also exhibits a higher fraction of cold storage reads at 49.7%, compared to 39.6% on Ethereum. Persistent state growth, a permanent cost priced as a transient one, reaches 435 GB on Base versus 30 GB on Ethereum, with different composition.

We further find that execution outcomes are equally unstable: gas estimates vary across nearby historical states for 46.0% of transactions on Base, compared to 13.9% on Ethereum, with especially high sensitivity for MEV and DeFi activity. Storage access patterns also diverge across execution states, limiting the effectiveness of access lists and complicating parallel execution.

Our measurements provide an empirical foundation for multi-dimensional gas metering and explicit pricing of state growth. They show that state-sensitive execution behavior complicates workload estimation and transaction parameterization, directly affecting the predictability of transactions' execution and user experience.

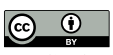
2012 ACM Subject Classification Applied computing → Digital cash

Keywords and phrases Ethereum, EVM, gas metering, state growth, workload analysis, Layer-2

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.34

Related Version *Full Version*: `https://arxiv.org/abs/2606.19869` [23]

Supplementary Material *Software*: `https://github.com/liobaheimbach/EVM-Workloads-in-the-Wild` [22]



© Lioba Heimbach, Kushal Babel, and Jason Milionis;
licensed under Creative Commons License CC-BY 4.0



1 Introduction

Executing a transaction on an EVM-compatible blockchain consumes resources of fundamentally different kinds. Some are transient, i.e., constrained within a block and capacity is reset each block, such as compute, I/O, and bandwidth. Others are permanent: persistent state, i.e., accounts, contract storage, and bytecode, accumulates indefinitely on every node. Gas metering collapses all of these into a single scalar through fixed relative prices, a design that holds only if the resource mix stays constant. A second assumption runs through the transaction submission pipeline: that a transaction behaves at execution as it did in simulation, and that state drift between the two points will not materially change what it accesses, how much gas it uses, or whether it succeeds. Both assumptions are under strain.

Assumption 1 violated: The workload mix shifts. Ethereum meters opcode costs in a single unit of account called *gas*, fixing the relative prices of opcodes (cf. Section 2). EVM-compatible Layer-2 (L2) systems like Base inherit this mechanism unchanged, even though their lower fees and higher throughput drive substantially different on-chain usage patterns than Layer-1 (L1) Ethereum.[1] An appropriate *fixed* mapping from resource usage to relative prices is not necessarily known in advance. More fundamentally, the resource mix itself changes both across chains and over time on the same chain. We find this concretely on Base: its workload is read- and compute-heavy where Ethereum's is write-heavy, and Base's persistent state growth is over an order of magnitude higher than Ethereum's, exceeding what differences in gas limits alone would predict. This state is permanent, yet priced under the same fixed scalar as transient resources. Ethereum has historically adjusted opcode prices and storage rules after discovering that certain operations were systematically mispriced, particularly storage-related opcodes such as `SLOAD` and `SSTORE` [5, 40, 10, 41, 12]; however, the protocol is not built to react to every workload shift. Thus, collapsing a high-dimensional workload into scalar gas is inherently brittle.

Assumption 2 violated: The state moves between simulation and execution. High-throughput systems also change the landscape of opcode pricing because many execution costs depend on the blockchain state. For example, the gas cost of storage writes depends on whether a storage slot already exists and on whether its value transitions between zero and non-zero, making execution costs sensitive to prior state. Control-flow paths also vary with storage and account values, creating execution-context dependencies.

In high-throughput settings, rapid state updates also increase uncertainty about the state a transaction will encounter at execution time. Since gas usage must be estimated and a gas limit specified prior to submission, differences between the estimation state and the execution state can lead to variation in gas usage or even transaction failure.

Emerging execution models further amplify the consequences of such uncertainty. Under delayed or asynchronous execution, incorrect gas limit estimation can incur direct penalties, as charges may be levied based on the declared gas limit rather than actual execution, unlike in synchronous execution where overestimation is largely harmless [14, 34]. For users, high throughput means many transactions may intervene between submission and execution.

As a result, the same transaction may consume different gas, touch different storage keys, or fail when evaluated on different historical states, complicating workload estimation and reducing transaction predictability, with direct consequences for user experience. State-

[1] While L2 transactions also incur additional costs related to posting data to Ethereum L1, these costs are orthogonal to execution on the L2 and are outside the scope of this work.

dependent access patterns further hinder mechanisms that rely on advance knowledge of accessed state, such as access lists, and increase the cost of retries in optimistic parallel execution, where re-execution may occur on a different state.

1.1 Our Contributions

Our goal is to clarify current EVM workload patterns and their dependence on execution state. To this end, we conduct a measurement study on Ethereum L1 and Base, an EVM-compatible optimistic rollup with the largest transaction volume and total value locked (TVL) among L2s [15]. Studying these two systems allows us to contrast execution behavior under differing throughput and block-time regimes while keeping the execution environment (the EVM and its opcode pricing) identical. We pursue the following research questions:

1. Workload mix: what are the differences in opcode patterns and caching behavior, as reflected by cold versus warm access rates, between Ethereum L1 and Base L2?
2. Persistent state growth: how quickly does persistent state grow in practice, and how is this growth split between storage slots and contract bytecode?
3. State sensitivity: how sensitive are transaction execution results to evaluation on nearby historical states relative to the state at which they actually executed?

To answer these questions, we make the following tactical contributions:

1. *Trace-level decomposition.* We sample 3,000 blocks per day per chain and decompose each transaction into: (i) opcode-level execution gas, (ii) intrinsic gas, (iii) refunds, and (iv) persistent state deltas, including net storage slots written and net contract bytecode growth. This decomposition allows us to reason about execution as a multi-resource workload while remaining grounded in the EVM's native accounting.
2. *State sensitivity datasets.* We build two complementary datasets that quantify dependence on historical state. First, we probe the RPC surface by calling `eth_estimateGas` at multiple historical states and measuring how estimates and failure modes evolve as the estimation state drifts from the execution state. Second, using replay-based tracing, we re-execute transactions on multiple historical states to measure how opcode usage, storage access patterns, and total execution gas consumption change with state age.
3. *Labeling and stratification.* To connect low-level execution costs to higher-level behavior, we introduce transaction- and address-level labels and use them to stratify both workload composition and state sensitivity across application categories, including extractive, i.e., maximal extractable value (MEV), activity. State sensitivity is highly heterogeneous across transaction types.

1.2 Key Findings and Implications

Our measurements reveal that workload shifts between L1 and L2 have direct implications not only for resource pricing and system efficiency, but also for workload estimation and transaction predictability experienced by users. We summarize the key takeaways below.

1. *One-dimensional pricing does not adapt to workload shifts.* The workload differences we observe between L1 and L2 are substantial enough that the right relative opcode prices should differ across chains. The instability is not confined to cross-chain differences: Ethereum's gas limit doubling during 2025 shifted its resource profile measurably toward more compute-heavy, Base-like patterns without any repricing of opcodes. When the workload mix drifts, one-dimensional gas metering cannot adapt on its own. The only recourse is manual repricing, which has historically lagged behind workload changes.

In particular, Base L2 MEV is substantially different from Ethereum L1 MEV, being dominated by optimistic MEV [39] (speculative on-chain probing) and more state-sensitive in both the variety of slots accessed and the values it writes. These findings motivate multi-dimensional fee markets pricing distinct resources separately.

2. *State growth is substantial (435 GB on Base, 30 GB on Ethereum in 2025), compositionally diverse, and bursty.* Storage slots account for 61.4% of Base's growth and 50.9% of Ethereum's, while bytecode contributes 25.4% and 18.2%, respectively. Simple transfers account for 33.2% of Ethereum transactions and only 6.7% of gas but contribute 15.4% of state growth through persistent account creations. Daily state growth varies by up to an order of magnitude, and slot deletion rates are higher on Ethereum (34.6%) than on Base (22.5%), implying stronger incentives for storage reusability on Ethereum. Current gas pricing treats state writes as transient costs, failing to reflect their permanent nature; the true burden falls on every node.
3. *State sensitivity degrades transaction predictability and complicates execution design.* Gas estimates change across nearby historical states for 46.0% of transactions on Base and 13.9% on Ethereum, with MEV and DeFi activity exhibiting especially high sensitivity. Storage access patterns also diverge across execution states, limiting the effectiveness of access lists, complicating gas limit estimation for delayed execution designs, and increasing the cost of retries in optimistic parallel execution. This is already a user experience problem, and its consequences grow as chains adopt higher throughput, encrypted mempools, or delayed execution models.

2 Background

Transaction model and state machine replication. Blockchains follow the paradigm of state machine replication (SMR): at any given point, blockchain nodes maintain the state data structure; in Ethereum and EVM-compatible systems, this state consists of accounts and their associated data (balances, contract code, and contract storage). A distributed set of nodes then agree on an ordered log of transactions (consisting of an ordered chain of blocks), and each node deterministically applies every transaction in the order determined by the log to the previous maintained state (i.e., we say that a transaction is executed against a particular *pre-state*), in order to reach the new state (we call this the *post-state*) that will be used to apply the following transaction in the ordered log, and so forth.

In practice, transactions are constructed and signed at a time during which both pending transactions for inclusion at the following block might not be known, as well as the node or entity (such as a wallet) might have an incomplete view of the possible pre-state of the transaction. This can, e.g., happen in two key regimes: (1) when the chain is of high-throughput or low-latency and there are lots of pending transactions or the transaction under-submission might only be included after an unpredictable delay due to network factors, or (2) as an inherent property of the system in the case of *delayed execution*, i.e., an execution model in which consensus over transaction inclusion and ordering is reached before the corresponding state transitions have been computed, so that the last k blocks remain unexecuted at submission time; examples include Monad [31] and Avalanche's ACP-194 specification [34]. Delayed execution increases performance by removing execution from the critical path of consensus, and pipelining it instead. In any case, there is inherent uncertainty in the transaction's pre-state. As a result, the same transaction's execution can follow different control-flow paths, touch different storage keys, or fail when executed in the context of various possible pre-states. We say a transaction is *state-sensitive* if re-executing

it on a different pre-state produces different gas usage, different storage access patterns, or a different execution outcome (e.g., success vs. revert). The extent of this sensitivity is one of the key constituents under exploration in our work.

Ethereum accounts, storage, and the EVM. Ethereum maps 160-bit addresses to *accounts*. "Externally owned accounts" (EOAs in short) are controlled by cryptographic private keys, and are transaction originators. "Contract accounts" additionally store EVM bytecode (corresponding to the contract) and maintain persistent storage. Contract storage can be conceptually mapped to a key–value map from 256-bit keys to 256-bit values. Storage slots can be accessed via the `SLOAD` (read) and `SSTORE` (write) opcodes.

Persistent state is a long-term resource: it accumulates across blocks and directly affects the (ongoing) storage as well as the sync costs of operating both a validator and a full node. State can grow through the creation of new accounts (e.g., first-time recipients of value transfers), new contract deployments (e.g., bytecode and initialized storage slots), and contract storage writes.

The EVM is a stack-based virtual machine. Transactions can invoke smart contract functions directly or indirectly via call-family opcodes (e.g., `CALL`, `DELEGATECALL`), which create nested call frames and enable the composability of smart contract calls. In addition to EVM bytecode, Ethereum exposes a small set of special-purpose system contracts (called "precompiles") that implement, for instance, common cryptographic and arithmetic operations at fixed addresses.

Gas accounting and state-dependent costs. To guard against denial-of-service-style attacks, Ethereum charges transactions for each executed opcode in units of *gas*, as outlined in Section 1. A transaction sender (hence the transaction itself) must specify a *gas limit*, which caps the total gas that may be *used* by execution. If execution runs out of gas, all state changes are reverted and the sender is charged the full gas limit of the failing frame. If the transaction reverts explicitly (e.g., via `REVERT`), state changes are also undone, but only the gas consumed up to the revert point is charged; the remainder is returned to the sender. Under Ethereum's EIP-1559 fee mechanism [8], blocks are constrained by a gas target and a maximum gas limit. The total gas used by a block is the sum of gas used by its included transactions in canonical order and must not exceed the block gas limit. The protocol adjusts a per-block *base fee* in response to aggregate gas demand. Users specify additional fee parameters (including a priority fee to be tipped to the validator of the block), but the mapping from opcodes to gas costs is fixed by the protocol design.

Transaction gas can be further split into *intrinsic* and *execution* components. Intrinsic gas covers fixed per-transaction overheads, including the base cost, calldata charges, and (when present) up-front structural declarations including, for example, access lists [11] or authorization lists. On the other hand, execution gas corresponds to the gas charged as per the EVM opcodes executed. Ethereum also features gas refunds for certain operations (e.g., storage zeroing), but refunds are bounded and have been reduced over time [12] for reasons relating to the reduction of DoS vectors.

State-dependence in gas costs arises from two mostly-disjoint mechanisms. First, `SSTORE` gas depends on the current slot value and whether allocation / deallocation occurs (zero $\leftrightarrow$ non-zero transitions) or merely overwrite occurs (non-zero $\leftrightarrow$ non-zero transitions), as codified by EIP-2200 [41] and refund rules under EIP-3529 [12]. Second, while EIP-2929 [10] cold/warm charges are local to a single transaction – the first access within a transaction is always charged cold – the *set* of slots and accounts a transaction touches is itself state-

dependent, because control flow and data-structure layout can lead the same transaction to access different keys under different pre-states. Access lists [11] can pre-warm specified entries, but doing so effectively requires predicting this state-dependent access pattern in advance. In addition, the values stored in state can influence control flow, causing transactions to follow different execution paths depending on the state they encounter.

Layer-2 solutions, typical applications, and workload types. EVM-compatible L2 solutions, called rollups, preserve EVM execution semantics while changing the underlying execution regime. They typically offer much lower fees and higher throughput by executing transactions off-chain. Only minimally sufficient post-execution data is posted to the base layer (Ethereum L1 in this case) to substantiate the state transition. Base, the L2 we study in this work, is an optimistic rollup. Ethereum L1 has ∼12-second blocks, whereas Base has ∼2-second blocks with higher gas capacity per block.

The core activity of the EVM lies in its smart contracts, which are particularly prolific on high-throughput L2s. *Tokens* (e.g., of the ERC-20 standard) implement fungible balances maintained by accounts in persistent (contract) storage. Many *decentralized finance* (DeFi) applications, whose goal is to mirror functions of traditional finance – such as token exchange, lending, and liquidity provision – often give rise to highly gas- and storage-intensive contracts. *Maximal extractable value* (MEV) refers to profit opportunities obtained by specialized actors from the strategic ordering, inclusion, and exclusion of transactions; one common MEV strategy is exchange-rate arbitrage (for example between a *centralized* and a *decentralized* exchange, otherwise known as CEX-DEX arbitrage).

3 Related Work

Gas mispricing and the challenge of one-dimensional pricing. Prior work on gas pricing has focused on calibrating the relative pricing of opcodes to try to reflect the actual resource consumption within Ethereum L1. Perez and Livshits [32] demonstrated how mispriced gas costs enable denial-of-service attacks. Parallel opcode-benchmarking efforts have measured the runtime costs of individual EVM opcodes under synthetic inputs to assess whether gas prices track real resource consumption [1]; we complement this line of work by measuring the realized opcode mix in live 2025 workloads and how it drifts across chains and time. Ethereum has undergone repeated gas cost adjustments (e.g., EIP-1884 [40], EIP-2929 [10]).

However, even if these relative prices are well-calibrated, the one-dimensional model of metering leads to underutilization of some resources and overutilization of others when the workload mix changes [17, 2]. Our empirical findings in this work provide direct evidence that the mix of resource consumption varies both across time, and across different EVM ecosystems (L1 versus L2), thus motivating multidimensional fee markets. Beyond theoretical foundations, concrete proposals such as EIP-8011 [35] have begun to define multidimensional gas components for deployment, though they have not been grounded in an empirical study of live L1 and L2 workload mix drift. Furthermore, simply inheriting L1 gas prices for L2 chains leads to resource imbalances due to differing workload characteristics.

Recent concurrent work by Silva [37] performs an empirical analysis of gas metering on Ethereum, breaking down gas usage according to opcodes and quantifying potential gains from multidimensional fee markets. While this work shares similarities with our gas decomposition methodology, we extend the analysis to compare L1 and L2 workloads, analyze the breakdown as a function of the activity type (such as DeFi, MEV, ERC20, etc.), and explicitly measure the sensitivity of the gas used by various opcodes to the state at which the transaction ends up executing.

State growth and state access patterns. Silva [38] recently examined state growth scenarios and the impact of gas repricing on Ethereum, projecting future state sizes under different parameterizations. However, this analysis does not decompose state growth by transaction category or separately track storage slots versus contract bytecode. Our measurements quantify both dimensions and reveal that bytecode constitutes roughly 25% of total state growth on Base, whereas simple transfers dominate state growth on Ethereum. Ren et al. [33] analyze Ethereum workloads from a key-value storage perspective, focusing on storage access patterns and their implications for database design. Their work complements ours by examining storage system internals, whereas we focus on protocol-level resource consumption and its variation across execution contexts. The persistent growth of blockchain state has motivated various proposals for state expiry and history pruning [7, 6, 30, 29]. Our analysis highlights the current state of persistent state growth and shows, through transaction-level decomposition, how different application categories contribute to state bloat.

Predictability of state access pattern for parallel and delayed execution. Many recent works [24, 42, 4] have analyzed state access patterns across transactions to understand parallelizability of EVM workloads. These works, however, assume that the set of storage slots accessed by a transaction is deterministic and known before execution. Our state sensitivity measurements challenge this assumption: we show that the same transaction can create different amounts of I/O workloads and access different storage keys when executed on different historical states. This variability threatens one of the key redeeming qualities of optimistic parallel execution strategies in the form of caching, as speculative execution may bring different data into cache than what is eventually needed upon retry after a conflict.

Heimbach et al. [25] analyze EIP-2930 access lists and find that they are often populated incorrectly or suboptimally, resulting in missed gas savings or even gas penalties. Our work shows that optimal access lists are sensitive to the state at which the transaction ends up executing. Cabral et al. [13] investigate how gas usage and minimum gas limits vary across executions at different states, and the efficacy of different gas limit estimation algorithms as a function of the gap between estimation state and execution state. Our findings in this work are consistent with their results that the gas estimation quality degrades as the state keeps evolving. We however take the sensitivity analysis further and break it down according to the sensitivity of various opcodes, and transaction categories. While Cabral et al. discover that largely gas estimates remain valid for about 11 blocks, we show that it depends on transaction type and blockchain context: e.g., MEV transactions on Base are far more state-sensitive than other types.

4 Data Collection

Data is collected by running archival `reth` client Ethereum and Base nodes. While the analysis logic is client-agnostic, our data collection pipeline is implemented against `reth`-specific RPC interfaces,[2] as RPC call semantics and tracing implementations differ slightly across Ethereum client implementations. All code, scripts, queries, and the full opcode-to-category mapping used in this work are publicly available [22].

[2] `https://reth.rs/jsonrpc/trace`

4.1 Transaction Gas Cost Decomposition and State Growth

We now present the methodology for preparing our *long-term dataset*. For Ethereum and Base, we randomly sample 3,000 blocks per day for 2025. Sampling 3,000 blocks per day yields sufficient data for daily-granularity analysis. Note that processing all blocks, in combination with the multiple re-executions per transaction required for the state-sensitivity analysis, was computationally infeasible. This corresponds to sampling approximately 42% of daily blocks on Ethereum (out of roughly 7,200 blocks per day) and approximately 7% of daily blocks on Base (out of roughly 43,200 blocks per day). Blocks are sampled uniformly at random within each day and aggregate statistics such as state growth are extrapolated proportionally. While the sampling fraction is lower on Base, the absolute number of sampled transactions is comparable across chains (cf. Table 1a).

