Mechanism Design for Automated Market Makers

T-H. Hubert Chan[1]
University of Hong Kong, Hong Kong

Ke Wu
University of Michigan, Ann Arbor, MI, USA
Carnegie Mellon University, Pittsburgh, PA, USA

Elaine Shi
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

Blockchains have popularized automated market makers (AMMs), applications that run on a blockchain, maintain a pool of crypto-assets, and execute trades with users governed by some pricing function. AMMs have also introduced a significant challenge known as the Miner Extractable Value (MEV). Specifically, miners who control the contents and sequencing of transactions in a block can extract value by front-running and back-running users' transactions, creating arbitrage opportunities that guarantee them risk-free returns. MEV not only harms ordinary users, but more critically, encourages miners to auction off favorable transaction placements to users and arbitragers. This has fostered a more centralized off-chain eco-system, departing from the decentralized equilibrium originally envisioned for the blockchain infrastructure layer.

In this paper, we consider how to design AMM mechanisms that eliminate MEV opportunities. Specifically, we propose a new AMM mechanism that processes all transactions contained within a block according to some pre-defined rules, ensuring that some constant potential function is maintained after processing the batch. We show that our new mechanism satisfies two tiers of guarantees. First, for legacy blockchains where each block is proposed by a single (possibly rotating) miner, we prove that our mechanism satisfies arbitrage resilience, i.e., a miner cannot gain risk-free profit. Second, for blockchains where the block proposal process is decentralized and offers sequencing-fairness, we prove a strictly stronger notion called strategy proofness – roughly speaking, we guarantee that any individual user's best response is to follow the honest strategy.

Our results complement prior works on MEV resilience in the following senses. First, prior works have shown impossibilities to address MEV entirely at the consensus level. Our work demonstrates a new paradigm of mechanism design at the application (i.e., smart contract) layer to ensure provable guarantees of strategy proofness. Second, many works have attempted to augment the underlying consensus protocol with extra properties such as sequencing fairness. While most previous works heuristically argued why these extra properties help to mitigate MEV, our work demonstrates in a mathematically formal manner how to leverage such consensus-level properties to aid the design of strategy-proof mechanisms.

2012 ACM Subject Classification Theory of computation → Algorithmic mechanism design; Theory of computation → Algorithmic game theory

Keywords and phrases Mechanism design, game theory, strategy proof, blockchain

Digital Object Identifier 10.4230/LIPIcs.AFT.2025.7

Related Version *Full Version*: `https://arxiv.org/abs/2402.09357` [25]

Funding *T-H. Hubert Chan*: T-H. Hubert Chan was partially supported by the Hong Kong RGC grants 17203122 and 17202121.
Ke Wu: Ke contributed to this work while she was a Ph.D. student at CMU. See the grant acknowledgements of Elaine Shi.

[1] Author ordering is randomized.

Elaine Shi: This work is in part supported by NSF awards 2212746, 2044679, 1704788, a Packard Fellowship, a generous gift from the late Nikolai Mushegian, a gift from Google, and an ACE center grant from Algorand Foundation.

1 Introduction

Blockchains have popularized decentralized finance (DeFi), with one of its key applications being Decentralized Exchanges (DEX) based on Automatic Market Makers (AMMs) [14]. As of March 2021, the top six AMMs, including Uniswap, Balancer, and others, collectively held approximately \$15 billion in crypto assets [46]. A typical AMM exchange maintains a pool of capital called the "liquidity pool" with two crypto-assets X and Y. A smart contract specifies the rules how users can trade assets with the pool. For example, one commonly adopted rule is a *constant-product potential function* defined as follows. Let $\mathsf{Pool}(x, y)$ denote the pool's state where $x \geq 0$ and $y \geq 0$ represent the units of X and Y held by the pool, respectively. A constant product potential requires that $x \cdot y = C$ for some constant $C > 0$. This means that if a user buys δx amount of X from the pool, it needs to pay $-\delta y$ amount of Y such that $(x - \delta x)(y - \delta y) = C$.

DeFi applications such as AMMs have introduced opportunities for miners to profit, often in a risk-free manner, by front-running and/or back-running the users' transactions, a phenomenon known as Miner Extractable Value (MEV). Despite the decentralized nature of blockchains, the block proposal process in mainstream consensus protocols remains centralized. For each block, a single selected miner[2] has unilateral control over which transactions are included and their sequencing. By exploiting this capability, miners can profit, often in a risk-free manner. For example, in a sandwich attack [37, 41, 46, 51], a miner identifies a victim user attempting to purchase a crypto asset X at a maximum price of r, and inserts a $\mathsf{Buy}(X)$ transaction just before the victim's buy order and a $\mathsf{Sell}(X)$ transaction immediately after. Since purchasing X increases its price, the miner effectively buys at a lower price through front-running, forces the victim to buy at the worst possible price r, and then sells at a higher price through back-running, locking in a profit. Beyond sandwich attacks, miners can also take advantage of more sophisticated arbitrage opportunities to profit [37, 41, 49].

MEV is widely recognized as one of the most important challenges for blockchains today for several reasons. First, since MEV is extracted at the expense of users, it effectively increases the barrier of entry for ordinary users to engage with DeFi applications. Second, MEV undermines the stability and security of the underlying consensus protocol [24, 41, 50, 51]. Specifically, miners may be incentivized to fork the blockchain if doing so offers higher MEV rewards than standard block rewards. Third, the block producer's power in deciding the block contents and sequencing has given rise to an off-chain economy. Block producers enter private contracts with arbitragers and users alike, offering them favorable positions in the block. These private off-chain contracts have led to a centralizing effect in the underlying layer 1 (i.e., the consensus layer), causing the *de facto* layer 1 [31] to operate in a manner that significantly departs from the intended design, and its equilibrium behavior is not understood. A recent empirical measurement showed that today, more than 85% of the Ethereum blocks are built by two block producers [47].

[2] In this paper, no matter whether the underlying consensus is proof-of-stake or proof-of-work, we generically refer to a consensus node that produces blocks as a "miner" or a "block producer".

1.1 Our Results and Contributions

It would have been compelling if there existed a way to solve the MEV problem entirely at the consensus layer, without having to modify the existing applications. Unfortunately, the impossibility results in several previous works [13, 30] can be interpreted to mean that solving the MEV problem (in its most general form) entirely at the consensus layer, subject to today's architecture, is impossible. On the other hand, many works aimed to offer strengthened guarantees at the consensus layer such as sequencing fairness [2, 10, 34–36] or some form of privacy [17, 29]. While it is widely believed that these properties help to mitigate MEV, there has been relatively little formal investigation on how we can take advantage of these extra consensus-level properties in mechanism design.

Therefore, in this paper, we ask the following natural questions:

- Instead of working entirely at the consensus level, can we rely on mechanism design at the application level (i.e., smart contract level) to obtain provable guarantees of MEV resilience?
- How do strengthened guarantees at the consensus layer aid the design of strategy-proof mechanisms at the application layer?

Specifically, we ask these questions in the context of Automated Market Makers (AMMs) which represent one of the most important DeFi applications. We now summarize our results and contributions.

A mechanism design approach towards mitigating MEV. Applying the philosophy of mechanism design at the application layer, we want to design an AMM mechanism that removes MEV opportunities and provides strategy proofness by construction.

We devise a new AMM mechanism (to be executed as a smart contract on chain) with the following abstraction. In our mechanism, the pool holds two crypto-assets X and Y. A user can trade with the pool by posting a buy/sell *order* specifying how much of X (or Y) they want to buy/sell, and their worst acceptable exchange rate. When a new block arrives, the mechanism takes the block of orders as input, and applies an allocation rule to all orders contained within the block. The allocation rule decides which orders are partially or completely satisfied, and at what price. The mechanism maintains the following invariant: the pool's beginning state denoted $\mathsf{Pool}(x_0, y_0)$ and end state $\mathsf{Pool}(x_1, y_1)$ is guaranteed to satisfy some "natural" potential function Φ (e.g., the constant-product function mentioned above).

Our mechanism offers two tiers of guarantees depending on whether the properties of the underlying consensus. Specifically, we consider two models: 1) the *plain* model, capturing today's mainstream consensus protocols where for any particular block, the inclusion and sequencing of transactions are determined by a single, possibly strategic block producer; and 2) the *weak fair-sequencing* model, intended to capture a new generation of consensus protocols that offer sequencing fairness guarantees [2, 10, 34–36], e.g., through a decentralized sequencer. Our mechanism achieves the following desirable, two-tier properties:

1. **Arbitrage free** in the plain model. We guarantee that no arbitrager (e.g., user, block producer, or any intermediary) can gain risk-free profit, *even when the arbitrager (e.g., block producer) has unilateral control over the block contents and transaction sequencing.* Here, risk-free profit happens when an arbitrager can gain in one asset without losing in another with probability 1.
2. **Strategy proofness** in the weak fair-sequencing model. In the weak fair-sequencing model, our mechanism not only achieves arbitrage resilience, but also guarantees strategy proofness. Specifically, strategy proofness means that users are incentivized to report

their true demand and true belief of the relative value of the two crypto-assets, and no strategic behavior allows a user to gain. Later, we prove that *strategy proofness is a strictly stronger notion than arbitrage resilience* (Fact 1).

Our weak fair-sequencing model is meant to capture a decentralized sequencer that sequences the orders based on their arrival times (importantly, not based on the orders' submission time). While this model places additional constraints on the strategy space in comparison with the plain model, it **does not prevent front-running** and thus **does not trivialize** the mechanism design problem. Notably, in this model, a strategic user or miner can still wait for a victim to submit its order, and then immediately submit a dependent order. The strategic order can even front-run the victim's order if the strategic user's network is faster, similar to a rushing attack in the cryptography literature [19, 20].

Our work is also *among the first to formally articulate, from a mechanism designer's perspective, how extra properties at the consensus level lend to the design of strategy-proof mechanisms at the smart contract layer.*

Conceptual contributions. We put forth new modeling and definitions, which capture a mechanism design problem of a decentralized nature. In comparison with the classical mechanism design literature, our model and strategy spaces capture the "permissionless" nature of blockchains. Specifically, a strategic player may not only report its valuation/demand untruthfully, but also inject fake orders or post multiple orders. Such strategies are possible because the mechanism does not have a-priori knowledge of the number or the identities of the bidders. Compared to closely related works, our model circumvents the strong impossibilities shown by Ferreira and Parkes [30], because we relax some unrealistic restrictions they impose – see Section 1.2 for more details. Therefore, we believe that our model is better suited for capturing real-world mechanism design at the smart contract layer, particularly for AMMs. Our new model and definitions naturally give rise to many interesting open questions which we discuss in Section 1.3.

1.2 Comparison with Related Work

Comparison with most closely related work. Ferreira and Parkes [30] showed that in an overly pessimistic model as explained below, achieving arbitrage resilience is impossible, let alone strategy proofness. However, their impossibility result holds only in an overly stringent model that does not reflect the real-life design space. Specifically, when interpreted in our new framework, their impossibility holds only if the AMM mechanism has to respect the following specific structure: sort the incoming orders according to some rules (called "verifiable sequencing rules" in their paper), and run a legacy AMM contract that processes the sorted orders sequentially, such that the constant potential function must be maintained after executing each order. In comparison, in our model, the constant potential function only needs to be maintained at the end of processing the entire batch. Because of their strong impossibility result, Ferreira and Parkes [30] showed how to achieve a weaker guarantee in their model, that is, if the miner made risk-free profit, then the user should enjoy a price that is at least as good as if its order were the only one in the block.

Li et al. [39] inherit the same model as Ferreira and Parkes [30], and they study what is the profit-maximizing strategy for the miner and the implications for the users when the miner adopts the optimal strategy. Like Ferreira and Parkes [30], they adopt an overly restrictive model which requires them to give up on achieving arbitrage resilience, let alone strategy proofness.

One of the contributions we make is exactly to recognize why the existing models are too stringent and unrealistic, and suggest a better model for the study of MEV-resilient mechanism design.

Batch clearing at uniform price does not guarantee strategy proofness. A line of works explored the idea of batch clearing at uniform price. We stress that batch clearing at uniform price [1, 21, 22] does not automatically guarantee strategy proofness. For example, suppose there are many eligible orders and the mechanism can clear only a subset of them. If the mechanism selects the subset based on the declared valuation, then a strategic user can lie about its valuation to get selected.

The concurrent and independent work of Canidio and Fritsch [21, 22] suggested batch-clearing at a uniform price and using a different potential function than Uniswap's constant-product function. Their approach satisfies arbitrage resilience, but it does not satisfy strategy proofness *even when the miner is trusted to behave honestly*, for the reason stated above. In fact, Canidio and Fritsch [21, 22] does not even fully specify which subset of orders to clear when there are many eligible candidates. The work of Zhang et al. [48] also observed that batch clearing alone does not imply strategy proofness. In fact, they investigated the optimal strategy under batch clearing. The prior work of Ramseyer et al. [43] also considered batch exchanges that clear at a uniform price. Like Canidio and Fritsch [21, 22], their work does not provide strategy proofness, even when the miner is fully trusted. Besides batch trading at uniform pricing, other forms of batch trading [23] have also been considered, but they also do not satisfy our notion of strategy proofness.

Related works that do not address MEV. Milionis, Moallemi, and Roughgarden [40] consider how to design the demand curve for a market maker to maximize profit and meanwhile incentivize truthful reporting. Their work is of a completely different nature than ours, since they *do not aim to address the problem of MEV*. Specifically, they consider a simple model where users directly submit orders to the market maker. They *do not consider any arbitrage strategy* where users or miners try to front-run or back-run others' orders to make profit.

Bartoletti et al. [15] studied the miner's optimal MEV strategy under transaction reordering. Their work also does not provide a solution to mitigate or address MEV.

Understanding the impact of MEV. A line of works have empirically or theoretically investigated the profitability or impact of MEV [11, 12, 16, 37, 41, 42, 49, 52].

Empirical approaches towards mitigating MEV. Another line of work suggest that the users themselves take action to mitigate MEV, either by setting their slippage limits more cleverly [50], or by exploiting arbitrage opportunities themselves to lower their transactional costs [32]. There are also various blog posts on online forums that suggest alternative designs [33, 38]. However, these works are empirical and do not lend to the theoretical understanding of the equilibrium behavior of the eco-system.

Both academic research and real-world blockchain projects have made an effort to build decentralized sequencers [2, 10, 34–36], or encrypted mempools [17, 29]. The former approach removes the ability for a single block proposer to decide the block contents and sequencing, and achieves some form of sequencing fairness [34–36][3]. The latter approach allows users to

[3] Sequencing fairness is also commonly referred to as "order fairness". In this paper, we use the term

submit transactions in committed or encrypted format, which makes it harder for miners to front-run and back-run transactions. However, from a mechanism design perspective, we still lack mathematical understanding to what extent these new consensus/cryptographic abstractions can help us mitigate MEV and achieve strategy-proof DeFi mechanisms. In this sense, our work is *among the first to mathematically articulate how to rely on "sequencing fairness" to achieve strategy-proofness by construction.*

Sequencing fairness. A line of works [2, 10, 34–36] have studied how to achieve order fairness in consensus. Numerous blockchain projects are also building decentralized sequencers [3–8] which can be one approach for achieving sequencing fairness. Some works [9, 28] have also pointed out the price of sequencing fairness such as loss in welfare. Improving the underlying mechanisms for achieving sequencing fairness is outside the scope of this paper. We also leave it as future work to study how to optimize social welfare under strategy proofness.

Other related works. There is a recent line of work on transaction fee mechanism (TFM) design [26, 44, 45]. This line of work aims to design mechanisms such that users, miners, and user-miner coalitions are incentivized to behave honestly. However, the current modeling approach of this line of work captures only the utilities at the consensus layer. They cannot capture ordering and application-level MEV. The recent work of Bahrani et al. [13] showed strong impossibility results for fully solving this problem at the TFM-layer alone. In this sense, our work complements the line of work on TFM design by taking an application-level (i.e., smart-contract-level) approach towards achieving strategy proofness by construction.

1.3 Scope and Open Questions

Just like the recent literature including Ferreira and Parkes [30] and Li et al. [39], the scope of the present paper is restricted to how defend against MEV in a *standalone* two-asset AMM mechanism. We begin with the standalone setting because it serves as a necessary basis for understanding the compositional setting with multiple instances. It is also a widely adopted approach in the mechanism design and cryptography literature to begin with the standalone setting first. We leave it as an interesting open question how to achieve provable game-theoretic guarantees in a compositional setting where multiple instances can interact with each other. We stress that in general, unlike in the cryptography literature where there are composable notions of security [19, 20] which makes composition worry-free, most notions in game theory do not naturally compose, and composition is typically treated on a case-by-case basis.

Our new model and definitions give rise to many interesting open problems. One interesting question is how to extend our results to AMMs with multiple assets. Another interesting question is whether it is possible to achieve the stronger notion of strategy proofness without relying on sequencing fairness. Currently, our model assumes that all the orders are submitted in the clear, and we thus define strategy proofness in the ex post setting. A future direction is to understand how to define and achieve strategy proofness in an MPC-assisted model [45] or an "encrypted mempool" [17, 29] model where transactions are submitted in encrypted or committed format. Another interesting question is whether we can design strategy-proof AMM mechanisms when the execution syntax is atomic rather than partial fulfillment like what we consider.

"sequencing fairness" to avoid collision with the usage of "order" to mean a trade proposal.

We believe that the modeling work in our paper helps to lay the theoretical groundwork for exploring questions such as above in the future.

2 Definitions

2.1 Swap Mechanism for AMMs

A swap mechanism for a pair of assets (X, Y) has a state (also called the *pool state*) denoted $\mathsf{Pool}(x, y)$ where x and y are non-negative values that represent the amount of each asset currently held by the mechanism. A user can submit an order to trade with the mechanism in two ways: either buy X and pay in Y, or buy Y and pay in X. Suppose the user buys δx units of X and pays δy units of Y, then the updated state after the trade will become $\mathsf{Pool}(x - \delta x, y + \delta y)$.

Order. Each order is of the form (t, v, r, α) where

- $t \in \{\mathsf{Buy}(X), \mathsf{Buy}(Y), \mathsf{Sell}(X), \mathsf{Sell}(Y)\}$ is the type of the order indicating that the user wants to buy or sell and which asset;
- v is a non-negative value that denotes the maximum amount of the user wants to buy or sell;
- r denotes the user's acceptable exchange rate, i.e., the user believes that each unit of X is worth r units of Y. For example, if the order is of type $\mathsf{Buy}(X)$, then the user is willing to pay at most r units of Y for each unit of X; if the type is $\mathsf{Sell}(Y)$, then $1/r$ is the minimum asking price in X for each unit of Y.
- α is an arbitrary string denoting any additional auxiliary information, e.g., the submitter's identity, timestamping information, position in the block, and so on.