For each sampled block, we decompose gas usage into intrinsic and execution components. We then further break down both intrinsic gas and execution gas into their sub-components. Since intrinsic gas is not exposed as a structured breakdown by the RPC interface, we reconstruct its components using explicit accounting rules. Intrinsic gas comprises the minimum transaction gas (21,000), calldata costs, contract creation cost, access list gas (EIP-2930), and authorization list gas (EIP-7702). Execution gas is attributed per opcode. Each opcode is mapped to a single operation category (e.g., `SSTORE` to storage write, `SLOAD` to storage read); we do not decompose gas within an opcode. Gas refunds are tracked separately and capped according to EIP-3529. We additionally identify calls to EVM precompiles, including cryptographic and arithmetic precompiles, and account for protocol changes active during the measurement period, including the Pectra fork and EIP-7623 calldata pricing.

For the execution component, we trace all transactions and collect opcode-level execution data. Each execution of a call-family opcode (`CALL`, `CALLCODE`, `DELEGATECALL`, and `STATICCALL`) creates a new call frame. In RPC execution traces, the gas attributed to a frame includes the gas charged by opcodes executed in that frame as well as the gas consumed by all nested child frames spawned by subsequent call opcodes. The gas consumed by the call-family opcode itself is not exposed as a separate quantity.

To calculate the gas charged for call-type opcodes, we apply a depth-directed accounting model. For each call frame, which is created by a call-family opcode, we compute the gas attributed to that call as the total gas reported for the frame minus the gas consumed by its direct child frames and minus the gas consumed by other opcodes executed directly in the same frame. This yields the gas charged for the call opcode itself. We note that our depth-directed accounting attributes gas to the frame that consumes it; gas that is reserved or forwarded by the protocol but never consumed is not counted as a cost in any frame. We additionally track cold access patterns for account- and storage-level operations. This includes tracking account-related opcodes (e.g., `BALANCE`) and storage operations (`SLOAD`, `SSTORE`).

Finally, we quantify state growth over time. At the storage level, we track slot-level changes resulting from storage writes. We further record the creation of new accounts, as well as the deployment of contract code. These measurements allow us to track how storage and code footprint evolve over time.[3]

[3] Note that post-Cancun hard fork (EIP-6780), contract code removal is only possible for contracts created and destroyed within the same transaction.

4.2 State Sensitivity

To characterize how sensitive transaction behavior is to the blockchain state, we perform a state sensitivity analysis that evaluates execution results, gas costs, and storage access patterns under different historical states. The analysis covers all transactions included in the 3,000 blocks sampled per day during September 2025, a period of stable protocol parameters between the Pectra fork (May 2025) and subsequent gas limit adjustments. Due to the computational cost of re-executing every transaction at multiple historical states, the analysis is limited to a single month.

We compare two kinds of execution states. The *landed state* is the state under which a transaction actually executed on-chain, at its original position within block b. This serves as our ground-truth reference and is obtained from canonical execution traces via `debug_traceBlockByNumber`. The *lookback states* are historical pre-states at varying distance $N \geq 0$: for a transaction in block b, lookback N refers to the state at the end of block $b-1-N$, so lookback 0 is the state immediately preceding block b. Lookback states are obtained via `debug_traceCall`. Gas estimation is relatively inexpensive compared to full opcode-level re-execution, which allows us to use finer lookback granularity ($N \in \{0, \ldots, 20\}$) for gas estimates while limiting gas decomposition and storage access analysis to a sparser set ($N \in \{0, 5, 10, 20\}$) plus the landed state.

Gas estimate. For each sampled block in the analysis period, we process all included transactions and evaluate gas estimates through the `eth_estimateGas` API[4] at multiple historical states. Specifically, for a transaction included in block b, we request gas estimates at block heights $b-1-N$, with lookback distances $N \in \{0, 1, 2, \ldots, 20\}$.

Transaction gas cost decomposition. We analyze state sensitivity by re-executing transactions on historical blockchain states. For a transaction originally included in block b, we replay execution at the landed state and lookback distances $N \in \{0, 5, 10, 20\}$. Re-execution for $N \geq 0$ is performed using `debug_traceCall`, while the landed state uses canonical execution traces. Opcode-level gas usage is extracted as per the methodology of our long-term analysis (cf. Section 4.1). For each lookback level, we record execution success, total gas consumption, and per-opcode gas breakdowns.

Storage slot access patterns. We collect the set of storage slots accessed by transactions at multiple state lookback points, comprising the landed state and lookback distances $N \in \{0, 5, 10, 20\}$. At each lookback level, we identify which storage slots are read and which are written during execution.[5] We record both accessed slots and their values, enabling analysis of access-pattern and value divergence across historical states.

4.3 Address and Transactions Labels

We construct address- and transaction-level labels for Ethereum and Base using a multi-stage pipeline combining public labeling sources, on-chain measurements, and manual annotations. The pipeline is largely chain-agnostic, with chain-specific steps applied where required.

[4] `https://ethereum.org/developers/docs/apis/json-rpc/#eth_estimategas`

[5] Write sets are obtained from state diffs (`prestateTracer` in diff mode) and thus record *net-changed* slots, i.e., slots whose value after the transaction differs from their value before. Writes that restore a slot's original value, same-value writes, and writes reverted within the transaction are not captured.

We first retrieve address labels from Spellbook (Dune Analytics [18]) and from DefiLlama [16], which provides protocol metadata and categorical classifications across EVM-compatible chains. Next, we augment the label set with entries from the Kleros Curate registry [20]. Kleros Curate provides community-validated contract labels. These public sources cover 46.4% of Ethereum transactions but only 19.3% on Base, motivating the additional labeling steps described below, which raise coverage to 62.9% and 75.8%, respectively.

To identify protocol contracts that remain unlabeled , we query the respective chain RPCs to detect common liquidity pools and token contracts, including Uniswap V2/V3, Curve, Balancer, and Aerodrome deployments, as well as ERC20 tokens. This step is applied to all addresses with at least 100 transactions. We further enrich the labels using chain explorer APIs. For Ethereum, we query the Etherscan API, while for Base we query the Basescan API, targeting addresses with at least 1,000 transactions still unlabeled after earlier steps.

Next, we add chain-specific labels for MEV contracts. The labeling strategies differ between chains because the dominant MEV strategies differ: Ethereum L1 MEV is dominated by atomic arbitrage, sandwich, and CEX-DEX strategies, while on Base optimistic cyclic arbitrage dominates [39]. For Ethereum, we incorporate the top 1,000 addresses by gas usage, labeled as atomic arbitrage and CEX-DEX MEV contracts, obtained through Dune Analytics queries adapted from prior work [44, 26, 28, 27]. For Base, we add cyclic arbitrage MEV labels for the top 1,000 addresses by gas usage obtained through a Dune Analytics query adapted from prior work [39]. For Base only, we further analyze unlabeled contracts with at least 10,000 transactions to assess whether they exhibit spam or optimistic MEV behavior. Optimistic MEV, i.e., speculative on-chain probing that frequently results in no state change, is a phenomenon specific to low-fee chains such as Base and is not prevalent on Ethereum L1 [39]. Thus, we apply this labeling step only to Base. For each selected contract, we sample up to 100 transactions and examine whether they induce any state changes beyond gas accounting. We compute the proportion of sampled transactions that do not result in any state change other than gas accounting. Contracts for which more than half of the sampled transactions exhibit no state change are labeled as optimistic MEV. These MEV labels are applied only to addresses that remain otherwise unlabeled.

Finally, we manually label a small number of addresses that remain unlabeled after all automated steps. These labels are assigned in cases where external context is available, for example when a contract is deployed by a known deployer.

After label assignment, we map all address labels to a set of coarse-grained functional categories: *MEV*, *DeFi*, *Token*, *Infrastructure*,[6] *Bridge*, *NFT*, *CEX*, and *L2*.

In addition to address-level labels, we annotate transactions directly in the database. *Simple ETH Transfers* are identified by a gas usage of exactly 21,000, excluding EIP-4844 blob transactions, which are labeled as *L2*. *Contract Creation* transactions are identified by a `NULL` receiver, that is, top-level deployments only. Transactions that merely execute the `CREATE` or `CREATE2` opcode, such as calls to factory contracts, retain their own category.

These transaction-level annotations take precedence over address-level labels: a transaction meeting the *Simple ETH Transfer* criterion is counted as such even when its receiver carries a functional label.

[6] In the Infrastructure category, we group activity that supports the operation of on-chain protocols, such as price oracle updates and account abstraction transactions (e.g., smart contract wallet execution and paymaster-sponsored gas).

Table 1 Gas usage comparison between Ethereum and Base in 2025. Table 1a reports per-transaction gas statistics; access-list and authorization-list gas (EIP-7702) are included in intrinsic gas. Table 1b reports the execution gas breakdown across nine EVM operation categories.

(a) Per-Transaction Gas Statistics.

Metric	Ethereum	Base
Total Transactions	219,811,453	256,506,972
Mean Gas per Tx:		
Total Gas	103,284	232,936
Execution	82,441	217,340
Intrinsic	26,656	25,654
Access List	2,305	165
Auth List	290	144
Refund	6,770	10,735
Execution % of Total	79.82%	93.30%

(b) Execution Gas Breakdown.

Category	Ethereum	Base
Storage Read	22.73%	29.23%
Storage Write	34.88%	25.54%
Compute	20.21%	24.30%
Calls	10.18%	12.57%
Contract Ops	4.47%	4.78%
Logging	7.19%	3.45%
Transient Storage	0.15%	0.09%
Epilogue	0.20%	0.04%
Precompiles	0.00%	0.00%

5 Execution and State Characteristics

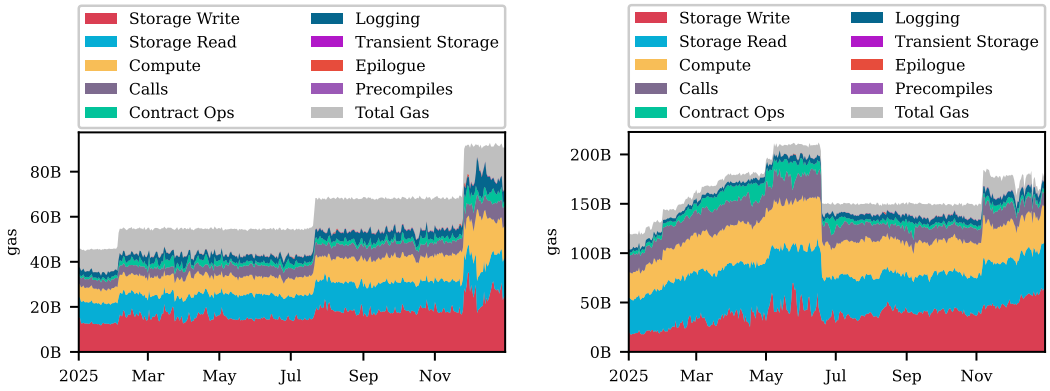
Our long-term dataset, built by sampling 3,000 blocks per day on both Ethereum and Base throughout 2025, comprises 219,811,453 Ethereum transactions and 256,506,972 Base transactions (cf. Table 1a). At the transaction level, average gas usage is significantly higher on Base (232,936 gas per transaction) than on Ethereum (103,284 gas).

5.1 Gas Decomposition

We commence the analysis by examining the execution gas decomposition in more detail (cf. Table 1a). Notice that the execution component accounts for a substantially larger share of total gas usage on Base, comprising 93.30% of total gas, compared to 79.82% on Ethereum. This gap reflects both the larger share of simple value transfers on Ethereum, which incur only intrinsic gas (cf. Section 5.4), and the fact that the average transaction on Base uses more gas.

Breaking execution gas down by operation category (Table 1b), we find that Base exhibits a more storage read-intensive workload than Ethereum, with storage reads accounting for 29.23% of execution gas on Base compared to 22.73% on Ethereum. In contrast, Ethereum spends a larger fraction of execution gas on storage writes. Base also allocates a higher share of execution gas to computation and call-related operations. Notably, a large fraction of call-related gas on Base is attributable to `STATICCALL`, which accounts for 66.8% of call gas. In contrast, `STATICCALL` constitutes only 22.4% of call-related gas on Ethereum. This indicates that cross-contract interactions on Base are predominantly read-only, reflecting a composability pattern centered on state inspection.

We further examine the temporal evolution of execution gas composition in Figure 1 for Ethereum and Base. On Base, we observe that while the workload is initially dominated by storage reads, it gradually shifts toward a higher share of storage writes over time. This shift may be related to changes in MEV strategies on Base. In particular, optimistic MEV dominated gas usage in early 2025 and is characterized by extensive state probing via read-only interactions [39]. Over time, we observe a reduction in such probing behavior and a corresponding increase in state-modifying execution (cf. Section 5.4), though other factors such as ecosystem maturation and gas limit increases during 2025 may also contribute to this shift. Other components such as logging remain comparatively stable.



(a) Ethereum. **(b)** Base.

Figure 1 Daily gas usage breakdown by EVM operation category for (1a) Ethereum and (1b) Base. For Ethereum, we sample 3,000 blocks per day from approximately 7,200 daily blocks. For Base, there are 43,200 blocks per day, we likewise sample 3,000 blocks per day. Compared to Ethereum, Base shows a higher proportion of gas spent on storage reads and a lower share of non-execution overhead (shown in grey).

Ethereum's execution mix also drifts measurably over the course of 2025: in January, storage writes accounted for 35.6% of Ethereum's execution gas and compute for 18.0%. By December, writes had eased to 32.5% while compute rose to 23.8%, even as total execution gas roughly doubled. This drift coincides with Ethereum's gas limit doubling from 30M to 60M during 2025, which increased capacity without repricing opcodes. The shift is in the direction of Base's more compute-heavy profile, although Ethereum remains more storage-write-intensive, suggesting that the capacity and cost regime a chain operates under shapes its workload composition.

5.2 Storage Access Patterns

To further characterize workload differences between Ethereum and Base, we examine storage access patterns by focusing on the `SLOAD` opcode, which reads values from contract storage. For each transaction, we record the total number of `SLOAD` operations and classify each access as cold or warm, depending on whether the corresponding storage slot is accessed for the first time in the transaction or has been previously accessed or pre-warmed via an access list.[7] This distinction is important because warm storage accesses are charged significantly less gas than cold accesses. Under EIP-2929 [10], the first `SLOAD` to a given storage slot in a transaction incurs a cold access cost of approximately 2,100 gas, whereas subsequent warm accesses to the same slot cost only about 100 gas.

Table 2 summarizes the results. Base transactions access more distinct storage slots than Ethereum, indicating lower key reuse: 49.7% of `SLOAD` operations on Base are cold, vs. 39.6% on Ethereum.

Base transactions also execute substantially more `SLOAD`s on average than Ethereum, with a heavier tail and a P25 of zero on Ethereum reflecting many transactions with no storage interaction (cf. Table 2).

[7] An access list is a list of addresses and specific storage slots that a transaction declares in advance it intends to access.

Table 2 `SLOAD` access pattern comparison between Base and Ethereum, showing total transactions, cold versus warm storage reads, and per-transaction `SLOAD` access statistics.

Chain	Total Txs	Cold	Warm	Accesses/Tx					
				Mean	Std	P25	P50	P75	P95
Ethereum	219,811,453	1,828,127,520 (39.6%)	2,791,584,524 (60.4%)	21.0	177.3	0.0	5.0	15.0	75.0
Base	256,506,972	7,401,539,024 (49.7%)	7,501,080,368 (50.3%)	58.1	261.1	3.0	14.0	49.0	243.0

Cold storage accesses are generally considered underpriced relative to other opcodes, even after the adjustments introduced through EIP-2929. Recent protocol designs and proposals, including newer EVM variants such as Monad [14] and ongoing discussions within the Ethereum Foundation [36], have introduced or considered higher costs for cold storage reads. In this context, the higher fraction of cold `SLOAD` operations and the larger number of distinct storage accesses observed on Base indicate that a larger share of the workload is concentrated in components that are comparatively underpriced.

5.3 State Growth

Related to cold storage accesses, state growth is commonly cited as a key constraint for scaling blockchain systems. We therefore continue with an analysis of state growth. Table 3 reports estimated yearly state growth on Ethereum and Base, with figures extrapolated from our sampled dataset. We extrapolate by dividing sampled counts by the per-chain sampling fraction (42% for Ethereum, 6.9% for Base). Thus, cross-chain ratios compare extrapolated full-year totals, not raw sampled counts.

We decompose state growth into three components: newly added storage slots, deployed smart contract bytecode, and account state. Storage growth from slots is computed assuming 64 bytes (i.e., 32 bytes for the key and value respectively). Account state includes both EOAs and contract accounts and consists of four 32-byte fields: nonce, balance, code hash, and storage root. This results in 128 bytes per account. Note that the exact byte footprint of account state may vary across client implementations due to differences in data representation. Our approximation provides a consistent basis for comparison.

In 2025, estimated state growth differs markedly between Base and Ethereum. Based on extrapolation from our sampled data, total state growth amounts to 435.22 GB on Base, compared to 29.92 GB on Ethereum, corresponding to a 14.55× difference. A gap in this direction is expected given Base's ∼6× higher block rate, substantially lower fees, and greater share of storage-write-heavy activity. The observed magnitude, however, exceeds a naive throughput-based extrapolation, indicating that workload composition, not just transaction volume, drives the difference.

The composition of this growth also differs substantially. On Base, newly added storage slots account for 61.4% of total state growth, compared to 50.9% on Ethereum, while deployed contract bytecode constitutes 25.4% on Base and 18.2% on Ethereum. Account state represents 30.9% of total state growth on Ethereum, compared to 13.1% on Base.

[8] Under EIP-161 [43], an account is dead if it is non-existent or empty, i.e., it has no code, zero nonce, and zero balance. We count dead-to-live and live-to-dead transitions at transaction boundaries, and net account growth is the difference between the two counts. A live-to-dead transition occurs when a prefunded address with zero nonce and no code receives a `CREATE2` deployment, which EIP-684 [9] permits because balance is not a collision criterion, and the resulting contract executes `SELFDESTRUCT` in the same transaction, for which EIP-6780 [3] still deletes the account. The prefunding is what makes the account live before the transaction, so the deletion is visible at a transaction boundary.

Table 3 State storage growth on Ethereum and Base in 2025, extrapolated from the sampled blocks. Table 3a reports storage-slot creations, deletions, and updates; net slot growth; bytecode allocation and removal; and account creations and deletions.[8] Table 3b converts net growth to logical GB using 64 bytes per storage slot and 128 bytes per account, reports per-transaction averages over all transactions, and gives Base/Ethereum ratios.

(a) State Changes.

Metric	Ethereum	Base
Storage Slots:		
Created	364,008,598	5,388,214,765
Deleted	125,870,269	1,209,883,109
Updated	1,254,823,628	12,699,968,280
Net Written	238,138,180	4,178,331,508
Contract Bytecode (GB):		
Allocated	5.44	110.60
Freed	0.00	0.00
Net Growth	5.44	110.60
Accounts:		
Created	72,230,074	446,944,680
Deleted	33,653	4,905
Net Growth	72,196,421	446,939,775

(b) Storage Totals & Ratios.