Note that given an order of the form $(\mathsf{Sell}(Y), v, r, \_)$[4], another way to view it is that the user wants to buy X; it is willing to pay at most r units of Y for each unit of X; moreover, it wants to buy as many units of X as possible subject to a capital of v units of Y. Henceforth, for an order of the type $\mathsf{Buy}(X)$ or $\mathsf{Sell}(X)$, we say that X is the *primary asset* of the order.

Swap mechanism. A possibly randomized (partial fulfillment) swap mechanism should define the following rules[5]:

- *Honest strategy.* Given a user's private type T, the initial state $\mathsf{Pool}(x, y)$, the honest strategy, often denoted $HS(x, y, T)$, outputs a vector of orders the user should submit. A user's private type T can contain information such as how many units of X and Y it currently holds, and the user's private valuation of the exchange rate between X and Y.
- *Allocation rule.* The allocation rule receives as input an initial state $\mathsf{Pool}(x, y)$, a list of orders, and for each order (t, v, r, α), it outputs the following:
  - the amount $v' \in [0, v]$ of primary asset that has been fulfilled – note that the fulfillment can be partial;
  - an average exchange rate $r' > 0$ at which the order was fulfilled. For a $\mathsf{Buy}(X)$ order, it means that the user pays $v' \cdot r'$ units of Y in exchange for v' units of X. For a $\mathsf{Sell}(Y)$ order, the user obtains v'/r' units of X for the v' units of Y sold. We require that for a $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$ order, $r' \leq r$, i.e., the purchase price cannot be higher than the specified maximum rate r; and for a $\mathsf{Buy}(Y)/\mathsf{Sell}(X)$ order, $1/r' \leq 1/r$.

[4] Here, the ignore symbol _ means that we are ignoring the content of this field in the current context.
[5] In Section 5, we pose the open question of how to achieve strategy proofness for all-or-nothing fulfillment mechanisms.

In a real-world instantiation, the pool state is recorded on the blockchain, and the allocation rule is executed on the blockchain (e.g., in the form of a smart contract).

Invariant on pool state. We consider swap mechanisms that satisfy the following invariant on pool state. Given some initial state $\mathsf{Pool}(x, y)$, and the outcome output by the allocation rule, one can uniquely determine the ending state $\mathsf{Pool}(x', y')$. We require that initial and ending pool state must satisfy some constant potential function, that is, $\Phi(x', y') = \Phi(x, y)$. We define potential function and requirements on the potential function below.

Potential function. We consider swap mechanisms that respect a constant potential function $\Phi(\cdot, \cdot)$. Specifically, suppose the pool's initial state is $\mathsf{Pool}(x, y)$, and changes to $\mathsf{Pool}(x', y')$ after the mechanism processes a batch of orders. Then, it must be that

$$\Phi(x, y) = \Phi(x', y').$$

In practice, the most widely adopted approach is a constant-product market maker where $\Phi(x, y) = x \cdot y$. In other words, suppose the initial pool state is (x, y) and some user buys δx amount of X, then it must pay $-\delta y$ units of Y where δy can be calculated by solving the following equation:

$$(x - \delta x)(y - \delta y) = xy.$$

Assumptions on the potential function. We assume the standard assumption that the potential function $\Phi(\cdot, \cdot)$ is increasing, differentiable, and concave.

Market exchange rate. In our swap mechanism, we will make use of the notion of a market exchange rate, as defined below.

▶ **Definition 2.1** (Market exchange rate). *Given a pool state* $\mathsf{Pool}(x, y)$, *the current market exchange rate is defined as*

$$r(x, y) = \frac{\partial \Phi / \partial x}{\partial \Phi / \partial y}(x, y).$$

Intuitively, it means that to buy an infinitesimally small dx amount of X, we need to pay $r(x, y) \cdot dx$ units of y.

Throughout this paper, whenever we say rate, it always means how much y one has to pay per unit of x rather than the other way around.

2.2 Arbitrage Resilience

Arbitrage resilience means that an arbitrager has no strategy such that it gets a net gain in one asset without any loss in the other.

▶ **Definition 2.2** (Arbitrage resilience). *We say that a mechanism satisfies arbitrage resilience iff given any initial pool state, any input vector of orders, with probability* 1 *over the random coins of the mechanism, the following must hold: there does not exist a subset of orders whose joint outcomes result in $\delta x \geq 0$ net gain in X and $\delta y \geq 0$ net gain in Y, such that at least one of δx and δy is strictly greater than* 0.

The definition above is consistent with Ferreira and Parkes [30]'s notion of no risk-free return, although their scheme cannot guarantee no risk-free return (i.e., arbitrary resilience), whereas ours does.

▶ Remark 2.3. If a mechanism satisfies Definition 2.2, it means that it satisfies arbitrage resilience in a very strong sense: i.e., an arbitrager (e.g., block producer) cannot make risk-free profit *even when it can 1) fully control the block contents; 2) control the sequencing of orders within the block; 3) inject its own orders; and 4) drop others' orders.*

Indeed, the mechanism described later in this paper will satisfy arbitrage resilience even when the underlying consensus does not provide any sequencing fairness guarantees.

2.3 Plain Model

The plain model is meant to capture mainstream consensus protocol today where the contents of each individual block is determined by a single block producer responsible for proposing or mining that block.

Strategy space in the plain model. In the plain model, we assume that a strategy user or miner with intrinsic type (t, v, r, α) may engage in the following strategies:

- Post zero or multiple arbitrary orders which may or may not reflect its intrinsic type – this captures strategies that involve misreporting valuation and demand, as well as posting of fake orders;
- Censor honest users' orders – this captures a strategic miner's ability to exclude certain orders from the assembled block;
- Arbitrarily misrepresent its own auxiliary information field α, or even modify the α field of honest users' orders – meant to capture the ability of a miner to decide the sequencing of the transactions within a block, where the arrival-time and position information may be generically captured by the auxiliary information field α.
- Decide its strategy *after* having observed honest users' orders.

The coalition of a miner with 0 demand and a user with some positive demand as captured by its type $(t, v > 0, r, \alpha)$ can simply be viewed as a single strategic player with type $(t, v > 0, r, \alpha)$.

2.4 Weak Fair-Sequencing Model

We define a weak fair-sequencing model, meant to capture a new generation of consensus protocols that employ a decentralized sequencer and offers some form of sequencing fairness [2, 10, 34–36]. Such a decentralized sequencer will sequence the transactions based on their (approximate) arrival times. We stress that even in the weak fair-sequencing model, it is possible for a strategic user to observe a victim's order, post a dependent order, and have the dependent order race against and front-run the victim's order. Such a front-running attack can succeed especially when the strategic user's network is faster than the victim's. In particular, we stress that the weak fair-sequencing model is sequencing orders based on their *arrival times, not the time of the submission of these orders.*

Recall that a user's intrinsic *type* is of the form (t, v, r, α) where (t, v, r) denotes the user's true valuation and budget. In the weak fair-sequencing model, we will use the α field to encode the order's arrival time – a smaller α means that the user arrives earlier.

We shall assume that under honest strategy, a user's order should always be populated with the correct α whose value is determined by nature, and equal to the time at which the order is generated plus the user's network delay. A strategic user is allowed to delay the submission of its order.

Strategy space in the weak fair-sequencing model. We consider the following strategy space in the weak fair-sequencing model:

- A strategic user or miner with intrinsic type (t, v, r, α) is allowed to post zero or multiple bids of the form $(\_, \_, \_, \alpha')$ as long as $\alpha' \geq \alpha$. This captures misreporting valuation and demand, posting fake orders, as well as delaying the posting of ones' orders.
- The strategic user or miner can decide its strategy *after* observing honest users' orders.

Compared to the plain model, the weak fair-sequencing model imposes some restrictions on the strategy space. Specifically, in the plain model, a strategic user or miner can arbitrarily modify the α field of its own order or even others' orders, and a strategic miner may censor honest users' orders. In the weak fair-sequencing model, a strategic user or miner can no longer under-report its α, cannot modify honest users' α, and cannot censor honest users' orders, because the sequencing of the transactions is determined by the underlying decentralized sequencer.

Importantly, despite these constraints on the strategy space, the weak fair-sequencing model still permits front-running-style attacks as mentioned earlier, and thus mechanism design remains non-trivial even under the slightly restricted strategy space.

2.5 Strategy Proof

Defining a user's preference among outcomes through a partial ordering. To define strategy proofness, we first need to define a ranking system that expresses a user's preference among different outcomes.

We can use a pair $(\delta x, \delta y)$ to denote the outcome, meaning that the user has a net gain of δx in X, and it has a net gain of δy in Y (where a net loss is captured as negative gain). Consider two outcomes $(\delta x_0, \delta y_0)$ and $(\delta x_1, \delta y_1)$, and suppose that the user's intrinsic type is $T = (\mathsf{Buy}(X), v, r, \_)$. Naturally, for such a user, outcome $(\delta x_1, \delta y_1)$ is at least as good as $(\delta x_0, \delta y_0)$, henceforth denoted $(\delta x_0, \delta y_0) \preceq_T (\delta x_1, \delta y_1)$, if one of the following is true:

- $\delta x_0 \leq \delta x_1$, $\delta y_0 \leq \delta y_1$. In other words, relative to $(\delta x_0, \delta y_0)$, the user gains no less in either asset in the latter outcome $(\delta x_1, \delta y_1)$.
- $\delta x_0 \leq \delta x_1 \leq v$ (or $q \leq \delta x_1 \leq \delta x_0$), and $r(\delta x_1 - \delta x_0) \geq \delta y_0 - \delta y_1$. In other words, the latter outcome $(\delta x_1, \delta y_1)$ is closer to satisfying the demand v, and moreover, the user paid at most r *marginal* price for each *extra* unit of X in the latter outcome.