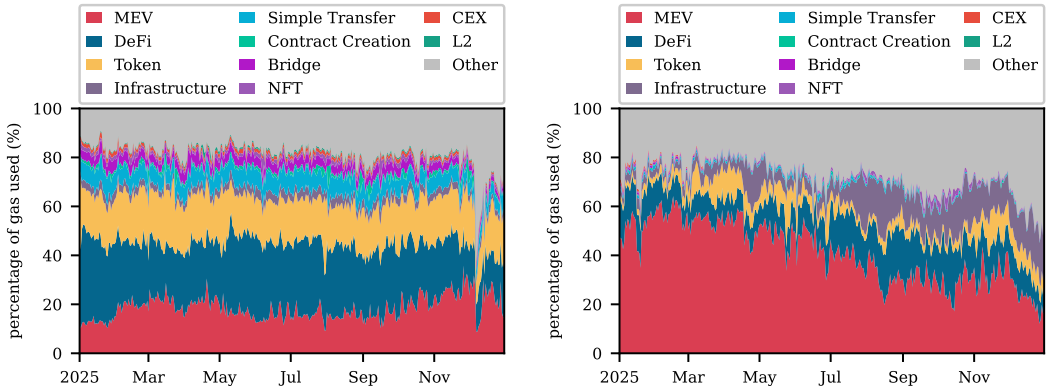
Metric	Ethereum	Base
Total Storage Growth (GB):		
Storage Slots	15.24	267.41
Bytecode	5.44	110.60
Account State	9.24	57.21
Total	**29.92**	**435.22**
Per-Transaction Averages:		
Slots/TX	0.45	1.13
Bytecode/TX (bytes)	10.3	29.9
Ratios (Base/Ethereum):		
Net Slots		17.55x
Bytecode Growth		20.33x
Net Accounts		6.19x
Total Storage		14.55x

Account state growth also differs vastly between the two chains. On a per-transaction basis, Base transactions induce significantly more persistent state, with an average net growth of 1.13 storage slots per transaction compared to 0.45 on Ethereum, and 29.9 bytes of net bytecode growth per transaction compared to 10.3 bytes on Ethereum. This is reflective of the difference in dollar cost between the two chains, along with the usage patterns and maturity of the ecosystem. A larger fraction of newly created storage slots is subsequently deleted on Ethereum (approximately 35%) than on Base (approximately 22%), indicating higher slot turnover and stronger incentives for storage reuse on Ethereum.

To illustrate how individual contracts contribute to state growth, we examine the address `0x9ec...2b7`. Within our sampled blocks, this address is involved in 86,046 transactions and contributes 336,378,240 bytes of deployed contract bytecode, 37,315,630 net added storage slots (37,534,300 created), and 7,475,172 account births. In aggregate, this corresponds to an estimated 3.681 GB of additional state in our sampled blocks, which extrapolates to 53.01 GB of yearly storage added. This address acts as a deployer for the XEN token, which is known to generate substantial on-chain storage usage [19], and serves as an example of how a single contract can account for a non-negligible share of observed state growth. This single contract represents 12.18% of storage growth while only making up 0.03% of transactions.

In addition to storage being considered one of the main bottlenecks for blockchain scalability [21], the EVM collapses a high-dimensional resource-consuming workload into a single scalar unit of metering (gas). As a result, long-term state growth is priced indirectly through short-term, per-block execution limits, creating unnecessary fluctuations in the price of state growth. To examine the implications of this design choice, we analyze the distribution of daily state growth on Ethereum and Base.

Daily storage growth varies substantially over time on both chains (cf. Appendix A of the full version [23]). For example, on Base, daily storage growth ranges from 354.5 MB at the 1st percentile to 3,075.3 MB at the 99th percentile. This variance is driven by bursty, ecosystem-specific events, with concentration from a small number of contracts often dominating daily state totals.



(a) Ethereum. **(b)** Base.

Figure 2 Daily share of total gas usage by transaction category on (2a) Ethereum and (2b) Base over 2025. Gas usage is normalized per day and aggregated by transaction category. The anomaly in the share of gas used visible on Ethereum in December 2025 coincides with a `Prysm` client issue immediately following the Fusaka hard fork.

The large variance in daily state growth highlights a fundamental misalignment between short-term, transient execution-based gas pricing and the long-term, cumulative costs of persistent storage. Per-block gas limits can only regulate instantaneous resource usage, forcing a trade-off between restrictive limits that cause overall underutilization of chain and permissive limits that allow bursts of activity to translate into excessive and persistent state growth. As a result, volatile workloads are either constrained in the short term or, in the worst case, allowed to cause excessive long-term storage growth under the current gas pricing.

5.4 Behavior by Address Category

Next, we investigate differences in workload composition by transaction category, considering both gas usage and storage growth. We focus on the main categories observed on both chains: MEV, DeFi applications such as DEXs and lending protocols, ERC-20 token interactions, infrastructure services such as oracles or account abstraction, and simple transfers. On Base, transaction activity is also more concentrated across a smaller set of receiver addresses, reflecting a more clustered workload structure (cf. Appendix B of the full version [23]). Figure 2 shows the evolution of gas usage by category over time, while Table 4 summarizes transaction counts, average gas usage, and average storage growth per transaction.

MEV constitutes a particularly large share of activity on Base, accounting for 41.2% of total gas usage and 43.6% of transactions. Although its share decreases toward the end of the year (Figure 2b), MEV remains the dominant gas-consuming category. In contrast, MEV accounts for only 18.8% of total gas usage on Ethereum. Notice that towards the end of the year, the "Other" category grows on Base, suggesting that some emerging or evolving MEV patterns may not be fully captured by our conservative labeling (i.e., only labeling addresses with a significant number of transactions and gas usage). On Ethereum, ERC-20 token transactions account for 29.1% of transactions and 17.9% of total gas usage, whereas on Base they account for only 7.9% of transactions and 6.0% of gas usage. DeFi activity contributes a similar share of gas usage on both chains, although DeFi transactions on Base consume more gas per transaction on average.

Table 4 Gas usage and storage growth by address category on Ethereum and Base in 2025. For each chain, shows transaction count and percentage, average gas per transaction and percentage of total gas, average storage per transaction (KB) and percentage of total storage. "Other" includes unlabeled transactions and categories with less than 0.1% gas usage on both chains.

Category	Ethereum						Base					
	Transactions		Gas Used		Storage Growth		Txs		Gas Used		Storage Growth	
	#	Pct	Avg	Pct	Avg (KB)	Pct	#	Pct	Avg	Pct	Avg (KB)	Pct
MEV	14,458,276	6.6	295,627	18.8	0.17	20.1	111,896,672	43.6	220,110	41.2	0.04	14.8
DeFi	31,854,577	14.5	180,632	25.3	0.03	7.7	33,704,931	13.1	228,276	12.9	0.06	6.3
Token	63,950,730	29.1	63,563	17.9	0.03	14.6	20,354,249	7.9	174,677	6.0	0.22	15.0
Infrastructure	3,382,961	1.5	210,106	3.1	0.07	2.0	7,368,706	2.9	792,576	9.8	1.05	25.6
Simple Transfer	72,928,518	33.2	21,000	6.7	0.03	15.4	20,152,658	7.9	21,000	0.7	0.02	1.2
Bridge	5,769,963	2.6	138,939	3.5	0.04	1.7	1,779,509	0.7	140,766	0.4	0.02	0.1
NFT	2,363,354	1.1	140,973	1.5	0.08	1.5	2,770,415	1.1	134,890	0.6	0.08	0.7
Contract Creation	215,137	0.1	1,670,843	1.6	5.52	9.5	214,308	0.1	1,128,603	0.4	4.65	3.3
CEX	1,760,373	0.8	169,362	1.3	0.18	2.6	256,270	0.1	265,441	0.1	0.05	0.0
L2	1,525,605	0.7	74,233	0.5	0.03	0.3	0	0.0	-	0.0	-	0.0
Other	21,601,959	9.8	206,551	19.7	0.14	24.4	58,009,254	22.6	287,422	27.9	0.17	32.9

Appendix C of the full version [23] provides a detailed gas decomposition by category. For MEV transactions, the breakdown shows a more storage read-intensive and less storage write-intensive profile on Base than on Ethereum, consistent with optimistic MEV that reads on-chain prices during execution. Infrastructure transactions on Base, by contrast, exhibit a strongly storage write-heavy profile. Overall, the decomposition reveals systematic differences in workload composition across categories and chains.

Storage growth further differentiates transaction categories. Contract creation dominates per-transaction storage growth on both chains, which is not surprising given that transactions in this category deploy new contracts. Infrastructure transactions on Base also contribute disproportionately to overall storage growth relative to their share of transactions. In contrast, simple transfers account for a large fraction of transactions on Ethereum (33.2%), while contributing little to total gas usage (6.7%) yet 15.4% of state growth. This pattern indicates a high incidence of simple transfers to new addresses on Ethereum, increasing account-state growth. Notably, this form of state expansion is not directly reflected in transaction fees, as simple transfers incur only the minimum intrinsic gas cost of 21,000 gas.

6 State Sensitivity Analysis

In addition to analyzing how transactions behave in the specific block and position in which they are included, we also study how their behavior varies as a function of the execution state. This perspective is particularly relevant for user experience on high-throughput chains, where multiple blocks may be processed between transaction creation by the user and execution, allowing the underlying state to change. It is likewise important in blockchains with delayed execution [31, 34], where the execution of preceding blocks may not have completed at submission time and the exact state against which a transaction executes is not yet known.

To capture these effects, we execute each transaction we sampled in September 2025 against historical states at varying lookback distances N, emulating the view of the user creating the transaction. Specifically, a transaction included in block b is re-executed on the state corresponding to the end of block $b - 1 - N$.

Table 5 Gas estimate variance across state lookback distances for September 2025. CV denotes the coefficient of variation (standard deviation divided by mean) of gas estimates across lookbacks. Max Successful Simulation Lookback reports the maximum lookback distance (in blocks), up to 20, at which transaction simulation succeeds without error.

			CV (%)						Max Successful Simulation Lookback				
Chain	Txs	Non-Zero (%)	Mean	P50	P75	P90	P95	P99	P1	P5	P10	P25	P50
Ethereum	19,262,259	13.9	0.57	0.00	0.00	0.41	2.79	14.84	0	1	3	15	20
Base	22,420,978	46.0	6.88	0.00	0.91	15.65	40.64	139.74	0	4	15	20	20

6.1 Gas Estimation

We begin by analyzing the values returned by the gas estimation API, `eth_estimateGas`, which estimates the minimum gas required for a transaction's successful execution, across a range of state lookback distances. For each transaction, we simulate execution on historical states up to a maximum lookback of 20 blocks. If execution fails at a given lookback distance x, we do not continue simulations further into the past. For transactions that execute successfully across multiple lookbacks, we record (i) the coefficient of variation (CV) of the gas estimate, and (ii) the maximum lookback distance at which execution succeeds. We define the CV of a transaction's gas estimate as $\text{CV} = \sigma/\mu$, where μ and σ are the mean and standard deviation of the gas estimates obtained across successful lookbacks. A CV of zero indicates that the gas estimate is identical across all lookback distances, while a higher CV indicates greater sensitivity to execution state. The reported transaction counts and non-zero fractions cover all transactions with at least one successful estimate. The CV itself is defined only for transactions with at least two successful estimates, which make up 95.1% of the population on Ethereum and 98.8% on Base.

On Ethereum, only 13.9% of the full population have observed non-zero gas estimate variance across lookbacks, compared to 46.0% on Base (cf. Table 5). Consistent with this, the mean coefficient of variation (CV) of gas estimates is substantially higher on Base than on Ethereum. This difference persists across the distribution, as reflected in consistently higher percentile values (e.g., 14.84% at the 99th percentile on Ethereum versus 139.74% on Base). Overall, this indicates greater sensitivity of gas requirements to execution state on Base than on Ethereum. This difference likely reflects higher per-block transaction throughput and thereby more frequent state updates per block on Base along with a different workload composition. We note that our lookback windows are defined in blocks rather than wall-clock time. Since Ethereum produces blocks every ∼12 seconds and Base every ∼2 seconds, 20 blocks corresponds to ∼240 seconds on Ethereum but only ∼40 seconds on Base. The closest comparable wall-clock interval on Ethereum is a lookback of 5 blocks (∼60 seconds), which still exceeds the Base window. Even under this more conservative comparison, the sensitivity gap remains substantial, indicating that Base experiences genuinely higher state churn once block-time differences are accounted for.

The distribution of maximum successful simulation lookback distances, on the other hand, shows that simulations on Base tend to remain successful further into the past than on Ethereum. In particular, the 25th percentile is 20 blocks on Base, compared to 15 blocks on Ethereum. This is not a pure measure of code executability: because simulations retain the original transaction fee fields, a transaction can be rejected when its fee cap is below the historical base fee. The cross-chain difference may therefore reflect block times and fee dynamics as well as state-dependent execution.[9]

[9] The minimum fee a transaction needs to pay to be included in a particular block.

Table 6 Gas variance by opcode group across state lookback distances. The table reports the mean coefficient of variation (CV%) and the fraction of transactions with non-zero (NZ) variance. The overall columns refer to total transaction gas. The opcode-group columns refer to each group's share of total transaction gas, capturing shifts in gas composition across execution states.

	Overall	S-Read	S-Write	Trans	Compute	Calls	Contract	Log
Ethereum CV%	0.64	0.56	1.55	0.08	0.63	0.60	0.50	1.03
Ethereum % NZ	9.5	9.4	9.5	2.3	9.5	8.8	7.9	9.4
Base CV%	5.22	2.82	6.76	1.64	3.20	3.70	7.71	4.40
Base % NZ	42.6	42.5	20.1	5.7	42.6	42.1	20.7	18.1

We further examine gas estimate variance by transaction category (cf. Appendix D of the full version [23]). MEV and DeFi transactions affect the largest share of transactions with non-zero gas estimate variance on both chains (54.2% and 53.0% on Base), and MEV also carries the largest CV magnitude there (mean CV 10.11%). Contract Creation is instead among the least variable categories on Base (0.0% non-zero variance), since a deployment's gas cost is dominated by its own init code rather than by the state it executes against. Notably, the maximum successful lookback for MEV transactions on Base is substantially larger than on Ethereum (5th percentile at 20 blocks on Base versus 0 on Ethereum), likely reflecting the contrast between traditional MEV strategies on L1 and optimistic MEV strategies on L2s, which are by design crafted to operate under unknown state.

At the low-variance end, simple transfers exhibit greater state sensitivity on Ethereum (3.0% non-zero variance) than on Base (0.0%). This variance arises from EIP-7702, which allows EOAs to temporarily delegate execution to contract code, causing the same transfer to execute differently depending on whether delegation was active at a given historical state.

Overall, DeFi and MEV transactions account for the largest share of transactions whose gas requirements vary with the underlying execution state, while variability on Ethereum is also driven by time-related effects.

6.2 Gas Decomposition

Next, we analyze gas decomposition across state lookbacks to identify which gas components exhibit the highest variability, including storage operations, logging, and computation, as well as variability in total transaction gas. This analysis includes transactions with at least two successful executions across the considered lookback distances, and the variability metrics are computed over the successful executions of each transaction. Transactions that succeed at fewer than two lookbacks are excluded, and failed executions do not contribute to the reported variability. Most included transactions succeed at all four lookbacks (91.1% on Base and 85.6% on Ethereum), with the remainder contributing two or three observations. We evaluate execution at lookback distances of 0, 5, 10, and 20 blocks.

Table 6 shows that gas variability is higher on Base than on Ethereum, with overall gas CV roughly 8× higher on Base (5.22% versus 0.64%) and larger shifts in the composition of gas across opcode groups.[10]

[10] This analysis uses four replay states, whereas gas estimation uses 21, and therefore has fewer opportunities to detect variation. Because variability is computed only across successful re-executions, the reported CVs and non-zero shares exclude state changes that cause replay failure and should be interpreted as conservative estimates.

On Ethereum, the compositional variability is concentrated in storage writes and contract-related operations. On Base, it spans storage reads, writes, compute, and calls (cf. Table 6).

The per-category breakdown (cf. Appendix E of the full version [23]) reveals that MEV transactions on Base exhibit the highest variability of any Base category, with the mean CV of the opcode group's gas share reaching 11.45% for contract-related opcodes, 9.40% for storage writes, and 5.55% for logging. Base *Infrastructure* follows a similar pattern at a slightly lower level (9.11% and 8.21% respectively), while DeFi is more moderate (up to 4.33%). On Ethereum most categories are considerably less variable, but *Infrastructure* is the exception and carries the single largest value in the table, a 15.91% mean share CV for storage writes. Variability is thus concentrated in the categories whose gas composition depends most on the state they encounter.

6.3 Storage Slot Access Patterns

Finally, we analyze variance in storage slot accesses across state lookbacks. Specifically, we examine whether the set of storage slots accessed by a transaction changes as a function of the execution state. High variability in accessed slots has important implications. First, it limits the effectiveness of access lists, which require transactions to predeclare accessed storage slots in order to receive a gas discount. If accessed slots vary across execution states, access lists cannot be populated reliably. Second, many optimistic parallel execution approaches execute transactions speculatively in parallel and roll back upon conflicts, assuming that re-execution is faster due to cached state. This is no longer the case if retrying execution on a different state results in accesses to uncached storage slots.

We quantify storage slot overlap between two lookback distances N_1 and N_2 using three per-transaction metrics based on the Jaccard index. Let R_i and W_i denote the sets of storage slots read and written, respectively, at lookback N_i. *Read overlap* is the Jaccard similarity of the readsets, i.e., $|R_1 \cap R_2| \,/\, |R_1 \cup R_2|$. *Write overlap* is the Jaccard similarity of the writesets, i.e., $|W_1 \cap W_2| \,/\, |W_1 \cup W_2|$. *Value consistency* is the fraction of slots in the write intersection whose written values are identical, i.e., $|\{s \in W_1 \cap W_2 : v_1(s) = v_2(s)\}| \,/\, |W_1 \cap W_2|$. Each metric is computed over the transactions for which it is defined, so the three populations differ. Read overlap requires non-empty readsets at both lookbacks, which also excludes transactions whose re-execution produced no trace at either lookback. Write overlap additionally requires a non-empty write union. A transaction whose writeset vanishes entirely at one lookback scores zero, while transactions with empty writesets at both lookbacks are excluded. Value consistency requires an overlapping write. On Base at the (landed, $N = 0$) pair, these populations comprise 23.3M, 10.1M, and 8.6M transactions, respectively. We report the mean and median of each metric across its eligible transactions.

Appendix F of the full version [23] reports the distribution of the number of storage slots accessed per transaction at each lookback distance. We find that Base transactions access substantially more slots on average (22.0 at the landed state) than Ethereum (14.7), with high variability. Overlap across lookbacks is also limited, particularly on Base.

Next, we analyze the mean and median overlap of storage slot accesses across execution states for Ethereum and Base. We consider overlap separately for storage slots that are read, slots that are written, and, conditional on overlapping writes, whether the values written are identical. Table 7 reports these overlaps for different pairs of lookback distances. We note that the (landed, $N = 0$) comparison is particularly significant: it represents the minimum realistic gap between simulation and execution, capturing only the state changes introduced by transactions preceding the given one within the same block. Even under this best-case scenario, the divergence on Base is clearly visible, and it is concentrated in the *write* sets rather than the read sets.

Table 7 Storage slot overlap across state lookback distances N. Read overlap denotes the Jaccard similarity of the read-sets at the two states, and write overlap is defined analogously for the write-sets. Value overlap denotes the fraction of overlapping written slots whose values are identical. Each metric is averaged over the transactions for which it is defined. The write and value populations are subsets of the read population.