▶ Remark 2.4 (Why define a partial ordering rather than a real-valued utility). The reader may wonder why we do not define a real-valued utility which would have given a total ordering on all outcomes. The reason why we define only a partial ordering and allow some outcomes to be incomparable is because a strategic user (e.g., whose intrinsic demand is to buy up to v units of X) can act arbitrarily, which may cause its net gain δx in X to be either negative, or greater than the intrinsic demand v. We allow some of these outcomes to be incomparable. For example, suppose relative to $(\delta x_0, \delta y_0)$, the latter outcome $(\delta x_1, \delta y_1)$ buys some extra units at a margial price better than the specified rate r, but it overshoots the intrinsic demand, then these two outcomes are incomparable.

Finally, the case for other types including $\mathsf{Buy}(Y)$, $\mathsf{Sell}(X)$, and $\mathsf{Sell}(Y)$ types, a partial ordering can be symmetrically defined – we give the full definition of the partial ordering in the online full version. Moreover, the online full version also gives more explanations why we choose to define a partial ordering to rank the outcomes rather than a real-valued utility.

Definition of strategy proofness. Since in our paper, we consider deterministic mechanisms, we will define strategy proofness only for a deterministic mechanism. Note that the definition can easily be extended to randomized mechanism using suitable notions of stochastic dominance.

In the definition below, we use $HS(T)$ to denote the honest strategy of a user with intrinsic type T – for a direct-revelation mechanism, the honest strategy is simply to reveal the user's true type. Further, we use $\mathsf{out}^u(x_0, y_0, \mathbf{b})$ to denote the outcome of user u when the mechanism is executed over initial pool state $\mathsf{Pool}(x_0, y_0)$, and a vector of orders $\mathbf{b}$.

▶ **Definition 2.5** (Strategy proofness). *Given a deterministic swap mechanism, we say that it satisfies strategy proofness (w.r.t. some partial ordering relation $\preceq_T$), iff for any initial pool state $\mathsf{Pool}(x_0, y_0)$, for any vector of orders $\mathbf{b}_{-u}$ belonging to all other users except u, for any intrinsic type T of the strategic user u, for any possible strategic order vector $\mathbf{b}'$ of the user u, either $\mathsf{out}^u(x_0, y_0, \mathbf{b}_{-u}, HS(T)) \succeq_T \mathsf{out}^u(x_0, y_0, \mathbf{b}_{-u}, \mathbf{b}')$ or $\mathsf{out}^u(x_0, y_0, \mathbf{b}_{-u}, HS(T))$ and $\mathsf{out}^u(x_0, y_0, \mathbf{b}_{-u}, \mathbf{b}')$ are incomparable w.r.t. $\preceq_T$.*

More intuitively, the definition says that *no strategic play can result in a strictly better outcome than the honest strategy.*

Strategy proofness implies arbitrage resilience. The fact below shows that strategy proofness implies arbitrage resilience.

▶ **Fact 1.** *Suppose that the potential function Φ is increasing. We have that strategy proofness implies arbitrage resilience.*

Proof. We can prove the contra-positive, that is, a mechanism that is not arbitrage free cannot be strategy proof. Suppose that the mechanism is not arbitrage free. This means that there exists a list of orders $S = \{(t_i, v_i, r_i, \alpha_i)\}_i$ such that under honest execution, a subset of the orders $S' \subseteq S$ will enjoy $\delta x \geq 0$ and $\delta y \geq 0$, and at least one of the two is strictly positive. Now imagine that there is a world that consists of the orders $S \backslash S'$. In this case, a strategic user or miner with 0 demand can inject a set of fake orders S', and clearly this strategic behavior has positive gain, thus violating strategy proofness. Note that this strategy works even in the weak fair-sequencing model, as long as the strategic user's inherent arrival time α is no larger than the orders in S'. ◀

Since our plain-model mechanism gives an example that satisfies arbitrage resilience but not strategy proof, we conclude that strategy proofness is *strictly* stronger than arbitrage resilience. Specifically, at the beginning of Section 3.3, we explain why we cannot achieve strategy proof in the plain model. The strategies presented in this section also help to illustrate why the stronger notion of strategy proofness is more desirable.

3 Our Swap Mechanism

3.1 Construction

Our swap mechanism has two phases. In phase 1 (line 3c and 4c), the mechanism matches $\mathsf{Buy}(X)$ orders with $\mathsf{Buy}(Y)$ orders, and (partially) executes them at the initial rate r_0, such that at the end, there is no change to the initial pool state $\mathsf{Pool}(x_0, y_0)$. Phase 2 (line 3d and 4d) is a $\mathsf{Buy}(X)$-only phase, in which a sequence of $\mathsf{Buy}(X)$ orders (or $\mathsf{Buy}(Y)$ orders) are (partially) executed one by one. In phase 2, when the mechanism attempts to execute an order, it will execute as much as possible until either the demand has been fulfilled, or the new market price has reached the asking price. The details of the mechanism are described in Figure 1.

Our swap mechanism

Input: A current pool state $\mathsf{Pool}(x_0, y_0)$, and a vector of orders $\mathbf{b}$. Since the mechanism does not make use of the auxiliary information field, we simply assume each order is a tuple of the form (t, v, r).

Mechanism:

1. Let $r_0 := r(x_0, y_0)$ be the initial exchange rate. Ignore all $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$ orders whose specified rate $r < r_0$, and ignore all $\mathsf{Buy}(Y)/\mathsf{Sell}(X)$ orders whose specified rate $r > r_0$. Let $\mathbf{b}'$ be the remaining orders.
2. Let $\sigma = \sum_{(t,v,r)\in\mathbf{b}'} \beta(t, v, r)$ where $\beta(t, v, r) = \begin{cases} v & \text{if } t = \mathsf{Buy}(X) \\ -v & \text{if } t = \mathsf{Sell}(X) \\ -v/r_0 & \text{if } t = \mathsf{Buy}(Y) \\ v/r_0 & \text{if } t = \mathsf{Sell}(Y) \end{cases}$
   We call $\sigma \geq 0$ the $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$-dominant case, and $\sigma < 0$ the $\mathsf{Buy}(Y)/\mathsf{Sell}(X)$-dominant case.
3. The $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$-dominant case (if $\sigma \geq 0$):
   - **a.** Sort $\mathbf{b}'$ such that all the $\mathsf{Buy}(Y)/\mathsf{Sell}(X)$ orders appear in front of the $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$ orders. Write the resulting list of orders as $\{(t_i, v_i, r_i)\}_{i\in[n']}$.
   - **b.** Henceforth we assume that there exists an index $j \in [n']$ such that $\sum_{i=1}^{j} \beta(t_i, v_i, r_i) = 0$. If not, we can find the smallest index $j \in [n']$ such that $\sum_{i=1}^{j} \beta(t_i, v_i, r_i) > 0$, and split the j-th order into two orders $(t_j, v_{j,0}, r_j)$ and $(t_j, v_{j,1}, r_j)$, resulting in a new list with $n' + 1$ orders, such that $v_{j,0} + v_{j,1} = v_j$, and moreover, index j of the new list satisfies this condition.
   - **c.** Phase 1: Fully execute the first j orders at the initial rate r_0.
   - **d.** Phase 2: For each $i \geq j + 1$ in sequence, fulfill as much of the i-th remaining order as possible, that is, pick the largest $v \leq v_i$ such that subject to the constant-function market maker Φ, the new market rate $r \leq r_i$ if v units are to be executed; execute v units of the i-th order.
4. The $\mathsf{Buy}(Y)/\mathsf{Sell}(X)$-dominant case is symmetric (if $\sigma < 0$):
   - **a.** Sort $\mathbf{b}'$ such that all the $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$ orders appear after the $\mathsf{Buy}(Y)/\mathsf{Sell}(X)$ orders. Write the resulting list of orders as $\{(t_i, v_i, r_i)\}_{i\in[n']}$.
   - **b.** Henceforth we assume that there exists an index $j \in [n']$ such that $\sum_{i=1}^{j} \beta(t_i, v_i, r_i) = 0$. If not, we can find the smallest index $j \in [n']$ such that $-\sum_{i=1}^{j} \beta(t_i, v_i, r_i) > 0$, and split the j-th order into two orders $(t_j, v_{j,0}, r_j)$ and $(t_j, v_{j,1}, r_j)$, resulting in a new list with $n'+1$ orders, such that $v_{j,0}+v_{j,1} = v_j$, and moreover, index j of the new list satisfies this condition.
   - **c.** Phase 1: Fully execute the first j orders at the initial rate r_0.
   - **d.** Phase 2: For $i \geq j + 1$ in sequence, fulfill as much of the i-th remaining order as possible, that is, pick the largest $v \leq v_i$ such that subject to the constant-function market maker Φ, the new market exchange rate $r \geq r_i$ if v units are to be executed; execute v units of the i-th order.

Figure 1 Our swap mechanism.