Lookback Pair	Ethereum						Base					
	Read (%)		Write (%)		Value (%)		Read (%)		Write (%)		Value (%)	
	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median
Landed vs 0	97.1	100	94.6	100	78.5	100	93.5	100	80.8	100	67.8	80
Landed vs 5	92.2	100	83.5	100	65.3	83	87.9	100	66.5	100	50.6	50
Landed vs 10	90.8	100	79.7	100	62.4	67	86.4	100	63.0	98	46.7	50
Landed vs 20	89.3	100	75.6	100	59.6	50	84.9	100	59.3	86	43.1	36
0 vs 5	94.3	100	86.7	100	68.7	100	90.0	100	72.6	100	52.0	50
0 vs 10	92.8	100	82.8	100	64.6	75	88.1	100	68.3	100	47.2	50
0 vs 20	91.4	100	78.5	100	61.1	57	86.2	100	63.8	96	43.1	35
5 vs 10	97.5	100	93.3	100	71.5	100	93.1	100	81.5	100	54.4	57
5 vs 20	96.0	100	88.3	100	65.2	75	90.4	100	74.5	100	46.7	50
10 vs 20	97.4	100	92.3	100	68.3	100	92.3	100	79.8	100	49.9	50

In line with our previous analysis, transactions on Base exhibit greater state sensitivity than those on Ethereum. We compare the landed state to the start-of-block lookback state ($N = 0$), which captures the effect of state modifications by preceding transactions within the same block. Read sets remain largely stable on both chains. The mean read overlap is 97.1% on Ethereum and 93.5% on Base, with a median of 100% on both, meaning the typical transaction reads exactly the same slots. Write sets diverge far more. The mean write overlap is 94.6% on Ethereum but only 80.8% on Base, and the gap widens at larger lookback distances. At $N = 20$, mean write overlap falls to 75.6% on Ethereum and 59.3% on Base.

Most notably, even when transactions write to the same storage slots, the values written do not necessarily coincide. On Ethereum at the (landed, $N = 0$) pair, transactions mostly write to the same slots (mean write overlap 94.6%, median 100%). However, the written values often differ: mean value overlap is only 78.5% (median 100%), indicating that a non-trivial tail of transactions updates the same slot with different values. On Base this tail effect is stronger: mean value overlap of 67.8%, with a median of 80%. State sensitivity thus manifests at two distinct levels. Value divergence, i.e., writing different values to the same slots, reflects the expected dependency of computation on mutable state such as pool balances or oracle prices. Slot divergence, i.e., writing to a different set of storage locations, is more consequential, as it indicates that the transaction's execution path itself has changed under the altered pre-state. On Base, where write overlap at lookback 0 is already only 80.8%, both forms of divergence are prevalent even within a single block.

One class of transactions for which divergent execution outcomes are expected are DEXes within DeFi. Because execution outcomes depend on mutable on-chain state, such as pool balances and prices, even small differences in the execution state can lead to different results. Consistent with this, even at lookback 0, MEV transactions on Base already exhibit write overlap of only 45.3% and value overlap of 57.6%, compared to 81.0% and 65.0% on Ethereum. At lookback 5, these drop further to 30.8% and 38.0% on Base versus 59.2% and 35.9% on Ethereum. DeFi transactions follow a similar pattern, with value overlap falling from 63.8% at lookback 0 to 43.3% at lookback 5 on Base, compared to 68.1% and 51.8% on Ethereum. The full category-level breakdown is reported in Appendix G of the full version [23].

State drift grows monotonically with lookback distance. On Base, mean value overlap falls from 67.8% at lookback 0 to 43.1% at lookback 20; on Ethereum, from 78.5% to 59.6%. Two factors contribute: elapsed wall-clock time and the number of state-modifying transactions

in the intervening blocks. Our data does not separately quantify these contributions, but the lookback-10 and lookback-20 measurements preview the simulation–execution gap that higher-throughput chains will increasingly encounter. In this regime, DeFi and MEV divergence will likely intensify.

Overall, while many transactions access largely stable sets of storage slots across lookbacks, a non-trivial subset exhibits substantial variability. As a result, both the storage accessed and the values written by a transaction are often difficult to predict in advance, complicating workload estimation and degrading transaction predictability from a user perspective.

7 Concluding Discussion and Future Directions for Mitigation

Our measurements reveal that EVM execution conditions are not stable: the resource mix varies across chains and shifts over time; persistent state growth is large and compositionally diverse; and execution outcomes (gas usage, storage access, success or failure) change when a transaction is re-evaluated on nearby historical states. Two assumptions underlying current gas metering are strained by these findings: that the workload mix is stable enough for one-dimensional gas pricing with fixed relative prices to remain appropriate, and that the state observed at simulation matches the state used at execution. Below, we first discuss how next-generation execution models sharpen both problems, then turn to mitigations.

7.1 Future Designs Amplify Both Problems

Delayed and asynchronous execution. Under delayed or asynchronous execution [31, 34], state drift between submission and execution means gas-limit-based charging can systematically overbill users or cause reverts. Unlike synchronous execution, where overestimating the gas limit is largely harmless, asynchronous designs penalize over-declaration (e.g., charging on the full declared gas limit) because actual usage is not yet known at inclusion time [31]. State-sensitive transactions then either pay a margin to absorb the drift or risk failing.

Optimistic parallel execution. Under optimistic parallel execution, diverging storage-access patterns across nearby states mean speculative caches are frequently invalidated on conflict-driven retry, potentially eroding parallelism's performance gains. Perhaps more importantly, statically declared access lists [11, 25] face the same issue: if the slots a transaction touches depend on state, the list cannot be populated accurately ahead of time.

Encrypted mempools and multiple concurrent proposers. Encrypted mempools widen the simulation-execution gap further: contending transactions remain hidden until decryption, so simulation cannot anticipate the writes that will land between submission and execution. Multiple concurrent proposers fragment block production across parties, so the ordering and pre-state of a transaction are not known until aggregation. Both widen the gap our measurements quantify between submission-time and execution-time state.

7.2 Mitigations

Multi-dimensional fee markets. When the workload mix drifts over time, one-dimensional gas metering cannot adapt on its own. The only recourse is manual repricing. Multi-dimensional fee markets [17, 2, 35] remove this rigidity by pricing distinct resources separately, letting prices for different resources converge to different equilibria. Our cross-chain and temporal measurements provide an empirical basis for the dimensions that matter: computation, storage reads, storage writes, call-related operations, and calldata.

Explicit pricing of persistent state. Persistent state growth is large (435 GB on Base versus 30 GB on Ethereum in 2025), highly bursty (Appendix A of the full version [23]), and compositionally diverse: storage slots dominate on both chains, bytecode accounts for roughly 25% of Base's growth, and account births remain a material component, especially for simple transfers on Ethereum. Because gas pricing treats state writes as transient per-block costs, it fails to reflect their permanent nature. An explicit storage cost per byte (rather than gas), whether as ongoing rent or upfront deposits [7, 6], should account for storage slots, contract bytecode, and account creation.

Reducing state sensitivity. Gas estimates change across nearby historical states for 46% of transactions on Base and 13.9% on Ethereum, with DeFi and MEV exhibiting especially high sensitivity. Some of this sensitivity is intrinsic to state-dependent opcode costs (cold versus warm access, zero versus non-zero `SSTORE` transitions). Given that sensitivity is structural, mitigations should target how the residual gap is billed rather than try to eliminate it. Under asynchronous execution, penalizing over-declaration is necessary for DoS protection, since actual usage is not yet known at inclusion time. Refunding a portion of the gap between declared limit and observed execution gas caps the overbilling penalty when a transaction lands on a more favorable state, while still discouraging inflated limits that reserve capacity the user does not need. For optimistic parallel execution, the category-level heterogeneity we observe suggests that schedulers can prioritize pre-warming and serialization for high-sensitivity categories (MEV, DeFi) while speculating freely on transfers and ERC-20 traffic, rather than a uniform speculation rule.

References

1 Amjad Aldweesh, Maher Alharby, Maryam Mehrnezhad, and Aad van Moorsel. The OpBench Ethereum Opcode Benchmark Framework: Design, Implementation, Validation and Experiments. *Performance Evaluation*, 146:102168, 2021. doi:10.1016/j.peva.2020.102168.

2 Guillermo Angeris, Theo Diamandis, and Ciamac Moallemi. Multidimensional Blockchain Fees Are (Essentially) Optimal. In *7th Conference on Advances in Financial Technologies (AFT 2025)*, LIPIcs, pages 24:1–24:23, 2025. doi:10.4230/LIPIcs.AFT.2025.24.

3 Guillaume Ballet, Vitalik Buterin, and Dankrad Feist. EIP-6780: SELFDESTRUCT Only in Same Transaction. Ethereum Improvement Proposals, March 2023. Created: 2023-03-25. URL: https://eips.ethereum.org/EIPS/eip-6780.

4 Dvir David Biton, Roy Friedman, and Yaron Hay. Ethereum Conflicts Graphed. In *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9. IEEE, 2025. doi:10.1109/ICBC64466.2025.11114508.

5 Vitalik Buterin. EIP-150: Gas Cost Changes for IO-Heavy Operations. Ethereum Improvement Proposals, September 2016. Created: 2016-09-24. URL: https://eips.ethereum.org/EIPS/eip-150.

6 Vitalik Buterin. A State Expiry and Statelessness Roadmap. Ethereum Foundation notes, 2021. URL: https://notes.ethereum.org/@vbuterin/verkle_and_state_expiry_proposal.

7 Vitalik Buterin. Possible Futures of the Ethereum Protocol, Part 5: The Purge, October 2024. Blog post. URL: https://vitalik.eth.limo/general/2024/10/26/futures5.html.

8 Vitalik Buterin, Eric Conner, Rick Dudley, Matthew Slipper, Ian Norden, and Abdelhamid Bakhta. EIP-1559: Fee Market Change for ETH 1.0 Chain. Ethereum Improvement Proposals, April 2019. Created: 2019-04-13. URL: https://eips.ethereum.org/EIPS/eip-1559.

9 Vitalik Buterin and Renan Rodrigues de Souza. EIP-684: Revert Creation in Case of Collision. Ethereum Improvement Proposals, March 2023. Created: 2023-03-20. URL: https://eips.ethereum.org/EIPS/eip-684.

10 Vitalik Buterin and Martin Swende. EIP-2929: Gas Cost Increases for State Access Opcodes. Ethereum Improvement Proposals, September 2020. Created: 2020-09-01. URL: https://eips.ethereum.org/EIPS/eip-2929.

11 Vitalik Buterin and Martin Swende. EIP-2930: Optional Access Lists. Ethereum Improvement Proposals, August 2020. Created: 2020-08-29. URL: https://eips.ethereum.org/EIPS/eip-2930.

12 Vitalik Buterin and Martin Swende. EIP-3529: Reduction in Refunds. Ethereum Improvement Proposals, April 2021. Created: 2021-04-22. URL: https://eips.ethereum.org/EIPS/eip-3529.

13 Danilo Rafael de Lima Cabral, Pedro Antonino, and Augusto Sampaio. Demystification and Near-Perfect Estimation of Minimum Gas Limit and Gas Used for Ethereum Smart Contracts. *Journal of Cloud Computing*, 2025. doi:10.1186/s13677-025-00751-y.

14 Category Labs. Monad Initial Spec Proposal. https://category-labs.github.io/category-research/monad-initial-spec-proposal.pdf, 2024.

15 CoinGecko. Layer 2 Blockchain Rankings. https://www.coingecko.com/en/chains/layer-2, 2025.

16 DefiLlama. DefiLlama Adapters: Open-Source DeFi Protocol Adapter Library. https://github.com/DefiLlama/DefiLlama-Adapters, 2024.

17 Theo Diamandis, Alex Evans, Tarun Chitra, and Guillermo Angeris. Designing Multidimensional Blockchain Fee Markets. In *5th Conference on Advances in Financial Technologies (AFT 2023)*, Leibniz International Proceedings in Informatics (LIPIcs), 2023. doi:10.4230/LIPIcs.AFT.2023.4.

18 Dune Analytics. Spellbook: A Community Maintained Library of Blockchain Analytics. https://github.com/duneanalytics/spellbook, 2024.

19 Evmos. XEN Token State Bloat Report. https://medium.com/evmos/xen-token-state-bloat-report-56a77a0715c0, 2022.

20 gmkung. blockscout-kleros-api: API Wrapper for Blockscout and Kleros Data. https://github.com/gmkung/blockscout-kleros-api, 2025.

21 Halborn. Blockchain's Storage Problem Is Growing: Are There Any Solutions? https://www.halborn.com/blog/post/blockchains-storage-problem-is-growing-are-there-any-solutions, 2022. Industry analysis / blog post.

22 Lioba Heimbach, Kushal Babel, and Jason Milionis. EVM Workload Analysis Code. https://github.com/liobaheimbach/EVM-Workloads-in-the-Wild, 2026. Public code and analysis artifact.

23 Lioba Heimbach, Kushal Babel, and Jason Milionis. EVM Workloads in the Wild: Evidence for Multi-Dimensional Gas Metering, State Growth, Delayed Execution, and Parallelism, 2026. Full version with appendices. doi:10.48550/arXiv.2606.19869.

24 Lioba Heimbach, Quentin Kniep, Yann Vonlanthen, and Roger Wattenhofer. DeFi and NFTs Hinder Blockchain Scalability. In *Financial Cryptography and Data Security (FC), Bol, Brac, Croatia*, May 2023.

25 Lioba Heimbach, Quentin Kniep, Yann Vonlanthen, Roger Wattenhofer, and Patrick Züst. Dissecting the EIP-2930 Optional Access Lists. In *Financial Cryptography and Data Security (FC), Willemstad, Curaçao*, March 2024.

26 Lioba Heimbach, Vabuk Pahari, and Eric Schertenleib. Non-Atomic Arbitrage in Decentralized Finance. In *IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA*, May 2024.

27 hildobby. Atomic MEV Table. https://dune.com/queries/3493305, 2024.

28 hildobby. MEV Sandwich Trades. https://dune.com/hildobby/sandwiches, 2024.

29 George Kadianakis, lightclient, and Alex Stokes. EIP-4444: Bound Historical Data in Execution Clients. Ethereum Improvement Proposals, November 2021. URL: https://eips.ethereum.org/EIPS/eip-4444.

30 Piper Merriam. EIP-7927: History Expiry Meta. Ethereum Improvement Proposals, March 2025. URL: https://eips.ethereum.org/EIPS/eip-7927.

31 Monad Foundation. Monad: The High-Performance EVM Blockchain. https://www.monad.xyz/, 2025.

32 Daniel Perez and Benjamin Livshits. Broken Metre: Attacking Resource Metering in EVM. In *Network and Distributed System Security Symposium (NDSS)*, 2020. doi:10.14722/ndss.2020.24267.

33 Yanjing Ren, Jia Zhao, Jingwei Li, and Patrick P. C. Lee. An Analysis of Ethereum Workloads from a Key-Value Storage Perspective. In *2025 IEEE International Symposium on Workload Characterization (IISWC)*, pages 394–406, Irvine, CA, USA, October 2025. doi:10.1109/IISWC66894.2025.00040.

34 Arran Schlosberg and Stephen Buttolph. ACP-194: Continuous Execution. https://github.com/avalanche-foundation/ACPs/tree/main/ACPs/194-continuous-execution, 2025.

35 Maria Silva, Davide Crapis, Anders Elowsson, and Toni Wahrstätter. EIP-8011: Multidimensional Gas Metering. Ethereum Improvement Proposals, August 2025. Status: Draft. URL: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-8011.md.

36 Maria Silva, Wei Han Ng, and Ansgar Dietrichs. EIP 8038: State-access Gas Cost Update. https://github.com/ethereum/EIPs/blob/master/EIPS/eip-8038.md, 2025.

37 Maria Inês Silva. Going Multidimensional: An Empirical Analysis on Gas Metering in the EVM, June 2025. Ethereum Research forum post. URL: https://ethresear.ch/t/going-multidimensional-an-empirical-analysis-on-gas-metering-in-the-evm/22621.

38 Maria Inês Silva. State Growth Scenarios and the Impact of Repricings, November 2025. Ethereum Research forum post. URL: https://ethresear.ch/t/state-growth-scenarios-and-the-impact-of-repricings/23476.

39 Ozan Solmaz, Lioba Heimbach, Yann Vonlanthen, and Roger Wattenhofer. Optimistic MEV in Ethereum Layer 2s: Why Blockspace Is Always in Demand. In *Seventh International Conference on Advances in Financial Technologies (AFT), Pittsburgh, PA, USA*, October 2025.

40 Martin Holst Swende. EIP-1884: Repricing for Trie-Size-Dependent Opcodes. Ethereum Improvement Proposals, March 2019. Created: 2019-03-28. URL: https://eips.ethereum.org/EIPS/eip-1884.

41 Wei Tang. EIP-2200: Structured Definitions for Net Gas Metering. Ethereum Improvement Proposals, July 2019. Created: 2019-07-18. URL: https://eips.ethereum.org/EIPS/eip-2200.

42 Toni Wahrstätter. Execution Dependencies, April 2025. Ethereum Research forum post. URL: https://ethresear.ch/t/execution-dependencies/22150.

43 Gavin Wood. EIP-161: State Trie Clearing (Invariant-Preserving Alternative). Ethereum Improvement Proposals, October 2016. Created: 2016-10-24. URL: https://eips.ethereum.org/EIPS/eip-161.

44 Fei Wu, Danning Sui, Thomas Thiery, and Mallesh Pai. Measuring CEX-DEX Extracted Value and Searcher Profitability: The Darkest of the MEV Dark Forest. In *7th Conference on Advances in Financial Technologies (AFT 2025)*, 2025.

Price Elasticity of Gas Demand on L1 and L2: Evidence from Ethereum and Arbitrum

Pranay Anchuri
Offchain, Princeton, United States

Akaki Mamageishvili
Offchain, Zurich, Switzerland

Abstract

We estimate the causal price elasticity of gas demand on Ethereum mainnet (L1) and Arbitrum One (L2), a quantity necessary for calibrating fee mechanism simulations, evaluating resource pricing reforms, and explaining observed usage patterns. A two-way fixed effects panel regression instrumented by each wallet's own lagged base fee removes the congestion-driven endogeneity that causes naive regressions to substantially underestimate demand sensitivity. On Ethereum mainnet (full year 2025), the pooled IV elasticity is -0.006^{***}, near-inelastic: a 10% fee increase reduces total gas demand by approximately 0.06%. On Arbitrum One (October 2025–April 2026), the pooled IV elasticity is -0.036^{**}. Both chains are inelastic in the aggregate, with L2 measurably more responsive than L1. A per-resource decomposition of L2 demand reveals elasticities ranging from modestly elastic computation (-0.027^{*}) to -0.27^{***} for refunds, with storage growth (-0.15^{***}) and calldata (-0.06^{*}) in between. Behavioral clustering identifies always-on protocol wallets as near-inelastic and high-volume operators as substantially more responsive, with cluster-level elasticities up to roughly 6× the pooled estimate. These results establish an empirical foundation for downstream simulations and for evaluating fee mechanism designs.

2012 ACM Subject Classification Applied computing → Economics

Keywords and phrases gas pricing, demand elasticity, instrumental variables, panel data, fixed effects, Arbitrum, Ethereum, transaction fee mechanism

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.35

Related Version *Full Version*: https://arxiv.org/pdf/2606.13555

1 Introduction

On Ethereum, the base fee adjusts in each block in response to block utilization, targeting 50% block capacity on average. Despite the mechanism's theoretical properties being well-studied [26, 25, 7, 18], the empirical demand response to fee changes remains poorly quantified. Knowing how much users reduce their onchain resource consumption when fees rise is necessary for multiple reasons. First, it allows calibrating simulations of the fee adjustment rule under different parameters. Second, evaluating potential throughput gains from multi-resource fee mechanisms requires demand elasticity estimates for each resource. Third, it helps explain current usage patterns under different fee regimes. Fourth, it helps find the effects of repricing certain opcodes[1].

Estimating demand elasticity naively is complicated by omitted variable bias. Higher network demand drives the base fee up (through the EIP-1559) and gas usage up (through more users transacting), producing a spurious positive correlation between fees and gas even

[1] The Ethereum ecosystem is considering increasing the gas target/limit while also increasing the gas cost of storage creation; Ethereum improvement proposal numbered 8037 (https://eips.ethereum.org/EIPS/eip-8037) is specifically proposed for this. Monad (also an EVM chain) increased the gas cost of storage creation at initialization time relative to Ethereum's default: https://category-labs.github.io/category-research/monad-initial-spec-proposal.pdf

© Pranay Anchuri and Akaki Mamageishvili;
licensed under Creative Commons License CC-BY 4.0

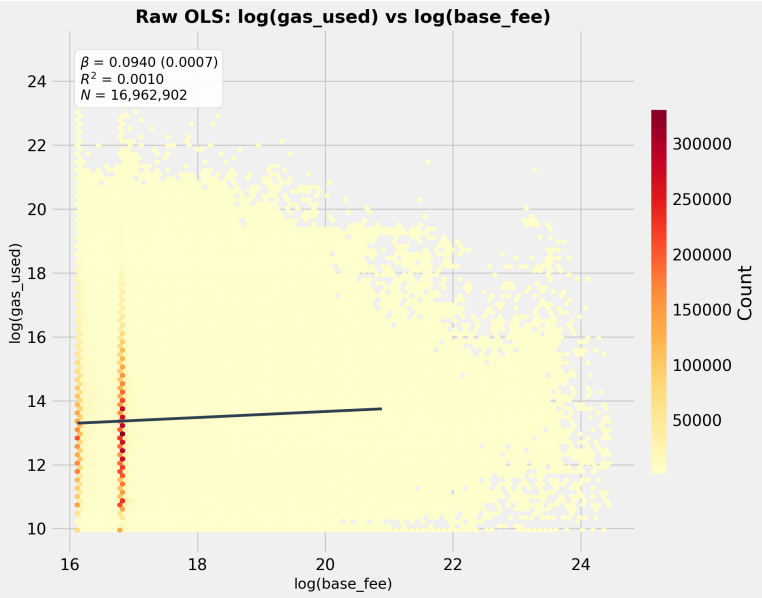


Figure 1 Raw pooled OLS of log gas usage on log base fee, Arbitrum L2 panel ($N = 16{,}939{,}781$ wallet-hour observations). The positive slope ($\hat{\beta} = +0.094$) contradicts the expected negative demand response, illustrating the upward bias induced by congestion-driven endogeneity. The two dense vertical bands at $\ln p \approx 16.1$ and ≈ 16.8 correspond to the Arbitrum base fee floor levels of 0.01 gwei and 0.02 gwei respectively (the floor was raised from 0.01 to 0.02 gwei mid-sample), where the base fee is anchored when demand falls below the gas target.

when isolated to wallet level. Figure 1 illustrates this directly: a pooled OLS regression of log gas usage on log base fee across 16,939,781 Arbitrum wallet-hour observations yields $\hat{\beta} = +0.094$, a positive coefficient that contradicts the expected negative demand response. This is in contrast to a typical demand elasticity estimation scenario where the price change is mostly exogenous, e.g., due to a supply drop. In the context of Ethereum and its rollups, the target supply is fixed and the price (gas fees) follows the demand exactly, as described above. Two-way fixed effects (FE: wallet and time) absorb level differences but still leave within-cell congestion variation uncontrolled. We address this residual endogeneity by using each wallet's own lagged base fee as an instrument: a fee level that is predetermined relative to current demand decisions and strongly correlated with the current fee through the autocorrelation built into the EIP-1559 fee adjustment rule.

Gas is priced as a single unit on both chains, but the underlying operations vary substantially in their resource footprint: a storage-creating transaction consumes a different mix of compute, I/O, and state resources than a pure computation. Evaluating proposals for multi-dimensional resource pricing requires knowing which dimensions actually respond to fee signals. The same decomposition informs the demand impact of opcode repricing, where increasing the cost of one resource category may or may not shift usage depending on the price sensitivity of that dimension.

The pooled elasticity is an average across all wallet types on the chain: oracle updaters that submit at fixed intervals, DeFi operators that optimize gas costs, and MEV searchers that time activity to congestion. These populations differ fundamentally in their relationship to fee changes, and a single estimate characterizes none of them.

Contributions

1. **Causal elasticity estimates for L1 and L2.** Using each wallet's own lagged base fee as an instrument, we obtain Instrumental Variables (IV) estimates of gas demand elasticity for Ethereum mainnet (-0.006) and Arbitrum One (-0.036).
2. **Per-resource decomposition for L2.** We estimate separate elasticities for seven Arbitrum resource dimensions, revealing that state-creating and state-clearing operations are far more price-sensitive than computation or reads, which can be justified by their time-insensitivity.
3. **Wallet-type heterogeneity.** K-means clustering on behavioral features identifies distinct user types with qualitatively different demand responses, from near-inelastic always-on protocol wallets to substantially elastic high-volume operators.

The pooled elasticity estimates confirm near-inelastic demand on both chains: a 10% fee increase reduces total gas demand by approximately 0.06% on Ethereum and 0.36% on Arbitrum. However, per-resource decomposition reveals that this aggregate inelasticity masks substantial resource-level heterogeneity: refunds (-0.27) and storage growth (-0.15) are the most price-sensitive dimensions, while computation (-0.027) is near-inelastic. Wallet clustering further identifies congestion-timing wallets for which the lagged-fee instrument is invalid, and restricts causal interpretation to the remaining segments.

The remainder of the paper is organized as follows. Section 2 reviews the EIP-1559 mechanism and L1/L2 fee regime differences. Section 3 describes the Ethereum and Arbitrum datasets. Section 4 develops the econometric framework. Section 5 presents the wallet clustering procedure. Sections 6 and 7 report the main elasticity results for L1 and L2 aggregate demand; Section 8 provides the per-resource decomposition. Section 9 discusses applications of elasticity analysis. Section 10 discusses implications and limitations.

2 Background

2.1 The EIP-1559 Fee Mechanism

Permissionless blockchains such as Ethereum need to control usage of their resources, to avoid long-lasting denial of service attacks and keep low hardware or bandwidth requirements for participating nodes in the network. For this reason, the Ethereum chain introduced Ethereum improvement proposal (EIP), numbered 1559, which adjusts the fee for resource consumption. There are different types of resources transactions consume, which are converted into gas units. Ethereum aims to operate on a target gas consumption. To achieve this goal, a simple rule is implemented: when the consumption of gas in the last block is larger than the target, the gas fee (in ETH) goes up; when the consumption of gas in the last block is lower than the target, the gas fee goes down; only when the gas usage is exactly equal to the target the gas fee stays the same. Gas fee changes should cause a short-term demand shift for the resources. Block sizes are usually upper bounded in terms of gas, commonly referred to as a block limit. The limit is set to two times the target. Recently, the target has moved up by a factor of 2 on Ethereum, from 15M to 30M, in an effort to scale the chain, with further increases expected.

A key design goal of EIP-1559 is fee predictability: the base fee is algorithmically calculated from the recent block utilization. This allows wallets to anticipate costs and plan resource consumption accordingly. This predictability both induces per-transaction optimization and deters participation during high-fee regimes. These are two distinct behavioral responses, and the former is the primary focus of this work.

2.2 L1 vs. L2 Fee Regimes

Major rollups adopt EIP-1559 or a variant of it to price their native resources and manage congestion. L2 fees are typically lower than L1 fees for two reasons: first, EIP-4844 expanded blob throughput so that L1 data availability costs for rollups are now minimal; second, lower hardware and bandwidth requirements reduce the marginal cost of an L2 block relative to an L1 block. Recent gas target increases on L1 have compressed the gap further, making L1 fees relatively cheap as well. The total fee incurred by an L2 end user consists of a native L2 congestion fee and an L1 data availability component; our analysis focuses on the native L2 base fee.

Recently, Arbitrum updated its pricing regime, introducing a gas ladder[2]. This pricing mechanism allows temporary outbursts of gas usage while keeping long-term targets fixed.

An interesting difference between these two types of chains is that Ethereum does not have a lower bound on gas price; it can go as low as 1 wei (10^{-18} ETH), while Arbitrum does have it. Recently, with the pricing mechanism update, Arbitrum increased it from 0.01 gwei to 0.02 gwei. A potential explanation for this is to deter spam transactions, the activity which is delegated to block builders on L1. Other major rollups enforce base fee floors that are substantially lower: OP Mainnet sets its minimum base fee to 0 wei (no effective floor), Base Mainnet to 0.005 gwei (5,000,000 wei), and Scroll to approximately 0.00012 gwei (set via configuration rather than the protocol). Arbitrum's 0.02 gwei floor is the highest among these rollups, roughly 4× Base's and orders of magnitude above the others.

The Arbitrum Nitro node is instrumented to track gas consumption across multiple resource dimensions (computation, storage reads and writes, storage growth, history growth, calldata, and refunds), attributing each opcode's cost to the corresponding category during execution. While the current Arbitrum fee mechanism charges a single gas price, the per-resource data provide a richer view of which operations are sensitive to fee changes, and form the basis for the per-resource demand decomposition in Section 8.

Note that when the base fee is at the floor for extended periods, fee variation is compressed and the identifying variation exploited by the instrument becomes sparse. This constraint is specific to L2 estimation and does not arise on L1, where fee variation is continuous and unbounded from below.

3 Data

3.1 Ethereum L1

The Ethereum dataset comprises all transactions executed on the Ethereum mainnet during 2025. Wallets are included in the study if they submitted at least 500 transactions during the year, ensuring sufficient within-wallet variation for the fixed effects estimation. After filtering on this threshold, the panel data contains 44,457 wallets observed over 8,751 hourly periods, yielding a total of 35,321,371 wallet-hour observations.

For each wallet-hour, the dependent variable is the wallet's total gas used, the independent variable is the wallet's own average base fee in that hour in gwei, and the control is the number of transactions submitted.

Both L1 and L2 panels are at hourly granularity. On Ethereum, we use daily time fixed effects rather than hourly: the Ethereum base fee adjusts each block (∼12 seconds), but the dominant fee variation is between days rather than within a day. Daily fixed effects efficiently absorb aggregate day-level demand shocks; the within-day, within-wallet hourly fee variation identifies β.

[2] Arbitrum gas-ladder AIP, Arbitrum Forum, 2025.

Table 1 L2 per-resource dimension coverage. Coverage is the fraction of total wallet-hours (16,939,781) where the resource consumption is non-zero.

Dimension	Description	Wallets	Wallet-hour obs.	Coverage (%)
computation	In-memory EVM execution	23,119	16,939,781	100.0
historyGrowth	Logs and events (append-only)	20,052	13,702,494	80.9
storageAccessRead	Reading existing state	22,652	16,126,735	95.2
storageAccessWrite	Updating existing state	20,883	13,475,351	79.6
storageGrowth	Increasing persistent state size	18,990	8,767,695	51.8
l2Calldata	Transaction calldata size on the child chain	20,799	15,782,132	93.2
refund	Gas refunded from storage-clearing operations	17,770	8,103,016	47.8

3.2 Arbitrum L2

The Arbitrum L2 dataset covers all Arbitrum One transactions from October 2025 through April 2026 (seven months). We use the same minimum threshold of 500 transactions to select the wallets. The resulting panel contains 23,119 wallets observed over 5,087 hourly periods, yielding a total of 16,939,781 wallet-hour observations.

In addition to aggregate gas usage, the Arbitrum dataset records per-transaction gas consumption across seven resource dimensions: *computation*, *historyGrowth*, *storageAccessRead*, *storageAccessWrite*, *storageGrowth*, *l2Calldata*, and *refund*. Table 1 describes each dimension. The dataset was constructed by instrumenting the Nitro execution engine to attribute each opcode's gas cost to one of these resource categories during transaction execution; the same opcode may map to different categories depending on execution context (for example, an `SSTORE` counts toward *storageGrowth* on first write but *storageAccessWrite* on updates). Computation is non-zero in every wallet-hour by construction; refund appears in fewer than half, reflecting the discretionary nature of state-clearing operations.

3.3 Panel Data Construction

Panels are constructed by aggregating raw transaction data to the (wallet, hour) level for both chains. For each wallet-hour, all transactions submitted by that wallet in the hour are aggregated to obtain total gas used and transaction count, and the base fee is averaged across those transactions. Time fixed effects are daily for Ethereum (365 day dummies over the hourly panel) and hourly for Arbitrum. We exclude system and precompile addresses from both panels.

The instrument, the wallet's own lagged base fee, is computed by retrieving the most recent period during which the wallet was active and the average base fee during that period.

4 Econometric Framework

In this section, we develop the estimation strategy for the price elasticity of gas demand. We begin by showing why a naive regression of gas usage on the base fee produces a biased estimate, formalize the bias using the omitted-variable formula, and then build the solution in two steps.

- Two-way fixed effects to absorb persistent wallet heterogeneity and time-dependent shocks.
- Instrumental variables to remove the residual endogeneity that fixed effects cannot eliminate.

4.1 The Identification Problem

The parameter of primary interest in this work is β, the price elasticity of gas demand: the percentage change in a wallet's gas consumption following a one-percent increase in the base fee. To isolate the source of bias, consider first a model with a single regressor,

$$\log g_i = \beta \log p_i + u_i, \tag{1}$$

where g_i is gas used, p_i is the base fee, and $u_i \sim N(0, \sigma^2)$. We use a log-log specification throughout.

The ordinary least squares (OLS) estimator satisfies

$$\hat{\beta}^{\text{OLS}} = \frac{\widehat{\text{Cov}}(\log p, \log g)}{\widehat{\text{Var}}(\log p)}, \tag{2}$$

and converges in probability to

$$\text{plim}\, \hat{\beta}^{\text{OLS}} = \beta + \frac{\text{Cov}(\log p, u)}{\text{Var}(\log p)}. \tag{3}$$

OLS is consistent if and only if $\text{Cov}(\log p, u) = 0$.

In our setting, this condition fails because of (latent) aggregate demand pressure, or congestion. Denote by D_t the aggregate level of demand in the time period t. Under the EIP-1559 adjustment rule, the base fee for each block is adjusted by a factor that is proportional to the excess utilization of the preceding block. Therefore, congestion raises the fee: $\text{Cov}(\log p, D_t) > 0$. At the same time, a high-congestion period reflects users submitting more transactions, which raises the usage: $\text{Cov}(\log g, D_t) > 0$. The term D_t is part of the error u in Equation (1) and these two conditions combine to give $\text{Cov}(\log p, u) > 0$. Therefore, the bias term in Equation (3) is positive. OLS overestimates the true elasticity β, biasing it toward zero or even reversing its sign. This is a structural bias and cannot be eliminated by increasing the sample size.

4.2 Two-Way Fixed Effects Model

The congestion described in Section 4.1 operates on two easily separable levels. First, wallets differ persistently: a high-frequency trading bot and a casual user differ in their gas consumption across all time periods for reasons unrelated to the base fee. Second, the network-level congestion shocks are common across all wallets within a given period. For example, a fee spike during an NFT drop affects all wallets simultaneously. Wallet fixed effects absorb the first source and the time fixed effects absorb the second.

Demand equation

The demand equation for wallet i in time period t is

$$\log g_{it} = \alpha_i + \gamma_t + \beta \log p_{it} + \phi \log n_{it} + u_{it}, \tag{4}$$

where p_{it} is wallet i's average base fee in period t, n_{it} is the number of transactions submitted (a control variable), and u_{it} is the residual. The term α_i absorbs all the time-invariant heterogeneity between wallets. These include bot versus human, typical gas usage per transaction, etc. The term γ_t absorbs all shocks at the period-level experienced by all wallets. These include network congestion level, time of day, day of week, and macro demand events. We also include the transaction count n_{it} because a wallet with more transactions mechanically consumes more gas. Controlling for n_{it} isolates the response in usage to the fee.

The control for n_{it} reflects a deliberate choice about the margin being estimated. The total gas g_{it} combines two distinct decisions made by the wallets: how many transactions to submit in a period t and how much gas to spend in each transaction. The extensive margin, transaction count, captures the participation decisions, whereas the intensive margin, gas used in a transaction, captures things such as reducing gas usage or substituting toward cheaper resources. By controlling for n_{it}, we partial out the extensive margin and identify β as the intensive margin elasticity. The latter is directly relevant in evaluating the efficiency of different fee mechanisms, since fee mechanisms should not merely deter activity but enable wallets to optimize their resource usage. Because n_{it} is fee-responsive, conditioning on it isolates the per-transaction (intensive-margin) response rather than the total effect, which is the margin most relevant to mechanism efficiency.[3] We condition on n_{it} rather than instrument for it because our estimand is the intensive-margin elasticity: instrumenting for n_{it} would recover the total elasticity, a different quantity that conflates fee-driven participation changes with per-transaction resource adjustments.

After absorbing the terms α_i and γ_t, the elasticity parameter β is identified from within-wallet and within-period variation. More specifically, β measures how a wallet i's gas usage changes when the fee deviates from both the wallet's own time-average and the period's cross-wallet average.

Two-way demeaning

Estimating β from Equation (4) requires either one-hot encoding the fixed effects or demeaning the outcome and the regressors. Let $\bar{y}_{i\cdot}$ denote the wallet time-mean, $\bar{y}_{\cdot t}$ the period cross-wallet mean, and $\bar{y}_{\cdot\cdot}$ the overall mean. The demeaned outcome is

$$\tilde{y}_{it} = y_{it} - \bar{y}_{i\cdot} - \bar{y}_{\cdot t} + \bar{y}_{\cdot\cdot}, \tag{5}$$

and the same transformation is applied to the regressors $\log p_{it}$ and $\log n_{it}$.

By the Frisch–Waugh–Lovell theorem [14, 20], OLS on the demeaned model is identical to including dummy variables for α_i and γ_t. We demean all variables by subtracting wallet and period means in alternation until convergence. This process terminates in fewer than 30 iterations in all panel datasets used in this study; the post-estimation fixed-effects recovery described in the full version of the paper applies the same algorithm to the full residuals and converges within 500 iterations.

Time granularity

Both analyses use hourly panels; the granularity of γ_t differs. On Ethereum mainnet, the base fee adjusts by at most $\pm 12.5\%$ per block (approximately 12 seconds), but the dominant variation is between days rather than within a day. We therefore use daily time fixed effects: γ_t absorbs day-level aggregate demand shocks, and β is identified from within-day, within-wallet hourly fee deviations. On Arbitrum, hourly fixed effects are used because the L2 base fee adjusts on an hourly schedule.

[3] Omitting $\log n_{it}$ from both stages reverses the sign of the estimate (+0.052 on Ethereum, +1.52 on Arbitrum): high-fee periods attract wallets that submit more and heavier transactions, so total gas co-moves positively with fees when transaction count is uncontrolled. Conditioning on n_{it} isolates the per-transaction response.

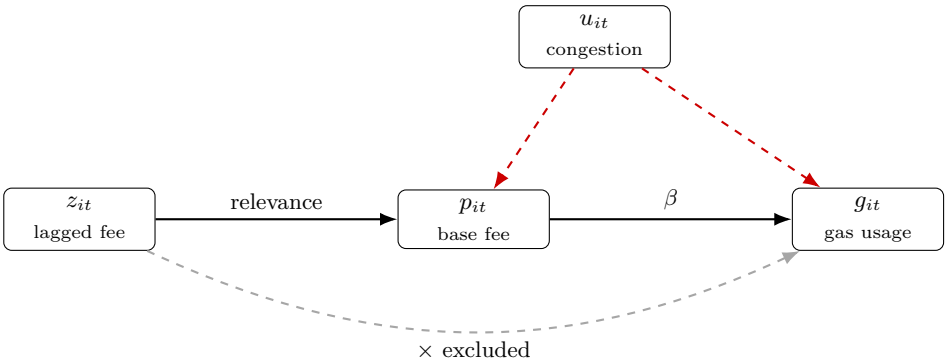


Figure 2 Causal structure of the gas demand model. Congestion u_{it} drives both the base fee p_{it} (base fee update rule) and gas usage g_{it} (through aggregate demand), inducing the endogeneity that biases OLS. The instrument z_{it} (the wallet's lagged fee) satisfies relevance (z_{it} predicts p_{it}) and the exclusion restriction (z_{it} has no direct path to g_{it}, marked ×). Instrumental variables estimation uses only the variation in p_{it} attributable to z_{it}, which is uncorrelated with u_{it}.