In the sorting steps (Lines 3a and 4a[6]), we may need to break ties among identical orders. We suggest two approaches for tie-breaking:

- In the presence of a centralized block proposer (i.e., when the arbitrager can be in full control of block creation), we suggest *random tie-breaking*. Note that the random tie-breaking is enforced by the mechanism (i.e., the smart contract). The random coins needed should come from a trusted source at the consensus layer, e.g., through the use of a fair coin toss protocol [18, 27]. This ensures that the miner or block producer cannot auction off favorable positions in the block to arbitragers or users.
  Note that in the plain model, we aim to achieve only arbitrage-free and not strategy proofness. So although it may seem like a user can adopt a Sybil strategy and submit fake bids to game the random tie-breaking, such strategies do not actually violate the arbitrage-free property..
- If the block proposal process is decentralized and ensures sequencing fairness, we suggest tie-breaking according to the time of arrival. Section 3.3 and Section 4 show that this approach allows us to achieve strategy proofness in the weak fair-sequencing model.

3.2 Proof of Arbitrage Resilience

We now prove that our swap mechanism satisfies arbitrage resilience regardless of how ties are broken in Lines 3a and 4a. As mentioned earlier, the arbitrage resilience property holds even when the arbitrager is in full control of block creation, can drop or inject orders, and can control the sequencing of orders within the block.

▶ **Theorem 3.1** (Arbitrage resilience). *The swap mechanism in Figure 1 satisfies arbitrage resilience. In particular, this holds no matter how ties are broken in Lines 3a and 4a.*

Proof. We prove it for the $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$-dominant case, since the $\mathsf{Buy}(Y)/\mathsf{Sell}(X)$-dominant case is symmetric. The mechanism essentially does the following. In phase 1, it (partially) executes a set of orders all at the initial rate r_0, such that there is no change to the initial pool state $\mathsf{Pool}(x_0, y_0)$. In phase 2, it executes only $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$ orders. Due to increasing marginal cost (Fact 2), in Phase 2, all the (partially) executed $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$ orders enjoy a rate that is at least r_0. Therefore, all the (partially) executed $\mathsf{Buy}(Y)/\mathsf{Sell}(X)$ orders enjoy a rate of r_0, and all the (partially) executed $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$ orders enjoy a rate that is r_0 or greater. Thus, it cannot be the case that there is a net gain in one asset without any loss in the other. ◀

3.3 A Refinement of the Mechanism for the Weak Fair-Sequencing Model

Why the mechanism is NOT strategy-proof in the plain model. As mentioned, our swap mechanism in Figure 1 does not fully specify how to break ties in the sorting steps of in Lines 3a and 4a. Moreover, our arbitrage resilience property does not care how the tie-breaking is done.

However, if we are not careful about the tie-breaking the resulting mechanism may not satisfy strategy proofness. For example, imagine that the tie-breaking is based on the α field of the order, which encodes the time at which the order is submitted. In this case, a strategic miner or user can simply make its own order have a small α to enjoy a better price, rather than truthfully reporting α.

[6] Note that the sorted order produced by (3a) will be consumed in the subsequent steps (3b), (3c), and (3d); similarly, the sorted order of (4a) are consumed in (4b), (4c), and (4d).

Refinement in the fair-sequencing model. In the weak fair-sequencing model, the arrival order α is decided by an underlying consensus protocol that provides some form of sequencing fairness. In other words, the consensus protocol itself records the approximate time at which each order is first seen. In practice, this arrival time is dependent on when the user submits the its order to the network, and its network delay. A sequencing-fair consensus protocol cannot prevent a user (or miner) from delaying the submission of its order. It also cannot prevent a strategic user with an extremely fast network from front-running a victim's order. Specifically, the strategic user can still observe what the victim submits, and then immediately submits a dependent order. If the strategic user's network is faster than the victim's, the strategic order may have an earlier arrival time than the victim's order! However, sequencing fairness from the consensus does tie the hands of strategic players in the following weak manner (hence the name "weak sequencing fair"): if a strategic user's order is generated at time α, it cannot pretend that the order was generated at $\alpha' < \alpha$ even if it has an extremely fast network with a delay of 0.

We can refine the mechanism in Figure 1 as follows to achieve strategy proofness in the weak fair-sequencing model (see also Section 4):

For tie-breaking in the sorting steps (Lines 3a and 4a), we now require that the sorting be stable, that is, in the sorted outcome, the relative ordering among all identical Buy(Y) orders must respect the arrival order encoded in the α field; the same holds for all Buy(X) orders.

We will assume that a user's honest strategy is to honestly report its type including the α field, that is, $HS(t, v, r, \alpha)$ simply outputs a single order (t, v, r, α).

▶ **Theorem 3.2** (Strategy proofness in the weak fair-sequencing model). *Suppose Φ is concave, increasing, and differentiable. In the weak fair-sequencing model, the above refined swap mechanism is strategy proof (see Definition 2.5).*

The proof of Theorem 3.2 is provided in Section 4.

4 Proof of Strategy Proofness in the Weak Fair-Sequencing Model

4.1 Useful Facts

We first prove a few useful facts.

▶ **Fact 2** (Increasing marginal cost). *Suppose that Φ is increasing, differentiable, and concave. Given two pool states* Pool(x, y), Pool(x', y') *such that $\Phi(x, y) = \Phi(x', y')$, and $x' \leq x$, it must be that $r(x, y) \leq r(x', y')$. In other words, the price of X goes up if the pool has less supply of X.*

Proof. Suppose $\Phi(x, y) = C$. Since Φ is increasing, by Lemma B.1 of [30], the potential function Φ defines a bijective decreasing function $h(\cdot)$ such that $\Phi(z, h(z)) = C$. Moreover, since Φ is concave, the induced function $h(\cdot)$ is convex (Lemma B.2 of [30]). Observe that Φ is differentiable, so $r(x, y) = -h'(x)$. Therefore, $r(x, y) \leq r(x', y')$ for $x' \leq x$ by the convexity of h. ◀

▶ **Fact 3** (No free lunch). *Suppose $(\delta x, \delta y)$ is the outcome resulting from the execution of a single order in the swap mechanism in Figure 1. If at least one of δx and δy is non-zero, then $\delta x \cdot \delta y < 0$.*

4.2 Proof

We now prove Theorem 3.2. Our mechanism is deterministic, so a deterministic strategy yields a deterministic outcome. Recall that a user has a partial ordering among the outcomes. Henceforth, if two outcomes satisfy $(\delta x_1, \delta y_1) \succeq (\delta x_2, \delta y_2)$, we say that $(\delta x_1, \delta y_1)$ is *at least as good as* $(\delta x_2, \delta y_2)$.

Suppose that the strategic user u's type is $(t^*, v^*, r^*, \alpha^*)$, its strategic order vector is $\mathbf{b}_u := \{(t_j, v_j, r_j, \alpha_j)\}_j$, the initial state is $\mathsf{Pool}(x_0, y_0)$, and the order vector from all other users is $\mathbf{b}_{-u}$.

▶ **Fact 4.** *Given the initial state* $\mathsf{Pool}(x_0, y_0)$*, the order vector from other users* $\mathbf{b}_{-u}$*, and a strategic order vector* $\mathbf{b}_u := \{(t_j, v_j, r_j, \alpha_j)\}_j$*, there exists an alternative strategic vector* $\mathbf{b}'_u$ *which contains only* $\mathsf{Buy}(X)$ *and* $\mathsf{Sell}(X)$*-type (or* $\mathsf{Buy}(Y)$ *and* $\mathsf{Sell}(Y)$*-type) order, such that the outcome of* $\mathbf{b}'_u$ *is the same as the outcome of* $\mathbf{b}_u$*.*

Proof. Since the mechanism is deterministic, given the initial state $\mathsf{Pool}(x_0, y_0)$, order vector from all other users $\mathbf{b}_{-u}$, and the strategic order vector $\mathbf{b}_u$, one can compute the whole order execution trace of the mechanism. For an order $b = (\mathsf{Sell}(Y), v_j, r_j, \alpha_j)$ in $\mathbf{b}_u$, if it is not executed, then it can be replaced with a $(\mathsf{Buy}(X), 0, r_j, \alpha_j)$. If it is partially fulfilled, let $(x_{\text{start}}, x_{\text{end}})$ denote the amount of asset X in the pool right before and right after b is executed, respectively. Let r_{end} denote the market exchange rate at x_{end}. Then b can be replaced with an order $(\mathsf{Buy}(X), x_{\text{start}} - x_{\text{end}}, r_{\text{end}}, \alpha_j)$, without changing the outcome. Similarly, we can replace a $\mathsf{Buy}(Y)$-type order with a $\mathsf{Sell}(X)$-type order without changing the outcome. The fact thus follows. ◀

▶ **Lemma 4.1.** *Suppose* Φ *is concave, increasing, and differentiable. For any strategic order vector* $\mathbf{b}_u$*, there exists a single order* b'_u *such that 1)* b'_u *results in an outcome at least as good as* $\mathbf{b}_u$*; 2) the arrival time used in* b'_u *is no earlier than the earliest arrival time in* $\mathbf{b}_u$*; and 3) either* $b'_u = (\_, 0, \_, \_)$ *or* b'_u *would be completely fulfilled under* $\mathsf{Pool}(x_0, y_0)$ *and* $\mathbf{b}_{-u}$*.*

Proof. To prove this lemma, we first show that we can coalesce all the $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$-type orders into one, and all the $\mathsf{Sell}(X)/\mathsf{Buy}(Y)$-type orders into one, as stated in the following claim.

▷ Claim 4.2. Suppose Φ is concave, increasing, and differentiable. For any strategic order vector $\mathbf{b}_u$, if it contains both $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$-type and $\mathsf{Sell}(X)/\mathsf{Buy}(Y)$-type orders, there exists another order vector $\mathbf{b}'_u$ which contains a single $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$-type order and a single $\mathsf{Sell}(X)/\mathsf{Buy}(Y)$-type order such that 1) $\mathbf{b}'_u$ results in an outcome at least as good as $\mathbf{b}_u$; 2) the arrival times used in $\mathbf{b'}_u$ are no earlier than the earliest arrival time in $\mathbf{b}_u$; and 3) an order in $\mathbf{b}'_u$ is either of the form $(\_, 0, \_, \_)$ or it would be completely fulfilled under $\mathsf{Pool}(x_0, y_0)$ and $\mathbf{b}_{-u}$.