Residual endogeneity

The above two-way demeaning substantially attenuates the bias. The FE-OLS estimate of β is -0.007 on Ethereum and -0.016 on Arbitrum, both negative and consistent with downward-sloping demand. Within-period demand shocks remain in $\tilde{u}_{it}$ even after demeaning. For example, a wallet that transacts more in period t because of within-period congestion that also raised the fee in that period, contributes a positive correlation between $\tilde{p}_{it}$ and $\tilde{u}_{it}$. The two-way FE removes between-period congestion but not within-period endogeneity. This motivates the instrumental variables approach below.

4.3 Instrumental Variables

Figure 2 illustrates the causal structure of the model. Congestion u_{it} drives both the base fee p_{it} and gas usage g_{it} through separate channels, creating the endogeneity that results in an inconsistent estimate of the elasticity β. An instrumental variable z_{it} that shifts the fee but has no direct effect on gas demand breaks this linkage.

For each wallet i in period t, the instrument is the wallet's own average base fee in its immediately preceding active period:

$$z_{it} = \log p_{i,t-1}, \tag{6}$$

where $t-1$ denotes the most recent period before t in which wallet i submitted at least one transaction.

For z_{it} to be a valid instrument, the properties of *Relevance* and *Exclusion* must be established.

Relevance

The instrument must be correlated with the current fee: $\mathrm{Cov}(z_{it}, p_{it}) \neq 0$. The base fee exhibits strong autocorrelation because EIP-1559 adjusts it continuously in response to prior block utilization, so the fee in one period is a strong predictor of the fee in the next. The first-stage partial F-statistic confirms this correlation empirically. Results reported in Sections 6 and 7 far exceed the conventional threshold of 10 for reliable inference.

Table 2 Fraction of active hours in above-median-fee periods, by cluster (L2).

Cluster	Label	n wallets	Median	P25	P75
C0	Hi-vol burst	2,608	0.367	0.268	0.474
C1	Long-running	3,279	0.481	0.328	0.596
C2	Low-intensity	5,818	0.270	0.239	0.323
C3	Fee-avoiding	4,253	0.270	0.224	0.329
C4	Variable gas	697	0.302	0.248	0.363
C5	Cong-active†	6,466	0.754	0.400	0.879

†Excluded from IV estimation (Section 4.3).

Exclusion Restriction

The validity of the instrument also relies on the assumption that it is uncorrelated with the error u_{it}. The lagged base fee is *predetermined*, as it is a function of gas utilization in periods preceding t. Therefore, it is established before a wallet makes a marginal transaction decision at time t. Moreover, the two-way FE controls for both the time-invariant wallet characteristics and time-specific demand shocks. Under these conditions, the lagged fee affects the current usage only through its influence on the current base fee.

However, the exclusion restriction property is violated by wallets that strategically time their activity with congestion cycles. If a wallet's activity is sparse then the lagged fee z_{it} represents the peak of a prior congestion. Now, if the congestion is serially correlated then the instrument z_{it} becomes an informative signal of the current error term u_{it}. This leads to a correlation between the instrument and the latent demand shocks.

To mitigate the bias introduced by this channel, we identify such congestion sensitive wallets based on the proportion of wallet's activity during periods where the base fee exceeds the median hourly base fee. Wallets with an anomalously high fraction are the ones that are disproportionately active during spikes. For these wallets, the lagged fee is invalid as an instrument. We identify this as cluster C5 (congestion-active, Section 5): the median C5 wallet conducts 75.4% of its active hours in above-median-fee periods (IQR: 40.0–87.9%), compared with 27.0–48.1% for the five retained clusters (Table 2). We therefore restrict causal interpretation to the five clusters whose timing patterns are consistent with the exclusion restriction.

4.4 Calculating β using Two-Stage Least Squares (2SLS)

Building on the discussion in this section, we now describe how we estimate the elasticity β. Prior to estimation, all variables are two-way demeaned as described in Section 4.2.

In the first stage, the demeaned fee is regressed on the demeaned instrument and the demeaned transaction count (control):

$$\tilde{p}_{it} = \pi\, \tilde{p}_{i,t-1} + \delta\, \tilde{n}_{it} + v_{it}. \tag{7}$$

Here $\tilde{p}_{i,t-1}$ is shorthand for the two-way demeaned instrument $\tilde{z}_{it}$, where $z_{it} = \log p_{i,t-1}$. The fitted values $\hat{p}_{it}$ capture only the variation in the fee that is driven by the instrument.

In the second stage, the fee is replaced with the fitted values from the first stage.

$$\tilde{g}_{it} = \beta\, \hat{p}_{it} + \phi\, \tilde{n}_{it} + \varepsilon_{it}. \tag{8}$$

With a single endogenous regressor (base fee) and a single instrument, the estimate from 2SLS reduces to

$$\hat{\beta}^{\mathrm{IV}} = \frac{\widehat{\mathrm{Cov}}(z,\, \tilde{g})}{\widehat{\mathrm{Cov}}(z,\, \tilde{p})}. \tag{9}$$

Interpretation

The 2SLS estimator in Equation (9) identifies the local average treatment effect (LATE). It is the average demand response for wallets whose fee in period t is affected by the lagged fee. This is the subpopulation of wallets that respond to prior price levels when deciding the current gas usage. Wallets that execute on a fixed schedule regardless of the base fee are not shifted by the instrument and consequently have negligible contribution to the estimate.

Preview of main results

Applying this framework to the full panels, the pooled IV elasticity is -0.006^{***} (SE $= 0.0005$) on Ethereum and -0.036^{**} (SE $= 0.014$) on Arbitrum. Both estimates are near-inelastic and are detailed in Sections 6 and 7. Section 5 and Section 8 decompose these pooled estimates by behavioral cluster and resource dimension, respectively.

5 Wallet Clustering

The IV β estimate from Section 4 is a precisely identified quantity, but it is a weighted average of demand responses across all wallet types in the data. In general-purpose blockchain networks like Ethereum and Arbitrum, wallets span a wide range of behavioral regimes: oracle updaters, arbitrageurs, DeFi operators, and retail users. These wallets differ not only in their average gas consumption but also in how they respond to fee changes. An oracle contract that must submit a transaction at regular intervals to maintain the on-chain state faces no meaningful price sensitivity. At the same time, a high-volume arbitrageur may substitute toward lower fee periods. Pooling all wallet types into a single regression Equation (4) produces an average elasticity estimate that does not characterize any particular wallet class.

Clustering the wallets into behaviorally homogeneous groups prior to IV estimation recovers within-cluster elasticities that are interpretable as the demand response of a coherent user type. It also serves a diagnostic role in our analysis: wallets that activate disproportionately during congestion spikes are localized in a single cluster identifiable by their high-fee fraction, as discussed in Section 4.3.

5.1 Clustering Procedure

Wallets are partitioned using k-means clustering based on a set of behavioral features computed from each wallet's transaction history over the entire sample period. Before clustering, we also standardize the features to zero mean and unit variance. We select the number of clusters k with the elbow method: the within-cluster sum of squared distances is computed for $k = 2, \ldots, 12$, and k is chosen at the point of diminishing returns.

Elbow plots for all three analyses are shown in the full version of the paper.all three exhibit a clear kink at $k = 6$.

Table 3 Clustering features. The top five are used in all analyses; the bottom four are added for the L2 per-resource clustering only. All features are standardised to zero mean and unit variance before clustering.

Feature	What it captures
Behavioural features (all analyses)	
log(tx_frequency)	Transaction rate per active hour; separates high-throughput operators from occasional users.
log(gas/tx)	Mean gas per transaction; separates simple transfers from complex contract interactions.
log(active_hours)	Total hours with at least one transaction; separates persistent from episodic wallets.
gas_cov	Coefficient of variation (σ/μ) of gas per transaction; low values indicate a fixed operation type, high values indicate volatile mixed activity.
high_fee_fraction	Fraction of active hours above the median hourly fee; identifies wallets that transact preferentially during congestion. Also used as the exclusion restriction diagnostic (Section 4).
Resource composition features (L2 per-resource only)	
frac_computation	Share of gas from EVM opcode execution.
frac_storageAccessRead	Share from reading existing storage slots.
frac_l2Calldata	Share from L2 calldata.
frac_storageGrowth	Share from creating new storage slots.

5.2 Features

Table 3 lists all features used for clustering. The five behavioral features are used in both the total-gas and per-resource analyses. For the resource level analysis, we add four resource composition features. Each of these features measures the fraction of a wallet's total gas attributable to a given resource dimension averaged over the entire time period.

The remaining dimensions (storageAccessWrite, historyGrowth, refund) are excluded from the clustering features as their variation is already captured by the included features.

5.3 Cluster Profiles

5.3.1 L1 clusters

Table 4 reports the six L1 clusters, ordered by size. Figure 3 shows a t-SNE projection of the five-dimensional feature space, confirming that the resulting clusters occupy geometrically distinct regions.

The clusters reflect qualitatively different wallet types. C3 is the most persistently active, with a median of 3309 active hours (38% of all hours in the year). C1 has the highest frequency (25 transactions per active hour) but operates in short bursts (53 median active hours) and generally avoids high-fee periods (hff = 0.34). These wallets are likely to be opportunistic and sensitive to the base fee. C0 is highly active during high-fee periods suggesting price insensitivity or time-sensitive applications. C4 is the smallest cluster but exhibits the highest gas coefficient of variation, indicating wallets that alternate between simple and complex transactions. C5 is similar to C1 with respect to high-fee fraction but uses 3.5× more gas per transaction.

Table 4 L1 (Ethereum) wallet clusters ($k = 6$, full year 2025). Medians reported for each feature. tx_freq: transactions per active hour; gas/tx: average gas per transaction (thousands); act. hrs: active hours in sample; gas CoV: coefficient of variation of gas per transaction; hff: high-fee fraction.

Cl.	Wallets (%)	tx_freq	gas/tx (K)	act. hrs	gas CoV	hff
C0	15,135 (34.0%)	2.52	130	330	0.92	0.69
C1	3,609 (8.1%)	25.1	50	53	0.96	0.34
C2	11,625 (26.1%)	1.54	35	543	0.68	0.50
C3	4,123 (9.3%)	2.93	114	3,309	0.82	0.50
C4	1,280 (2.9%)	3.04	106	368	3.33	0.53
C5	8,685 (19.5%)	2.86	171	303	1.01	0.34

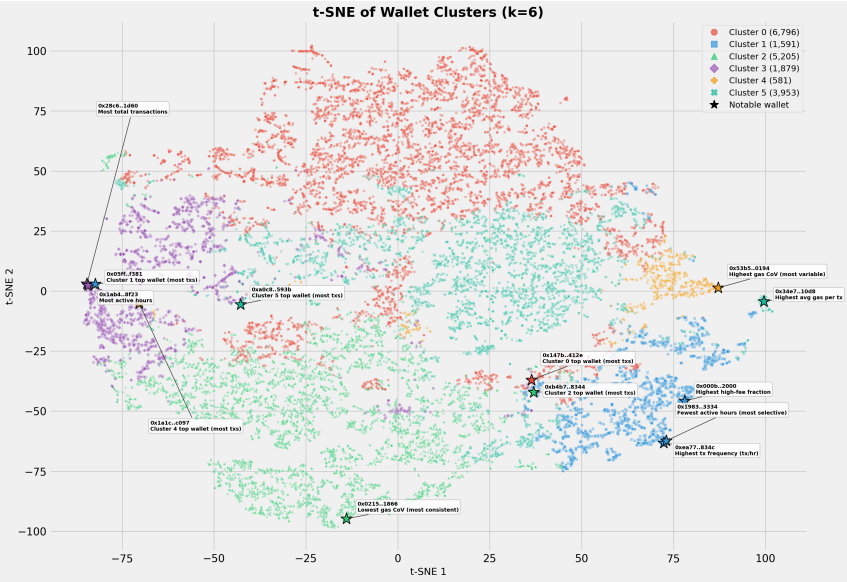


Figure 3 t-SNE projection of L1 wallet feature vectors ($k = 6$, five behavioral features). Each color represents one cluster. Clusters occupy well-separated regions, confirming meaningful behavioral separation.

5.3.2 L2 clusters (total gas)

Table 5 reports the six L2 clusters. Figure 4 shows the corresponding t-SNE projection.

C0 is the most active cluster by transaction rate (59 transactions per active hour) and with the highest mean hourly gas consumption (16.9M gas per hour) but with a very short median activity. C1 combines modest transaction activity with a very high number of active hours. C2 is the biggest cluster and has both low transaction frequency and gas usage. C3 has the lowest high-fee fraction, indicating wallets that are sensitive to the base fee. C4 has very high gas coefficient of variation. C5 is the cluster with highest high-fee fraction. The wallets transact predominantly in the most expensive gas fee regimes. As discussed in Section 4.3, the exclusion property is implausible for this cluster.

L2 clusters (per-resource)

The per-resource analysis uses nine features (five behavioral plus four resource composition fractions). Six clusters emerge at the same $k = 6$ elbow. C1 has high transaction frequency, high-fee fraction, and transactions dominated by computation and storage reads. C2 is the

Table 5 L2 (Arbitrum) total-gas wallet clusters ($k = 6$, October 2025–April 2026). Medians reported. gas/hr: mean gas used per active hour (millions). Other columns as in Table 4.

Cl.	Wallets (%)	tx_freq	gas/hr (M)	act. hrs	gas CoV	hff
C0	2,608 (11.3%)	58.74	16.87	24	1.02	0.53
C1	3,279 (14.2%)	8.43	2.14	2,858	1.25	0.51
C2	5,818 (25.2%)	2.31	0.17	532	0.97	0.54
C3	4,253 (18.4%)	5.24	1.11	166	0.90	0.34
C4	697 (3.0%)	3.42	0.51	898	4.42	0.48
C5	6,466 (28.0%)	16.81	0.90	117	1.48	0.90

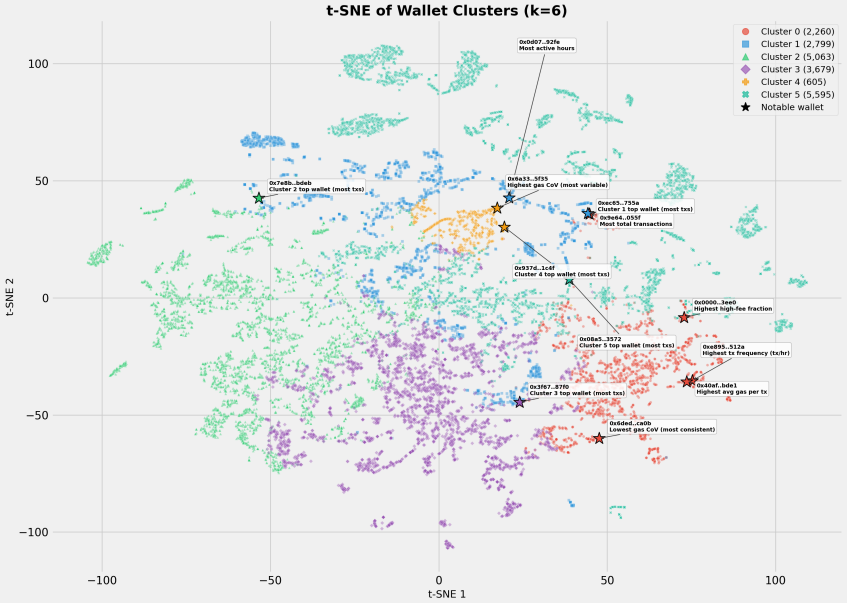


Figure 4 t-SNE projection of L2 total-gas wallet feature vectors ($k = 6$, five behavioral features). Clusters are well-separated. C5 (high-fee fraction 0.90, upper right) forms a tight distinct region.

high-volume cluster. C3 (35.0%) is the largest cluster, with general DeFi activity. C4 is computation-dominant. C5 wallets have significant storage-growth fractions. Full cluster profiles are reported in the full version of the paper.

Adversarial-activity validation for both L2 analyses is reported in the full version of the paper.

6 Results: Ethereum L1

6.1 First Stage

The instrument is strongly relevant on L1. The overall first-stage coefficient is $\hat{\pi} = 0.480$. Although the partial F-statistics are inflated by the large sample size, the first-stage coefficient confirms a genuinely strong first stage, reflecting the strong autocorrelation in the daily L1 base fee. The first-stage partial R^2 is 0.252, confirming that the lagged base fee explains a substantial share of current-fee variation even after partialling out the fixed effects.

Table 6 Ethereum L1 IV elasticity estimates by cluster. Standard errors clustered at the wallet level. $^{**}p < 0.01$, $^{***}p < 0.001$; n.s. not significant at 5%.

Group	Wallets (%)	$\hat{\beta}_{\text{OLS}}$	$\hat{\beta}_{\text{IV}}$	SE	Partial F
Pooled	44,449 (100%)	−0.0074	-0.0059^{***}	0.0005	11,873,625
C0 (hi-fee act.)	15,135 (34.0%)	−0.0110	-0.0236^{***}	0.0015	616,714
C1 (bursty)	3,609 (8.1%)	+0.0011	+0.0071 n.s.	0.0051	111,703
C2 (low-intens.)	11,625 (26.1%)	−0.0015	-0.0016^{**}	0.0006	1,147,259
C3 (always-on)	4,123 (9.3%)	−0.0022	−0.0010 n.s.	0.0007	15,170,218
C4 (variable)	1,280 (2.9%)	−0.0116	−0.0066 n.s.	0.0036	298,077
C5 (fee-avoid.)	8,685 (19.5%)	−0.0308	-0.0212^{***}	0.0022	592,378

6.2 IV Estimates

Table 6 reports 2SLS estimates for the pooled sample and each of the six clusters in Section 5.3.1. The pooled elasticity is -0.0059^{***} (SE = 0.0005): a 10% increase in the base fee reduces total L1 gas demand by approximately 0.06%. L1 demand is near-inelastic. The aggregate near-inelasticity reflects the large share of low-intensity (C2, 26.1%) and always-on (C3, 9.3%) wallets whose consumption does not vary with the fee. Given the size of the panel, statistical significance is easily attained; the economically relevant takeaway is that aggregate L1 gas demand is, for practical purposes, insensitive to fee changes. The two most elastic clusters are C0 (34.0%), with $\hat{\beta} = -0.0236^{***}$ (SE = 0.0015), and C5 (19.5%), with $\hat{\beta} = -0.0212^{***}$ (SE = 0.0022). Wallets in C3 (9.3%) have the highest first-stage F ($\hat{\pi} = 0.689$, $F = 15{,}170{,}218$) yet the elasticity is near-zero and statistically insignificant (-0.0010, $p = 0.17$). These are wallets that do not respond to fee signals.

On L1, the OLS and IV estimates are close in magnitude: daily fixed effects already absorb most of the between-day congestion that drives the bias, so instrumenting moves the pooled estimate only slightly and the sign of the adjustment varies across clusters. The correction is larger on L2 (Section 7), where hourly variation leaves more residual endogeneity.

The negative slope is visible after removing wallet and day fixed effects, confirming a downward-sloping demand relation in the partialled-out data. See full version of the paper.

7 Results: Arbitrum L2 Aggregate Demand

7.1 First Stage

The Arbitrum total-gas panel uses hourly time fixed effects. The pooled first-stage coefficient is $\hat{\pi} = 0.052$, significantly lower than on Ethereum. This reflects the frequency with which Arbitrum's base fee rests at its lower bound. Because the partial F-statistic scales with sample size, its large value reflects the size of the panel rather than instrument strength; the first-stage partial R^2 is only 0.007, confirming that the lagged fee explains a negligible share of current-fee variation on L2 after absorbing the fixed effects. The L2 estimates are therefore identified from limited fee movement.

The pooled partial F-statistic is 118,405, well above the commonly used threshold. Per-cluster F-statistics range from 1,361 (C0, high-volume but sparsely active per hour) to 113,875 (C1, long-running), confirming the relevance property of the instrument.

Table 7 Arbitrum L2 aggregate IV elasticity estimates by cluster. Standard errors clustered at the wallet level. $^{**}p < 0.01$, $^{***}p < 0.001$; n.s. not significant at 5%; n.i. not identified. † excluded from causal interpretation.

Group	Wallets (%)	$\hat{\beta}_{\text{OLS}}$	$\hat{\beta}_{\text{IV}}$	SE	Partial F
Pooled	23,119 (100%)	−0.0161	−0.0359**	0.0141	118,405
C0 (hi-vol.)†	2,606 (11.3%)	−0.0440	−1.163 n.i.	0.1795	1,361
C1 (long-run.)	3,279 (14.2%)	−0.0285	−0.0782**	0.0268	113,875
C2 (low-intens.)	5,818 (25.2%)	−0.0131	−0.0057 n.s.	0.0374	4,662
C3 (fee-avoid.)	4,253 (18.4%)	−0.0311	+0.0792 n.s.	0.0574	2,709
C4 (variable)	697 (3.0%)	−0.0306	−0.1272 n.s.	0.1687	2,749
C5 (cong.-act.)†	6,466 (28.0%)	−0.0102	+0.0314 n.s.	0.0249	7,163

7.2 IV Estimates

Table 7 reports IV estimates for the Arbitrum aggregate demand model. The pooled elasticity is -0.0359^{**} (SE = 0.0141), six times larger in magnitude than the Ethereum estimate. A 10% fee increase on Arbitrum reduces total gas demand by an estimated 0.36%, less elastic in absolute terms but measurably more responsive than Ethereum.

The most credible cluster-level estimate is C1 (long-running high-gas wallets, 14.2%), with $\hat{\beta} = -0.0782^{**}$ (SE = 0.0268, F = 113,875). The median number of active hours is highest in this cluster, giving the instrument strong identifying power. The IV estimate for C2 (25.2%) is near-zero and statistically insignificant (-0.0057, $p = 0.88$). Wallets in this cluster are occasional, low-frequency users that do not adjust gas usage in response to fee variation.

We exclude C5 and C0 from causal interpretation. C5 (28.0%, median high-fee fraction = 0.90) yields a positive estimate ($+0.0314$, $p = 0.21$), consistent with exclusion restriction failure: these wallets time activity to congestion periods, so the lagged fee captures position during the previous congestion period. C0 (11.3%) yields $\hat{\beta} = -1.163$, an anomalously large magnitude that is driven by very low active hours and is not identified.

The pooled estimate includes the C0 and C5 wallets whose instruments are potentially invalid (Section 4.3). Since C5 contributes a near-zero positive estimate and C0 an anomalously large negative one, the pooled -0.036 is an average over heterogeneous reliability; the valid-cluster estimates in Table 7 provide the within-cluster causal figures.

A natural alternative identification strategy would exploit the 0.01 → 0.02 gwei base fee floor increase mid-sample as a quasi-experimental discontinuity. This change, however, binds only for transactions priced at or near the floor – primarily spam and dust activity – while the 500-transaction wallets in our sample operate well above the floor throughout. The floor increase therefore does not induce a sharp discontinuity in the fee faced by our study population and cannot serve as an instrument for this sample.

The two-way demeaned binscatter confirms a negative demand slope after removing wallet and hour fixed effects and log transaction count (see full version).

8 Results: Arbitrum L2 Per-Resource Demand

8.1 Per-Dimension Pooled Estimates

Table 8 reports IV elasticity estimates for each of the seven Arbitrum resource dimensions using the pooled sample. The estimates span a wide range across resource dimensions. These are intensive-margin elasticities, estimated on wallet-hours with positive consumption of

Table 8 L2 per-resource IV elasticity estimates, pooled sample. Intensive margin (wallet-hours with $\text{gas}_d > 0$). $^{*}p < 0.05$, $^{***}p < 0.001$; n.s. not significant at 5%.

Dimension	Wallets	$\hat{\beta}_{\text{OLS}}$	$\hat{\beta}_{\text{IV}}$	SE	Partial F
computation	23,119	−0.0026	-0.0274^{*}	0.0123	118,405
historyGrowth	20,052	−0.0109	−0.0109 n.s.	0.0304	46,410
storageAccessRead	22,652	+0.0166	$+0.0344^{*}$	0.0153	118,185
storageAccessWrite	20,883	+0.0040	$+0.1186^{***}$	0.0274	47,843
storageGrowth	18,990	−0.0388	-0.1510^{***}	0.0446	26,104
l2Calldata	20,799	−0.0125	-0.0575^{*}	0.0228	71,727
refund	17,770	−0.0232	-0.2688^{***}	0.0414	30,666

the given resource; they describe the response of wallets that continue to use a resource, not the economy-wide change in its total consumption. Computation gas, the dominant component of most transactions, is modestly elastic at -0.0274^{*} (SE = 0.0123). Storage growth and refunds are the most price-sensitive: -0.1510^{***} (SE = 0.0446) and -0.2688^{***} (SE = 0.0414) respectively. L2 calldata shows a significant negative elasticity of -0.0575^{*} (SE = 0.0228). History growth is statistically insignificant from zero ($p = 0.72$).

8.2 Positive Elasticities

The positive IV estimates for storage access reads ($+0.034^{*}$) and writes ($+0.119^{***}$) do not represent a genuine demand response. When fees rise, wallets with lower-necessity operations exit the sample. This leaves a selected set of wallets whose operations require storage access. In such periods, the conditional average storage access rises even as total volume falls, producing a spurious positive elasticity. Because this selection operates whenever fees rise, it biases every per-resource estimate in the positive direction. The dimensions that remain significantly negative, storage growth and refunds, are therefore conservative: the true intensive-margin response is at least as large as the estimates reported here. The small or positive estimates for computation, reads, and writes should be read as responses among continuing wallets rather than economy-wide demand elasticities.

The share of active wallet-hours using storage reads is effectively constant across base-fee deciles (0.951–0.956), confirming that read-intensive wallets are the last to exit when fees rise and dominate the remaining sample at high fees. Storage writes show a clearer decline (0.835 at the lowest fee decile to 0.773 at the highest), consistent with write-light wallets exiting while write-intensive operations persist. By contrast, the dimensions with the most negative IV estimates, storage growth and refunds, show no systematic selection pattern across deciles, supporting their interpretation as genuine demand responses rather than selection artifacts.

8.3 Most Elastic Resources: Storage Growth and Refunds

Storage growth ($\hat{\beta} = -0.1510^{***}$) and refunds ($\hat{\beta} = -0.2688^{***}$) are the most price-sensitive resource dimensions in the pooled estimate. Both effects are concentrated in C3 of the per-resource clustering (35.0% of the sample): -0.3709^{***} for storage growth (SE = 0.0549) and -0.4157^{***} for refunds (SE = 0.0514).

The wallets in this cluster have routine operations for which creating new storage slots and storage cleanup are not time sensitive. When fees rise, they defer these operations, which lead to a measurable reduction in usage of these resources.

Table 9 L2 per-resource IV $\hat{\beta}$ for key clusters. $^{*}p < 0.05$, $^{***}p < 0.001$; n.s. not significant at 5%.

Dimension	C2 high-volume (12.8%)	C3 general DeFi (35.0%)
computation	-0.1590^{***}	-0.0232 n.s.
historyGrowth	-0.6859^{***}	-0.0354 n.s.
storageAccessRead	-0.3189^{***}	-0.0274 n.s.
storageAccessWrite	-0.3207^{***}	$+0.0069$ n.s.
storageGrowth	$+0.1274$ n.s.	-0.3709^{***}
l2Calldata	-0.3560^{***}	$+0.0107$ n.s.
refund	$+0.0781$ n.s.	-0.4157^{***}

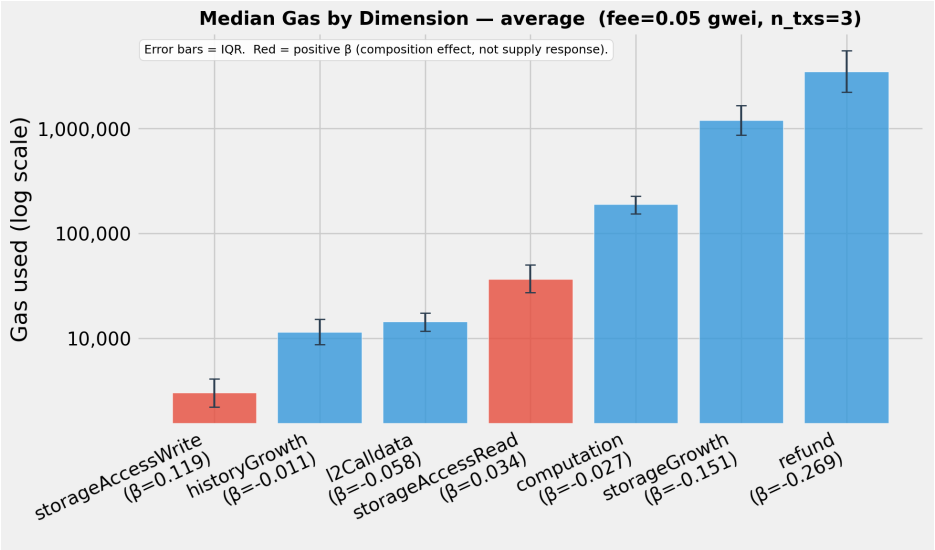


Figure 5 Median gas usage by resource dimension for the pooled Arbitrum L2 model (`average` regime, fee = 0.05 gwei). Error bars show the inter-quartile range. Blue bars have negative $\hat{\beta}$ (demand response); red bars have positive $\hat{\beta}$.

8.4 Cluster-Level Per-Dimension Results

Table 9 presents cluster-level estimates for the two clusters with the most interpretable IV results. C2 (12.8%) shows large negative elasticities across nearly all dimensions except storage growth and refunds, confirming price elasticity in this segment. C3 (35.0%) is near-zero across most dimensions but strongly negative for storage growth and refunds, consistent with the pattern observed in the pooled sample.

Figure 5 summarizes median gas usage and the pooled IV elasticity across dimensions.

9 Downstream applications

The IV estimates from Section 4.3 include the causal parameters of a demand model that can be used to simulate gas usage by the wallets in various gas fee regimes. In this section, we describe how the full demand model is used to generate synthetic gas demand distributions under counterfactual fee regimes. Such simulations help evaluate the performance of fee mechanism designs beyond the observed range. We present the construction of the demand model here. Figure 6 illustrates the model applied to a fee sweep from 0.005 to 5 gwei

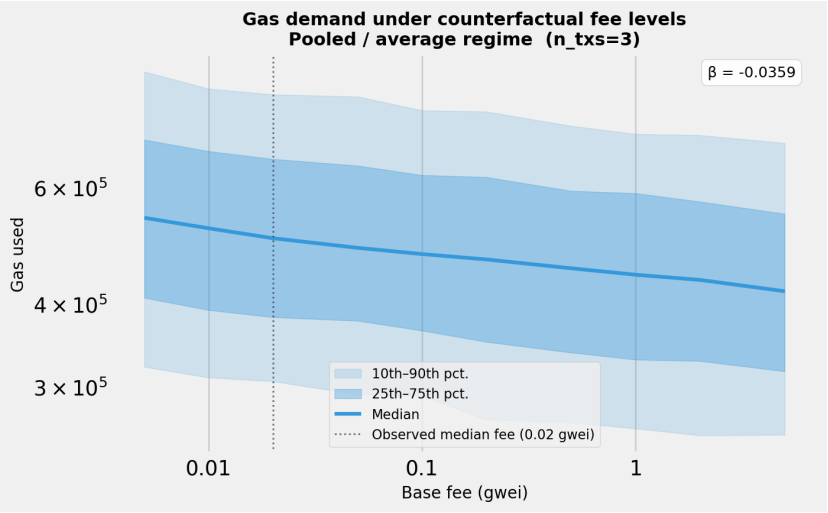


Figure 6 Simulated gas demand under counterfactual base fees, `average` regime. Bold line: median. Inner band: 25th–75th percentile. Outer band: 10th–90th percentile. Dotted vertical line: observed median base fee (0.02 gwei).

for the `average` regime ($\hat{\alpha} \in [P_{40}, P_{60}]$, $\hat{\gamma} \in [P_{40}, P_{60}]$, $n_{\text{txs}} = 3$). The downward-sloping median response reflects the pooled L2 elasticity ($\hat{\beta} = -0.036$) across the full fee range; the interquartile and 10th–90th percentile envelopes shift in parallel, confirming that the elasticity acts multiplicatively on the gas distribution rather than compressing or expanding it. A systematic evaluation of specific mechanism parameters is left to future work.

Recovering Fixed Effects

The 2SLS method described in Section 4.4 estimates only $\hat{\beta}$ and $\hat{\phi}$ of the structural demand equation (4). The wallet and time fixed effects are recovered by computing the residuals of the outcome and applying alternating projections. The residuals are then fitted with a Student-t distribution to accommodate heavy tails that are commonly observed in gas usage. The alternating projections procedure and the sampling algorithm used to generate counterfactual gas demand distributions are detailed in the full version.

10 Discussion

10.1 Why L2 Is More Elastic Than L1

Several factors may contribute to the higher measured elasticity on L2. The two chains attract different wallet populations: L2 users have opted into a lower-cost environment and may be more price-sensitive by selection, whereas L1 wallets include a higher share of protocol-level operations that are insensitive to fee changes. Arbitrum's base fee floor and gas ladder also create a distinct incentive environment compared to Ethereum. Finally, the difference in time granularity between the two estimates, daily fixed effects on L1 versus hourly on L2, may mechanically affect the measured elasticity in ways that are difficult to disentangle from a genuine behavioral difference.

One particular difference is the relative absence of a large share of reverted transactions on L1, unlike L2s, where the sequencer's mempool is kept private and arbitrageurs try to probabilistically backrun state changes. In times of high asset price volatility, which results in higher on-chain fees, such transactions dominate on-chain gas consumption. On L1, the task of filtering reverted transactions is delegated to the block builders.

10.2 Implications for Fee Mechanism Design

Ethereum and Arbitrum both use a variant of the same congestion management fee mechanism, namely, converting all resource usage into a single gas unit and pricing the gas according to its own EIP-1559 parameters. Note that not all resources are the same. Namely, state growth is a long-term issue for the chain: if the state grows too much, accessing and writing to state memory become slower, and it may no longer fit on disk. Other resources have more immediate constraints. Storage capacity not allocated at a given time can be allocated later, whereas computation and state-access resources not consumed in a given period cannot be recovered. These distinctions call for at least different ratios between limit and target per resource, instead of a constant ratio imposed by converting all resources to a single gas unit, to better utilize the chain. Fully-fledged multi-dimensional resource pricing schemes have been proposed and analyzed by a number of industry and academic works. Our findings on multi-dimensional resource elasticities can help inform optimal parameter ranges for each resource, given EIP-1559-type mechanisms are applied to each resource independently. As an illustration, the comparatively elastic refund and storage-growth dimensions (pooled -0.27 and -0.15) imply that pricing these resources more aggressively would meaningfully curb their use, whereas repricing computation (near-zero elasticity) would mainly raise revenue without changing usage. A multi-dimensional mechanism could therefore apply tighter independent pricing to the elastic, state-affecting resources, where price has traction.

10.3 Limitations

The IV estimator identifies a local average treatment effect: the demand response of wallets whose current fee is shifted by the lagged fee. Always-on wallets (C3 on L1) and congestion-timing wallets (C5 on L2) are excluded from causal interpretation; the pooled estimates do not characterize these populations. The L2 sample covers seven months (October 2025 through April 2026) and reflects the fee regime in effect during this window; as Arbitrum's base fee floor and gas ladder parameters evolve, the measured elasticities may shift. The 500-transaction minimum threshold ensures sufficient within-wallet variation for fixed-effects estimation but excludes low-frequency users, and the estimated elasticities may not generalize to that population. Finally, the per-resource decomposition is feasible for Arbitrum because the Nitro node tracks per-category gas consumption; no equivalent data exists for Ethereum L1.

The methodology applies to any chain with a congestion-responsive fee adjustment rule (an EIP-1559 variant or equivalent), sufficient wallet-level panel data, and temporal fee variation above the protocol floor. For chains with fixed-price auctions or purely flat fees, the lagged-fee instrument loses relevance and a different identification strategy would be required. Chains where the base fee spends most of its time at the floor (as L2 does during low-congestion periods) should use the partial R^2 alongside the partial F to assess instrument strength, as shown here for Arbitrum.

11 Related Work

This work draws on, and contributes to, four research areas: theoretical and empirical study of transaction fee mechanisms, multi-dimensional resource pricing, L2 fee economics, and the empirical estimation of demand elasticity under congestion pricing.

11.1 Transaction Fee Mechanism Design

Roughgarden's analysis of EIP-1559 [26, 25] establishes the design objectives of the protocol's base-fee adjustment rule and characterizes its incentive properties. Ferreira et al. [13] develop a class of dynamic posted-price mechanisms for blockspace, and Chung and Shi [7] provide a unified treatment of impossibility results for off-chain incentive-compatible transaction fee mechanisms. Leonardos et al. [18] give a control-theoretic characterization of base-fee dynamics under stochastic demand. The original proposal [6] motivates the mechanism in terms of wallet-level fee predictability, the property that underpins the present empirical strategy: predictable fees make the lagged base fee a relevant instrument for the current fee.

11.2 Empirical Studies of Gas Fees and Demand

Reijsbergen et al. [24] examine the first month after EIP-1559 deployment and document a substantial reduction in fee volatility with limited change in median fees. Liu et al. [19] extend this horizon and quantify the effect of the mechanism on waiting times and consensus security. Pierro and Rocha [23] provide an earlier empirical decomposition of gas-fee covariates including pending transaction count and exogenous costs. The closest predecessor to the present elasticity estimate is Donmez and Karaivanov [11], who use block-level data from the pre-EIP-1559 first-price regime and a queueing-theoretic specification to identify a strongly nonlinear utilization-price relationship, with sharp fee response above 90% block fullness. Their analysis predates EIP-1559, does not exploit wallet-level variation, and does not separate intensive from extensive margins; the present work addresses each of these in the post-EIP-1559 regime and additionally provides per-resource estimates for an L2. Silva [27] studies multi-resource demand elasticities on Ethereum, complementing our data collection; the focus is on the effects on per-resource demand as the Ethereum gas target increases.