Proof. According to Fact 4, we can assume that $\mathbf{b}_u$ contains only $\mathsf{Buy}(X)$ and $\mathsf{Sell}(X)$-type orders. Let $\mathbf{b}_{\mathsf{Sell}}$ and $\mathbf{b}_{\mathsf{Buy}}$ denote the vector of $\mathsf{Sell}(X)$-type and $\mathsf{Buy}(X)$-type orders in $\mathbf{b}_u$, respectively. Without loss of generality, assume that given $\mathbf{b}_{-u}$ and $\mathbf{b}_u$, we have the $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$-dominant case, (the $\mathsf{Buy}(Y)/\mathsf{Sell}(X)$-dominant case is symmetric).

For the $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$-dominant case, all $\mathsf{Sell}(X)$-type orders in $\mathbf{b}_{\mathsf{Sell}}$ will be executed at the initial exchange rate r_0. Therefore, consider an order $b = (\mathsf{Sell}(X), v, r, \alpha)$, where v denotes the total units of orders in $\mathbf{b}_{\mathsf{Sell}}$, r is the minimum asking rate in $\mathbf{b}_{\mathsf{Sell}}$, and α is the earliest arrival time in $\mathbf{b}_u$. Then a strategic order vector $(\mathbf{b}_{\mathsf{Buy}}, b)$ results in the same outcome as $\mathbf{b}_u$.

Now consider an order $b' = (\mathsf{Buy}(X), v', r', \alpha')$, where v' is the total units of order executed in $\mathbf{b}_{\text{Buy}}$, r' is the maximum asking rate in $\mathbf{b}_{\text{Buy}}$, and α' is the earliest arrival time of the order in $\mathbf{b}_{\text{Buy}}$ that is partially fulfilled. Let $\mathbf{b}'_u = (b', b)$. Compared to the units executed in $\mathbf{b}_{\text{Buy}}$, the units executed in b' have an earlier or the same arrival time. In addition, the asking rate r' in b' is larger than or equal to that in $\mathbf{b}_{\text{Buy}}$. Therefore, all units in b' will be executed. Moreover, by the increasing marginal cost (Fact 2), the exchange rate for b' is no more than the average exchange rate for all orders in $\mathbf{b}_{\text{Buy}}$. This means that $\mathbf{b}'_u$ results in an outcome that is at least as good as $(\mathbf{b}_{\text{Buy}}, b)$ according to the partial ordering. By the transitivity, $\mathbf{b}'_u$ results in an outcome that is at least as good as $\mathbf{b}_u$. $\lhd$

Next, we show that for any order vector $\mathbf{b}_u$ that contains a single $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$-type order and a single $\mathsf{Sell}(X)/\mathsf{Buy}(Y)$-type order, we can remove the part that "cancels off", and substitute it with a single order. This is formally stated in the following claim.

$\rhd$ Claim 4.3. Suppose Φ is concave, increasing, and differentiable. For any order vector $\mathbf{b}_u$ that contains a single $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$-type order and a single $\mathsf{Sell}(X)/\mathsf{Buy}(Y)$-type, there exists a single order b'_u such that 1) b'_u results in an outcome at least as good as $\mathbf{b}_u$; 2) the arrival time in b'_u is no earlier than the earliest arrival time in $\mathbf{b}_u$; and 3) b'_u is either of the form $(\_, 0, \_, \_)$ or it would be completely fulfilled under $\mathsf{Pool}(x_0, y_0)$ and $\mathbf{b}_{-u}$.

Proof. Because of Fact 4, we assume that the strategic order vector $\mathbf{b}_u$ contains a single $\mathsf{Buy}(X)$-type order $(\mathsf{Buy}(X), v_b, r_b, \alpha_b)$ and a single $\mathsf{Sell}(X)$-type order $(\mathsf{Sell}(X), v_s, r_s, \alpha_s)$. By our assumption, both orders are fully executed. If either v_b or v_s is zero, the result follows trivially; henceforth, we assume that both are non-zero. Similarly, if either $r_b < r_0$ or $r_s > r_0$, then the result also follows trivially. henceforth, $r_b \geq r_0$ and $r_s \leq r_0$.

We prove it assuming the $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$-dominant case case under the strategic order vector $\mathbf{b}_u$, since the argument for the $\mathsf{Buy}(Y)/\mathsf{Sell}(X)$-dominant case is symmetric. For the $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$-dominant case, if it were possible to execute all orders at r_0, then there is more demand in terms of only $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$ than $\mathsf{Sell}(X)/\mathsf{Buy}(Y)$. In this case, Phase 1 executes all $\mathsf{Sell}(X)/\mathsf{Buy}(Y)$ orders at r_0, and Phase 2 executes only $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$ orders.

Case $v_s \geq v_b$. Under the original $\mathbf{b}_u$, the user would sell v_s units of X at r_0 and would buy v_b units of X at a rate of r_0 or greater. Now, suppose we replace $\mathbf{b}_u$ with a single order $b'_u = (\mathsf{Sell}(X), v_s - v_b, r_s, \alpha_s)$. Under b'_u, it is still the case that all $\mathsf{Sell}(X)$ orders are completely executed at r_0.

Hence, we can decompose the original $\mathbf{b}_u$ equivalently into the following steps: (i) first execute b'_u; (ii) sell v_b units at rate r_0; (iii) buy back v_b units at rate r_0 or greater. Since steps (ii) and (iii) together will incur a non-negative loss in Y (but create no change in X), user u's outcome under b'_u is at least as good as $\mathbf{b}_u$.

Case $v_s < v_b$. Consider the original $\mathbf{b}_u$ which consists of $(\mathsf{Sell}(X), v_s, r_s, \alpha_s)$ and $(\mathsf{Buy}(X), v_b, r_b, \alpha_b)$. We will analyze what happens when we replace these two orders with a single order $b'_u = (\mathsf{Buy}(X), v_b - v_s, r_b, \alpha_b)$. Suppose in Phase 1 of the execution with $\mathbf{b}_u$, user u sells v_s units of X and buys $v' \leq v_b$ units of X at rate r_0. We separate the rest of the proof into two cases.

1. **Case $v' \geq v_s$.** For the execution with $\mathbf{b}_u$: In Phase 1, the net effect is to buy $v' - v_s$ units of X at rate r_0. In Phase 2, the user u buys the remaining $v_b - v'$ units starting at rate r_0.
   The execution with b'_u can be viewed as follows: In Phase 1', $v' - v_s$ units of $\mathsf{Buy}(X)$ will be executed at rate r_0. Then in Phase 2', the mechanism executes the rest $v_b - v'$ units in b'_u starting at rate r_0.

Hence, the two scenarios are equivalent, and the two outcomes are the same.

2. **Case $v' < v_s$.** We will view the execution of the orders $(\mathsf{Sell}(X), v_s, r_s, \alpha_s)$ and $(\mathsf{Buy}(X), v_b, r_b, \alpha_b)$ as follows.
   - *Phase 1:* User u sells v_s units of X and buy v' units of X at a rate of r_0, and it gains $(v_s - v') \cdot r_0$ units of Y in return.
   - *Phase 2a:* Some non-negative amount of $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$ orders from other users are executed at a starting market rate of r_0, let $v_{\text{other}} \geq 0$ be the units of X purchased. Note that if $v' > 0$, then $v_{\text{other}} = 0$.
     At the end of this phase, the market rate $r_1 \geq r_0$ by increasing marginal cost.
   - *Phase 2b:* Starting at rate $r_1 \geq r_0$, user u buys $v_s - v'$ units of X, which changes the market rate to $r_2 \geq r_1$.
   - *Phase 2c:* Starting at rate r_2, the user buys $(v_b - v_s)$ units of X, changing the market rate to r_3.

   The new execution involving b'_u can be viewed as the following:
   - *Phase 1'*: The $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$ orders of other users executed in the original Phase 1 cannot all be executed in the new Phase 1. In particular, the last $v_s - v'$ units of X cannot be fulfilled in Phase 1', and will be pushed to Phase 2a' – henceforth, we call this portion the residual.
   - *Phase 2a'*: The mechanism attempts to execute the residual from Phase 1' at a starting rate of r_0. The amount fulfilled must be at most $v_s - v'$.
   - *Phase 2b'*: The mechanism attempts to execute the (partial) orders originally considered in Phase 2a, at a starting price that is at least r_0. At most v_{other} units of X can be fulfilled. The ending market rate must be at least r_1.
   - *Phase 2c'*: The mechanism attempts to execute b'_u. Observe that the total units of X fulfilled in the original Phase 2a, 2b, and 2c is $v_{\text{other}} + v_b - v'$, and the total units of X fulfilled in the new Phase 2a', 2b', and 2c' is at most $(v_s - v') + v_{\text{other}} + v_b - v_s = v_{\text{other}} + v_b - v'$. Therefore, it must be that all of b'_u can be fulfilled and the ending market rate is at most $r_3 \leq r_b$.