11.3 Multi-Dimensional Resource Pricing

Diamandis et al. [10] introduce multi-dimensional EIP-1559-style pricing for non-fungible blockchain resources and characterize the equilibrium fee schedule. Angeris et al. [1] prove that multi-dimensional fees are essentially optimal in a broad class of demand models. Lavee et al. [17] and Kiayias et al. [16] compare the throughput and convergence properties of multi-dimensional versus single-dimensional pricing: Lavee et al. study stationary throughput and find that the ranking depends on demand functions, while Kiayias et al. take a dynamic approach and argue that a single-dimensional mechanism may reach equilibrium prices faster. Crapis et al. [9] study optimal design of a multi-dimensional pricing mechanism when resource usages are pairwise correlated.

11.4 Rollup Economics and MEV on L1/L2

Monnot [22] and Crapis [8] provide foundational economic analyses of rollup fee markets. Crapis et al. [32] study optimal strategies of L2 data posting on L1 and effects on end users.

Solmaz et al. [28] empirically find that a large percentage of L2 gas usage is due to optimistic MEV, which is not observed on L1 since the block builders can and do filter out reverted transactions. This phenomenon is theoretically studied in Mazorra et al. [21] and Wang et al. [33], where a similar conclusion is made: given fixed gas costs, probabilistic backrunning searchers will spend all expected value in reverted transactions (sometimes classified as spam).

11.5 Wallet Identification and Behavioral Clustering

In an account-based blockchain such as Ethereum, identifying wallet type is a prerequisite for any heterogeneity analysis. Victor [31] introduces the standard address-clustering heuristics for Ethereum. Béres et al. [4] extend this with timing- and graph-based features for finer profiling of addresses. Hu et al. [15] provide recent feature-engineering work specifically aimed at separating financial bots and MEV searchers from human users. The clustering procedure in Section 5 adopts features in this tradition (transaction frequency, gas variance, and high-fee timing) and augments them with per-resource consumption fractions for the multi-dimensional analysis.

11.6 Demand Elasticity under Congestion Pricing

The methodological framework used to estimate elasticity in this work has direct precedence in the electricity pricing literature. Borenstein [5] establishes that even small demand elasticities (around -0.025, a range comparable to the L1 elasticity estimate) translate to substantial welfare gains under dynamic pricing. Fabra et al. [12] use household-level panel data and a renewable-supply instrument to estimate elasticity under Spain's real-time pricing tariff, providing the closest methodological analogue to our wallet-level panel analysis. Tiedemann et al. [30] show that standard IV is inconsistent when high-frequency time series exhibit autocorrelation in both price and demand, informing the within-period exclusion concerns discussed in Section 4.3. The foundations of IV analysis are described in [2, 29, 34, 3].

12 Conclusion

We estimate the IV elasticities of gas demand for Ethereum L1 and Arbitrum L2. Gas demand is inelastic on both networks: pooled IV estimates of -0.006 (L1) and -0.036 (L2) imply that a 10% fee increase reduces total demand by at most 0.4%. Within L2, per-resource decomposition reveals that the aggregate inelasticity masks substantial resource-level heterogeneity: refunds and new storage slot creation respond meaningfully to price (-0.27 and -0.15 respectively), while computation is near-inelastic (-0.03).

References

1 Guillermo Angeris, Theo Diamandis, and Ciamac Moallemi. Multidimensional Blockchain Fees Are (Essentially) Optimal. In *7th Conference on Advances in Financial Technologies (AFT 2025)*, volume 354 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 24:1–24:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.AFT.2025.24.

2 Joshua D. Angrist and Alan B. Krueger. Does Compulsory School Attendance Affect Schooling and Earnings? *The Quarterly Journal of Economics*, 106(4):979–1014, 1991. doi:10.2307/2937954.

3 Joshua D. Angrist and Jörn-Steffen Pischke. *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton University Press, 2009.

4 Ferenc Béres, István András Seres, András A. Benczúr, and Mikerah Quintyne-Collins. Blockchain Is Watching You: Profiling and Deanonymizing Ethereum Users. In *IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, pages 69–78, 2021. arXiv:2005.14051.

5 Severin Borenstein. The Long-Run Efficiency of Real-Time Electricity Pricing. *The Energy Journal*, 26(3):93–116, 2005.

6 Vitalik Buterin, Eric Conner, Rick Dudley, Matthew Slipper, Ian Norden, and Abdelhamid Bakhta. EIP-1559: Fee Market Change for ETH 1.0 Chain. Ethereum Improvement Proposal, 2019. URL: https://eips.ethereum.org/EIPS/eip-1559.

7 Hao Chung and Elaine Shi. Foundations of transaction fee mechanism design. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2023)*, 2023. arXiv:2111.03151.

8 Davide Crapis. Rollups are Real: Rollup Economics 2.0. Notion, 2023. URL: https://davidecrapis.notion.site/Rollups-are-Real-Rollup-Economics-2-0-2516079f62a745b598133a101ba5a3de.

9 Davide Crapis, Ciamac C. Moallemi, and Shouqiao Wang. Optimal dynamic fees for blockchain resources. In Jeremy Clark and Elaine Shi, editors, *Financial Cryptography and Data Security - 28th International Conference, FC 2024, Willemstad, Curaçao, March 4-8, 2024, Revised Selected Papers, Part I*, Lecture Notes in Computer Science, pages 271–291. Springer, 2024. doi:10.1007/978-3-031-78676-1_15.

10 Theo Diamandis, Alex Evans, Tarun Chitra, and Guillermo Angeris. Dynamic Pricing for Non-Fungible Resources: Designing Multidimensional Blockchain Fee Markets. In *5th Conference on Advances in Financial Technologies (AFT 2023)*, volume 282 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 4:1–4:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.AFT.2023.4.

11 Anil Donmez and Alexander Karaivanov. Transaction Fee Economics in the Ethereum Blockchain. *Economic Inquiry*, 60(1):265–292, 2022. doi:10.1111/ecin.13025.

12 Natalia Fabra, David Rapson, Mar Reguant, and Jingyuan Wang. Estimating the Elasticity to Real-Time Pricing: Evidence from the Spanish Electricity Market. *AEA Papers and Proceedings*, 111:425–429, 2021. doi:10.1257/pandp.20211007.

13 Matheus V.X. Ferreira, Daniel J. Moroz, David C. Parkes, and Mitchell Stern. Dynamic posted-price mechanisms for the blockchain transaction-fee market. In *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, 2021. doi:10.1145/3479722.3480993.

14 Ragnar Frisch and Frederick V. Waugh. Partial time regressions as compared with individual trends. *Econometrica*, 1(4):387–401, 1933. doi:10.2307/1907330.

15 Hanzhi Hu, Tom Beer, and Daniel Perez. Detecting Financial Bots on the Ethereum Blockchain. *CoRR*, abs/2403.19530, 2024. doi:10.48550/arXiv.2403.19530.

16 Aggelos Kiayias, Elias Koutsoupias, Giorgos Panagiotakos, and Kyriaki Zioga. One-dimensional vs. multi-dimensional pricing in blockchain protocols. *CoRR*, abs/2506.13271, 2025. doi:10.48550/arXiv.2506.13271.

17 Nir Lavee, Noam Nisan, Mallesh M. Pai, and Max Resnick. Does your blockchain need multidimensional transaction fees? *CoRR*, abs/2504.15438, 2025. doi:10.48550/arXiv.2504.15438.

18 Stefanos Leonardos, Barnabé Monnot, Danijar Reijsbergen, Efstratios Skoulakis, and Georgios Piliouras. Dynamical Analysis of the EIP-1559 Ethereum Fee Market. *CoRR*, abs/2102.10567, 2021. arXiv:2102.10567.

19 Yulin Liu, Yuxuan Lu, Kartik Nayak, Fan Zhang, Luyao Zhang, and Yinhong Zhao. Empirical Analysis of EIP-1559: Transaction Fees, Waiting Times, and Consensus Security. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022. doi:10.1145/3548606.3559341.

20 Michael C. Lovell. Seasonal adjustment of economic time series and multiple regression analysis. *Journal of the American Statistical Association*, 58(304):993–1010, 1963. doi:10.1080/01621459.1963.10480682.

21 Bruno Mazorra, Christoph Schlegel, and Akaki Mamageishvili. Timing games: Probabilistic backrunning and spam. *CoRR*, abs/2602.22032, 2026. doi:10.48550/arXiv.2602.22032.

22 Barnabé Monnot. Understanding Rollup Economics from First Principles. Substack, 2022. URL: https://barnabe.substack.com/p/understanding-rollup-economics-from.

23 Giuseppe Antonio Pierro and Henrique Rocha. The Influence Factors on Ethereum Transaction Fees. In *2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, pages 24–31, 2019. doi:10.1109/WETSEB.2019.00010.

24 Danijar Reijsbergen, Stefanos Leonardos, Barnabé Monnot, Stratis Skoulakis, and Georgios Piliouras. Transaction Fees on a Honeymoon: Ethereum's EIP-1559 One Month Later. In *2021 IEEE International Conference on Blockchain (Blockchain)*, 2021. doi:10.1109/Blockchain53845.2021.00049.

25 Tim Roughgarden. Transaction Fee Mechanism Design. In *Proceedings of the 22nd ACM Conference on Economics and Computation (EC 2021)*, 2021. doi:10.1145/3465456.3467660.

26 Tim Roughgarden. Transaction Fee Mechanism Design for the Ethereum Blockchain: An Economic Analysis of EIP-1559. *Cryptoeconomic Systems*, 2021. arXiv:2012.00854.

27 Maria Silva. Empirical Analysis of Price Elasticities for Ethereum State and Burst Resources. Ethereum Research, 2026. URL: https://ethresear.ch/t/empirical-analysis-of-price-elasticities-for-ethereum-state-and-burst-resources/24166.

28 Ozan Solmaz, Lioba Heimbach, Yann Vonlanthen, and Roger Wattenhofer. Optimistic MEV in ethereum layer 2s: Why blockspace is always in demand. In Zeta Avarikioti and Nicolas Christin, editors, *7th Conference on Advances in Financial Technologies, AFT 2025, Pittsburgh, PA, USA, October 8-10, 2025*, LIPIcs, pages 28:1–28:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.AFT.2025.28.

29 Douglas Staiger and James H. Stock. Instrumental Variables Regression with Weak Instruments. *Econometrica*, 65(3):557–586, 1997. doi:10.2307/2171753.

30 Silvana Tiedemann, Raffaele Sgarlato, and Lion Hirth. Price Elasticity of Electricity Demand: Using Instrumental Variable Regressions to Address Endogeneity and Autocorrelation of High-Frequency Time Series. *Energy Economics*, 2024. arXiv:2306.12863.

31 Friedhelm Victor. Address Clustering Heuristics for Ethereum. In *Financial Cryptography and Data Security (FC) 2020*, volume 12059 of *Lecture Notes in Computer Science*, pages 617–633. Springer, 2020. doi:10.1007/978-3-030-51280-4_33.

32 Shouqiao Wang, Davide Crapis, and Ciamac C. Moallemi. A framework for combined transaction posting and pricing for layer 2 blockchains. In Christina Garman and Pedro Moreno-Sanchez, editors, *Financial Cryptography and Data Security - 29th International Conference, FC 2025, Miyakojima, Japan, April 14-18, 2025, Revised Selected Papers, Part I*, Lecture Notes in Computer Science, pages 56–72. Springer, 2025. doi:10.1007/978-3-032-07024-1_4.

33 Wenhao Wang, Aditya Saraf, Lioba Heimbach, Kushal Babel, and Fan Zhang. Blockspace under pressure: An analysis of spam MEV on high-throughput blockchains. *CoRR*, abs/2604.00234, 2026. doi:10.48550/arXiv.2604.00234.

34 Jeffrey M. Wooldridge. *Econometric Analysis of Cross Section and Panel Data*. MIT Press, 2nd edition, 2010.

Who Loses and Why? Market Microstructure of Perpetual Futures

Ruizhe Jia
Stanford University, CA, USA

Ciamac C. Moallemi
Columbia Business School, Columbia University, New York, NY, USA

Shouqiao Wang
Columbia Business School, Columbia University, New York, NY, USA

Xinmeng Zeng
Stanford University, CA, USA

Abstract

This extended abstract summarizes the essence of [1], which asks who loses money in perpetual-futures trading, and why. We study Hyperliquid, a leading decentralized perpetual-futures venue, using the complete on-chain record of orders and trades from March 27, 2025 through June 30, 2026 for its ten most actively traded contracts. We classify wallets into algorithmic market makers, front-end traders, and algorithmic takers; front-end traders trade manually through the venue's interface, and we interpret them as predominantly retail. For each group we then measure profit and loss trade by trade, valuing every trade against the midprice two hours later and subtracting the fees paid. On this measure front-end traders lose $1.11 billion, market makers gain $208 million, and algorithmic takers approximately break even. Three channels carry the transfer and account for the whole of it. The first is voluntary trading. Front-end traders demand immediacy, placing large aggressive orders whose size scales little with displayed liquidity, and so pay the highest execution costs; favorable post-trade price movement and maker income offset part of that cost. The second is forced exits. Leverage converts adverse price moves into forced liquidations, which exit at prices far worse than any voluntary trade, and auto-deleveraging returns only part of the loss. The third is fees. Front-end traders cross the spread on most of their volume and sit in the highest-fee tiers, and so pay most of the venue's fees. Post-trade price impact is organized by trade purpose, though group differences persist within each purpose category. These results identify immediacy, forced exits, and fees as the principal channels of wealth transfer in decentralized perpetual-futures markets.

2012 ACM Subject Classification Applied computing → Economics

Keywords and phrases Perpetual futures, cryptocurrency derivatives, decentralized exchanges, on-chain data, retail trading, market microstructure, execution costs, liquidations

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.36

Category Extended Abstract

Related Version *Full Version*: `https://ssrn.com/abstract=7270041` [1]

Funding This research was supported by one year of funding from the Input Output Research Hub (IORH) at Stanford University, funded by Input Output Global.

Acknowledgements We thank *Allium* for providing the data used in this study.

1 Motivation

Perpetual futures are derivative contracts with no fixed maturity; periodic funding payments between long and short positions keep their prices close to spot. They are now the dominant instrument in cryptocurrency trading, with reported 2025 notional volume of $61.7 trillion,



© Ruizhe Jia, Ciamac C. Moallemi, Shouqiao Wang, and Xinmeng Zeng;
licensed under Creative Commons License CC-BY 4.0



more than three times spot volume,[1] yet most of it occurs on offshore venues outside regulated reporting, leaving even basic facts about market participants unknown. The visible portion is already large: reported forced liquidations across crypto derivatives reached roughly $150 billion in 2025, more than $19 billion of it on October 10, 2025 alone.[2] These episodes point to large wealth transfers, but not to their economic structure. In [1] we ask who is on each side of these transfers, how large they are, and through which mechanisms they occur.

2 Setting and Data

Hyperliquid is a leading decentralized perpetual-futures exchange whose order book and matching engine run entirely on a public blockchain. Deposits, withdrawals, account equity, and margin usage are recorded in on-chain snapshots for each wallet address, and all orders, cancellations, trades, and forced liquidations are recorded at millisecond resolution with trade-time position size and realized profit and loss, so each wallet's complete event history is observable. Our sample runs from March 27, 2025 through June 30, 2026 and covers the ten perpetual contracts with the highest trading volume; order and fill records come from Allium, top-of-book snapshots from Tardis. In total, 729,540 active wallets are classified.

3 Empirical Approach

We classify pseudonymous wallets into economically interpretable trader types based on observed trading behavior, assigning labels at the coin-day level, so labels can vary across coins and over time:

- **Algorithmic market makers** continuously post, update, and cancel resting limit orders on both sides of the book, with most submissions add-liquidity-only (post-only).
- **Algorithmic takers** exhibit high-frequency activity, large order counts, and sustained trading over hours incompatible with manual trading; 69% of their volume takes liquidity.
- **Front-end traders** are identified primarily by market orders submitted through the venue's web front end, which Hyperliquid records as `FrontendMarket` orders; we interpret this group as predominantly retail.

A residual ambiguous group, 1.7% of platform volume, is excluded from group-level analysis.

We then measure each group's dollar profit and loss trade by trade, valuing every trade against the midprice 120 minutes later, with fees subtracted separately. This profit and loss splits into three channels that add to the total. Voluntary trading nets execution cost, post-trade price impact, and income earned as a maker. Net forced exits set losses on forced liquidations against the offset from auto-deleveraging (ADL), which closes profitable positions at the mark price when liquidations cannot clear. Fees are the third channel.

Finally, we partition every voluntary taker trade, an aggressive order that crosses the spread, by its trade purpose: an opening increases position size, a gain-realizing close reduces a position at a profit, and a loss-realizing close reduces a position at a loss.

4 Key Findings

1. **Front-end traders finance the venue.** Measured trade by trade at the 120-minute interval and net of fees, front-end traders lose $1.11 billion, market makers gain $208 million, and algorithmic takers approximately break even. The front-end loss is the sum

[1] See Reuters, April 22, 2026, citing CryptoQuant.
[2] See CryptoSlate (CoinGlass data) and CoinDesk, October 11, 2025.

of the three channels: $154 million on voluntary trading, $226 million on net forced exits, and $725 million in fees. The gap between the front-end loss and the other groups' gains is mostly the fee bill, which accrues to the venue rather than to the other trading groups.

2. **Channel one, voluntary trading: $154 million.** Front-end traders pay $501 million in execution cost on their voluntary taker trades, recover $299 million through favorable post-trade price movement, and earn $48 million as makers. Execution cost is 2.43 basis points per dollar of taker volume, against 0.70 for market makers and 1.70 for algorithmic takers. The gap comes from slippage: front-end traders concentrate volume in large aggressive orders, where slippage rises sharply, and scale trade size with displayed liquidity far less than market makers do.
3. **Inside that channel: post-trade price impact and trade purpose.** Price impact initially favors all three groups but subsequently declines only for front-end traders. Trade purpose organizes the divergence: loss-realizing closes follow recent returns and revert for every group, gain-realizing closes are contrarian for every group, and only openings differ, with front-end traders opening in the direction of recent returns while the other two groups open against them. A loss-realizing close is the voluntary counterpart of a forced exit: both exit quickly and pay for immediacy, and the post-trade price pattern on a liquidation resembles that of a loss-realizing close on a much larger scale.
4. **Channel two, net forced exits: $226 million.** Forced liquidations impose a $344 million loss on front-end traders, against which auto-deleveraging returns $118 million. Front-end traders account for 96.0% of three-group liquidation volume (97.6% excluding the October 10, 2025 episode). Liquidation trades pay roughly twenty times the voluntary half-spread, with adverse post-trade price moves an order of magnitude larger.
5. **Channel three, fees: $725 million.** That is 2.99 basis points of front-end volume, compared with $50 million (0.16 basis points) for market makers and $55 million (2.17 basis points) for algorithmic takers. Front-end traders trade overwhelmingly as takers, crossing the spread on 85% of their volume versus 20% for market makers; and because fee tiers depend on recent trading volume and HYPE staking, most front-end volume sits in the base fee tier while market-maker volume is concentrated in lower-fee tiers.

5 Implications

None of these mechanisms requires trading against better-informed counterparties in the usual sense. Front-end traders pay because they demand immediacy in large aggressive orders without conditioning size on displayed depth as market makers do, because the maker–taker schedule charges the aggressor and they take liquidity on 85% of their volume, and because leverage converts adverse price moves into forced exits at prices far worse than any voluntary trade. Funding payments, the channel specific to perpetual contracts, sit outside this measure: negligible over a two-hour window, they matter only as a carrying cost on positions held longer. The results give a wallet-level account of who loses, and why, on a leading decentralized venue: front-end traders pay for immediacy, in larger aggressive trades, in costly forced exits, and in less favorable fee tiers.

References

1 Ruizhe Jia, Ciamac C. Moallemi, Shouqiao Wang, and Xinmeng Zeng. Who loses and why? Market microstructure of perpetual futures, 2026. doi:10.2139/ssrn.7270041.