   Henceforth, we use the notation $\mathsf{Pay}(\text{Phase }*)$ to denote user u's payment in terms of Y in some phase. When the pay is negative, it means a gain in Y. Observe that

   $$\mathsf{Pay}(\text{Phase 1}) + \mathsf{Pay}(\text{Phase 2b}) \geq 0, \quad \mathsf{Pay}(\text{Phase 2c}) \geq \mathsf{Pay}(\text{Phase 2c'}).$$

   Therefore,

   $$\mathsf{Pay}(\text{Phase 1}) + \mathsf{Pay}(\text{Phase 2b}) + \mathsf{Pay}(\text{Phase 2c}) \geq \mathsf{Pay}(\text{Phase 2c'}).$$

   Observe that in the above, the left-hand side represents u's total payment in Y under the original $\mathbf{b}'_u$, and the right-hand side represents u's total payment in Y under the new b'_u. ◁

Lemma 4.1 follows by combining Claim 4.2 and Claim 4.3. ◀

▶ **Lemma 4.4.** *Suppose Φ is concave, increasing, and differentiable. Given any initial state $\mathsf{Pool}(x_0, y_0)$, any order vector $\mathbf{b}_{-u}$, any true arrival time α^* of user u, given an order b'_u with an arrival time later than α^*, there exists another order b_u with an arrival time exactly α^*, and moreover, user u's outcome under b_u is at least as good as its outcome under b'_u, and b_u is either completely executed or of the form $(\_, 0, \_, \_)$.*

Proof. Due to Fact 4, we may assume that b_u is either a $\mathsf{Buy}(X)$ or $\mathsf{Sell}(X)$ order. We prove it for a $\mathsf{Buy}(X)$ order, since the case for a $\mathsf{Sell}(X)$ order is symmetric. Let v be the amount of X bought by b'_u. We shall assume that $v > 0$ since the case $v = 0$ is trivial. Consider an order $b_u = (\mathsf{Buy}(X), v, +\infty, \alpha^*)$. Clearly, b_u will buy v units of X.

We consider the following cases:

- b_u buys $v_0 > 0$ units at r_0 in Phase 1, and then buys $v_1 \geq 0$ units in Phase 2 at a starting rate of r_0 and an ending rate of $r_1 \geq r_0$. In this case, by delaying the arrival time, b'_u can buy at most $v'_0 \leq v_0$ units in Phase 1 at r_0, and it needs to buy $v - v'_0 \geq v_1$ in Phase 2. Therefore, for b'_u, the starting rate in Phase 2 is r_0, and the ending rate must be at least r_1. Therefore, the average price paid per unit in b_u is no worse than the average price paid per unit in b'_u.
- b_u buys all v units in Phase 2. Since b'_u delayed the arrival, the order can be considered no earlier than b_u. Thus, before the mechanism tries to execute b'_u at least as many units of X will have been bought (by all users) as when the mechanism tries to execute b_u. This means b'_u will have an average price no better than b_u. ◀

Due to Lemmas 4.1 and 4.4, it suffices to consider strategies that submit a single order, declare the true arrival time α^*, and moreover, either the order has a 0 amount or it will be completely executed under $\mathsf{Pool}(x_0, y_0)$ and $\mathbf{b}_{-u}$ – henceforth, we call such strategies as *admissible, single-order* strategies. We can complete the proof of Theorem 3.2 by showing the following lemma.

▶ **Lemma 4.5.** *Suppose Φ is concave, increasing, and differentiable. For any admissible and single-order strategy S, the honest strategy results in an outcome that is at least as good as or incomparable to strategy S.*

Proof. Below we complete the proof of Lemma 4.5.

We prove it for the case when user u's type is either $(\mathsf{Buy}(X), v^*, r^*, \alpha^*)$ or $(\mathsf{Sell}(X), v^*, r^*, \alpha^*)$. The case for $\mathsf{Buy}(Y)/\mathsf{Sell}(Y)$ is symmetric. Given two outcomes $\mathsf{out}_0 = (\delta x_0, \delta y_0)$ and $\mathsf{out}_1 = (\delta x_1, \delta y_1)$ and a true demand δx^* for X, we say that they are *on the same side* of the goal δx^* iff $(\delta x_0 - \delta x^*) \cdot (\delta x_1 - \delta x^*) \geq 0$. We say that out_0 is *at least as close as* out_1 towards the goal iff $|\delta x_0 - \delta x^*| \leq |\delta x_1 - \delta x^*|$, and we say that out_0 is *closer* to the goal than out_1 iff $|\delta x_0 - \delta x^*| < |\delta x_1 - \delta x^*|$.

Our natural partial ordering relation implies the following:

R1. Suppose $(\delta x_0, \delta y_0)$ and $(\delta x_1, \delta y_1)$ are on the same side of the goal, and δx_0 is at least as close as δx_1 towards the goal. Moreover, if $(\delta x_0 - \delta x_1) \cdot (\delta y_1 - \delta y_0) < 0$, then $(\delta x_1, \delta y_1) \not\succeq (\delta x_0, \delta y_0)$.

R2. If δx_0 and δx_1 are on the same side of the goal, δx_0 is closer than δx_1 to the goal, and moreover, $\delta y_1 - \delta y_0 > r^* \cdot (\delta x_0 - \delta x_1)$, then $(\delta x_0, \delta y_0) \not\succeq (\delta x_1, \delta y_1)$.

R3. If δx_0 and δx_1 are on different sides of the goal, and moreover, $\delta y_1 - \delta y_0 \geq r^* \cdot (\delta x_0 - \delta x_1)$, then $(\delta x_0, \delta y_0) \not\succeq (\delta x_1, \delta y_1)$.

Due to Fact 4, we may assume that the strategic order must be of the type $\mathsf{Buy}(X)$ or $\mathsf{Sell}(X)$. Further, as shown in the following fact, we can in fact assume that the strategic order adopts the true time of arrival α^*, i.e., declaring a later time never helps.

Henceforth, let $(\delta x, \delta y)$ and $(\delta x', \delta y')$ denote the honest and strategic outcomes, respectively.

Case 1. Either user u has a true demand of 0 units, or the strategic order is opposite the direction of its true demand, i.e., if its type is $\mathsf{Buy}(X)$, it submits a single $\mathsf{Sell}(X)$ order; or vice versa. It must be that $\delta x \cdot \delta x' \leq 0$. Further, the honest outcome and strategic outcome must be on the same side of the true demand, and the honest outcome is at least as close to the goal as the strategic outcome. Our mechanism guarantees that either (i) $\delta x = \delta x' = 0$, or (ii) at least one of $\delta x'$ and $\delta x'$ is non-zero. In case (i), $\delta y = \delta y' = 0$, and the the two outcomes are the same.

In case (ii), because of no free lunch (Fact 3), at least one of the inequalities $\delta x \cdot \delta y \le 0$ and $\delta x' \cdot \delta y' \le 0$ must be strict; moreover, when an equality holds, both arguments of the product must be zero. Because $\delta x \cdot \delta x' \le 0$, this implies that $(\delta x - \delta x')(\delta y - \delta y') < 0$; by the above rule R1, the honest outcome is at least as good as or incomparable the strategic one.

Case 2. The user u has a non-zero amount of true demand, and moreover, the strategic order is in the same direction of the true demand. We consider the following cases.

- *Case 2a: $\delta x = \delta x'$.* By admissibility, the strategic order declares the same arrival time as the honest one; hence, if the orders from both strategies get executed for a non-zero amount, both execution will start at the same market exchange rate. Hence, it must be the case that $(\delta x, \delta y) = (\delta x', \delta y')$.
  Henceforth, we assume that $\delta x \neq \delta x'$.
- *Case 2b:* The honest outcome and the strategic outcome are on the same side of the goal, and the honest outcome closer to the goal than the strategic outcome; this case includes the scenario that the honest outcome is exactly at the goal. Since we can assume that the strategic order has the same arrival time as the honest order, the difference of $|\delta x - \delta x'|$ units are traded at a marginal price at least as good as r in the honest outcome. Due to the third rule of the natural partial ordering, the honest outcome is at least as good as the stategic one.
- *Case 2c:* The honest outcome and the strategic outcome are on the same side of the goal, and the strategic outcome is closer to the goal than the honest outcome, i.e., $|\delta x| < |\delta x'| \le |\delta x^*| = v^*$.
  This means that the honest outcome has not reached the goal of user u. Under the honest strategy, after user u's order has been executed (or attempted to be executed), the state of the market is such that if a further non-zero portion of the order is executed, this portion will incur an average rate of strictly worse than r^*. In the case of $\mathsf{Buy}(X)$, this is strictly larger than r^*; in the case of $\mathsf{Sell}(X)$, this is strictly less than r^*.
  Since the strategic order declares the same arrival time as the honest order, the difference of $|\delta x' - \delta x| > 0$ units must be traded at an average rate strictly worse than than r^* in the strategic outcome. By rule R2, the strategic outcome is not at least as good as the honest outcome. However, because of no free lunch, the two outcomes are actually incomparable.
- *Case 2d:* The honest outcome and the strategic outcome are on different sides of the goal, i.e., $|\delta x| < |\delta x^*| = v^* < |\delta x'|$.
  In this case, $\delta x \neq 0$. Similar to case 2c, under the honest strategy, after user u's order has been executed, the state of the market is such that if a further non-zero portion of the order is executed, this portion will incur an average rate of strictly worse than r^*.
  Because the strategic order declares the same arrival time as the honest one, the difference of $|\delta x' - \delta x|$ units must be traded at an average rate of strictly worse than than r^* in the strategic outcome. By rule R3, the strategic outcome cannot be at least as good as the honest outcome. ◀

5 Conclusion

In this paper, we propose new models for studying mechanism design for DeFi applications. Unlike the prior work of Ferreira and Parkes [30] and others' [39] which assume that the mechanism on the blockchain must be a first-in-first-out mechanism, we allow the the mechanism designer to specify the mechanism running on the blockchain. This allows us to circumvent the strong impossibility results of Ferreira and Parkes [30]. Depending on assumption on the underlying blockchain, we consider two possible strategies spaces. If the underlying blockchain does not enjoy sequencing fairness, we assume that the strategic user (or miner) can post orders after observing honest users' orders, insert fake orders, censor honest users' orders, and control the sequencing of the orders in the block. If the underlying blockchain enjoys sequencing fairness, we assume that the strategic user can do all of the above; however, it cannot censor honest users' orders, nor can it under-report its arrival time.

We design a novel mechanism that achieves arbitrage resilience (which was deemed impossible under Ferreira and Parkes [30]'s model), and additionally achieves strategy proofness if the underlying consensus offers sequencing fairness.

Our paper raises many interesting directions for future work. For example, can we achieve strategy proofness without relying on the sequencing fairness assumption? Can we extend the results to multi-asset swaps? Can we optimize social welfare and revenue under strategy proofness? Another interesting direction is whether we can achieve strategy proofness under an all-or-nothing fulfillment model, that is, any order is either completely fulfilled or not executed at all.

References

1 `https://cow.fi/cow-protocol`.

2 The espresso sequencer. `https://hackmd.io/@EspressoSystems/EspressoSequencer`.

3 `https://www.metis.io/decentralized-sequencer`.

4 `https://www.zeeve.io/blog/why-would-a-layer2-decentralize-its-sequencer/`.

5 `https://ethglobal.com/showcase/decentralized-sequencers-for-optimism-rollup-1o98i`.

6 `https://morph.ghost.io/introduction-to-decentralized-sequencer-network/`.

7 `https://docs.morphl2.io/docs/how-morph-works/decentralized-sequencers/morph-decentralized-sequencer-network/`.

8 `https://blog.kroma.network/decentralized-sequencers-d2a4aeaf1084?gi=ee7408176d62`.

9 `https://ethglobal.com/talks/ordering-so-fair-it-ain-t-fair-ordering-a5b60`.

10 Challenging periods reimagined: The key role of sequencer decentralization. `https://ethresear.ch/t/challenging-periods-reimagined-the-key-role-of-sequencer-decentralization/15110`.

11 Guillermo Angeris, Alex Evans, and Tarun Chitra. A note on bundle profit maximization, 2021.

12 Kushal Babel, Philip Daian, Mahimna Kelkar, and Ari Juels. Clockwork finance: Automated analysis of economic security in smart contracts. In *IEEE Symposium on Security and Privacy*, 2023.

13 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, September 23-25, 2024, Vienna, Austria*, volume 316 of *LIPIcs*, pages 29:1–29:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. `doi:10.4230/LIPICS.AFT.2024.29`.

14 Massimo Bartoletti, James Hsin-yu Chiang, and Alberto Lluch-Lafuente. A theory of automated market makers in defi. *CoRR*, 2021. `arXiv:2102.11350`.

15 Massimo Bartoletti, James Hsin-yu Chiang, and Alberto Lluch Lafuente. Maximizing extractable value from automated market makers. In *Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers*, pages 3–19, Berlin, Heidelberg, 2022. `doi:10.1007/978-3-031-18283-9_1`.

16 Massimo Bartoletti, James Hsin-yu Chiang, and Alberto Lluch Lafuente. Maximizing extractable value from automated market makers. In *Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers*, 2022. doi:10.1007/978-3-031-18283-9_1.

17 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.

18 Adithya Bhat, Nibesh Shrestha, Zhongtang Luo, Aniket Kate, and Kartik Nayak. Randpiper reconfiguration-friendly random beacons with quadratic communication. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, pages 3502–3524, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3460120.3484574.

19 R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, 2001.

20 Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 2000.

21 Andrea Canidio and Robin Fritsch. Batching trades on automated market makers. In *AFT*, 2023.

22 Andrea Canidio and Robin Fritsch. Arbitrageurs' profits, lvr, and sandwich attacks: batch trading as an amm design response. *CoRR*, 2024. doi:10.48550/arXiv.2307.02074.

23 Andrea Canidio and Felix Henneke. Fair combinatorial auction for blockchain trade intents: Being fair without knowing what is fair, 2024. arXiv:2408.12225.

24 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*, 2016.

25 T-H. Hubert Chan, Ke Wu, and Elaine Shi. Mechanism design for automated market makers. https://arxiv.org/abs/2402.09357.

26 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.

27 Sourav Das, Vinith Krishnan, Irene Miriam Isaac, and Ling Ren. SPURT: Scalable distributed randomness beacon with transparent setup. In *IEEE Symposium on Security and Privacy*, 2022. URL: https://eprint.iacr.org/2021/100.

28 Theo Diamandis and Guillermo Angeris. A note on the welfare gap in fair ordering, 2023. doi:10.48550/arXiv.2303.15239.

29 Justin Drake. Encrypted mempools. https://www.youtube.com/watch?v=XRM0CpGY3sw.

30 Matheus Venturyne Xavier Ferreira and David C. Parkes. Credible decentralized exchange design via verifiable sequencing rules. In *STOC*, 2023.

31 Tivas Gupta, Mallesh M Pai, and Max Resnick. The centralizing effects of private order flow on proposer-builder separation, 2023. arXiv:2305.19150.

32 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*, 2022.

33 Josojo. Mev capturing amm (mcamm). https://ethresear.ch/t/mev-capturing-amm-mcamm/13336.

34 Mahimna Kelkar, Soubhik Deb, and Sreeram Kannan. Order-fair consensus in the permissionless setting. In *APKC '22: Proceedings of the 9th ACM on ASIA Public-Key Cryptography Workshop*, 2022.

35 Mahimna Kelkar, Soubhik Deb, Sishan Long, Ari Juels, and Sreeram Kannan. Themis: Fast, strong order-fairness in byzantine consensus, 2021.

36 Mahimna Kelkar, Fan Zhang, Steven Goldfeder, and Ari Juels. Order-fairness for byzantine consensus. In *CRYPTO*, pages 451–480, 2020. doi:10.1007/978-3-030-56877-1_16.

37 Kshitij Kulkarni, Theo Diamandis, and Tarun Chitra. Towards a theory of maximal extractable value I: constant function market makers. *CoRR*, abs/2207.11835, 2022. doi:10.48550/arXiv.2207.11835.

38 F. Leupold. Cow native amms (aka surplus capturing amms with single price clearing). https://forum.cow.fi/t/cow-native-amms-aka-surplus-capturing-amms-with-single-price-clearing/1219/1.

39 Yuhao Li, Mengqian Zhang, Jichen Li, Elynn Chen, Xi Chen, and Xiaotie Deng. Mev makes everyone happy under greedy sequencing rule. In *Proceedings of the 2023 Workshop on Decentralized Finance and Security*, DeFi '23, pages 9–15. Association for Computing Machinery, 2023. doi:10.1145/3605768.3623543.

40 Jason Milionis, Ciamac C. Moallemi, and Tim Roughgarden. A myersonian framework for optimal liquidity provision in automated market makers. *CoRR*, abs/2303.00208, 2023. doi:10.48550/arXiv.2303.00208.

41 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.

42 Kaihua Qin, Liyi Zhou, Benjamin Livshits, and Arthur Gervais. Attacking the defi ecosystem with flash loans for fun and profit. In *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part I*, 2021. doi:10.1007/978-3-662-64322-8_1.

43 Geoffrey Ramseyer, Mohak Goyal, Ashish Goel, and David Mazières. Augmenting batch exchanges with constant function market makers, 2023. arXiv:2210.04929.

44 Tim Roughgarden. Transaction fee mechanism design. In *EC*, 2021.

45 Elaine Shi, Hao Chung, and Ke Wu. What can cryptography do for decentralized mechanism design? In *ITCS*, volume 251 of *LIPIcs*, pages 97:1–97:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.ITCS.2023.97.

46 Jiahua Xu, Krzysztof Paruch, Simon Cousaert, and Yebo Feng. Sok: Decentralized exchanges (dex) with automated market maker (amm) protocols. *ACM Computing Surveys*, 55(11), 2023. doi:10.1145/3570639.

47 Sen Yang, Kartik Nayak, and Fan Zhang. Decentralization of Ethereum's Builder Market. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 1512–1530, Los Alamitos, CA, USA, May 2025. IEEE Computer Society. doi:10.1109/SP61157.2025.00157.

48 Mengqian Zhang, Yunhao Li, Xinyuan Sun, Elynn Chen, and Xi Chen. Computation of optimal mev in decentralized exchanges. https://mengqian-zhang.github.io/papers/batch.pdf.

49 Liyi Zhou, Kaihua Qin, Antoine Cully, Benjamin Livshits, and Arthur Gervais. On the just-in-time discovery of profit-generating transactions in defi protocols. In *IEEE Symposium on Security and Privacy, SP*, 2021.

50 Liyi Zhou, Kaihua Qin, and Arthur Gervais. A2MM: mitigating frontrunning, transaction reordering and consensus instability in decentralized exchanges. *CoRR*, abs/2106.07371, 2021. arXiv:2106.07371.

51 Liyi Zhou, Kaihua Qin, Christof Ferreira Torres, Duc Viet Le, and Arthur Gervais. High-frequency trading on decentralized on-chain exchanges. In *IEEE Symposium on Security and Privacy*, 2021.

52 Patrick Zust. Analyzing and preventing sandwich attacks in ethereum. Bachelor's thesis.