

6th Conference on Information-Theoretic Cryptography

ITC 2025, August 16–17, 2025, University of California,
Santa Barbara, CA, USA

Edited by

Niv Gilboa



Editors

Niv Gilboa 

Ben-Gurion University of the Negev, Beer-Sheva, Israel
gilboan@bgu.ac.il

ACM Classification 2012

Mathematics of computing → Information theory; Theory of computation → Computational complexity and cryptography; Security and privacy → Cryptography

ISBN 978-3-95977-385-0

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/978-3-95977-385-0>.

Publication date

September, 2025

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.ITC.2025.0

ISBN 978-3-95977-385-0

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

- Christel Baier (TU Dresden, DE)
- Roberto Di Cosmo (Inria and Université Paris Cité, FR)
- Faith Ellen (University of Toronto, CA)
- Javier Esparza (TU München, DE)
- Holger Hermanns (Universität des Saarlandes, Saarbrücken, DE and Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Wadern, DE)
- Daniel Král' (Leipzig University, DE and Max Planck Institute for Mathematics in the Sciences, Leipzig, DE)
- Sławomir Lasota (University of Warsaw, PL)
- Meena Mahajan (Institute of Mathematical Sciences, Chennai, IN – Chair)
- Chih-Hao Luke Ong (Nanyang Technological University, SG)
- Eva Rotenberg (Technical University of Denmark, Lyngby, DK)
- Pierre Senellart (ENS, Université PSL, Paris, France)
- Alexandra Silva (Cornell University, Ithaca, US)

ISSN 1868-8969

<https://www.dagstuhl.de/lipics>

Contents

Preface	
<i>Niv Gilboa</i>	0:vii
Steering Committee	
.....	0:ix
Organization	
.....	0:xi
 Papers	
Amortized Locally Decodable Codes for Insertions and Deletions	
<i>Jeremiah Blocki and Justin Zhang</i>	1:1–1:23
Powerful Primitives in the Bounded Quantum Storage Model	
<i>Mohammed Barhoush and Louis Salvail</i>	2:1–2:20
Leakage-Resilience of Shamir’s Secret Sharing: Identifying Secure Evaluation Places	
<i>Jihun Hwang, Hemanta K. Maji, Hai H. Nguyen, and Xiuyu Ye</i>	3:1–3:20
Time-Space Tradeoffs of Truncation with Preprocessing	
<i>Krzysztof Pietrzak and Pengxiang Wang</i>	4:1–4:10
Information-Theoretic Random-Index PIR	
<i>Sebastian Kolby, Lawrence Roy, Jure Sternad, and Sophia Yakoubov</i>	5:1–5:15
MetaDORAM: Info-Theoretic Distributed ORAM with Less Communication	
<i>Brett Hemenway Falk, Daniel Noble, and Rafail Ostrovsky</i>	6:1–6:23
Linear-Time Secure Merge in $O(\log \log n)$ Rounds	
<i>Mark Blunk, Paul Bunn, Samuel Dittmer, Steve Lu, and Rafail Ostrovsky</i>	7:1–7:23
On the Definition of Malicious Private Information Retrieval	
<i>Bar Alon and Amos Beimel</i>	8:1–8:23
Revocable Encryption, Programs, and More: The Case of Multi-Copy Security	
<i>Prabhanjan Ananth, Saachi Mutreja, and Alexander Poremba</i>	9:1–9:23
Multi-Source Randomness Extraction and Generation in the Random-Oracle Model	
<i>Sandro Coretti, Pooya Farshim, Patrick Harasser, and Karl Southern</i>	10:1–10:23
New Results in Share Conversion, with Applications to Evolving Access Structures	
<i>Tamar Ben David, Varun Narayanan, Olga Nissenbaum, and Anat Paskin-Cherniavsky</i>	11:1–11:23
Key-Agreement with Perfect Completeness from Random Oracles	
<i>Noam Mazon</i>	12:1–12:11



Preface

The sixth Conference on Information-Theoretic Cryptography (ITC 2025) took place from August 16–17, 2025, at the University of California, Santa Barbara, USA. The general chair was Prabhanjan Ananth, and the program chair was Niv Gilboa. As in previous editions, the conference was held in cooperation with the International Association for Cryptologic Research (IACR).

In its sixth year, ITC continued its mission of uniting the cryptography and information theory communities, and advancing research in all aspects of information-theoretic techniques for cryptography and security.

We received a total of 18 submissions, maintaining a high standard of quality. Following our tradition, we facilitated interactive and anonymous discussions with the authors to clarify technical issues. With the assistance of external reviewers, the program committee selected 12 papers for presentation. The proceedings contain the revised versions of these papers. The revisions were not reviewed, and the authors bear full responsibility for the content.

This year, we continued the tradition of featuring “spotlight talks” and invited talks that highlight exciting developments in the field. These talks highlighted notable recent papers from top conferences such as Eurocrypt 2021, Crypto 2024 and TCHES 2025. The talks were given by both established researchers and students. The spotlight and invited talks aimed to provide a comprehensive overview of the most significant recent advances in information-theoretic cryptography.

We are deeply grateful to everyone who contributed to the success of the 6th ITC conference. Our sincere thanks go out to the authors who submitted their papers. We extend our heartfelt thanks to the PC members and external reviewers for their dedicated efforts in providing thorough reviews, insightful discussions, and expert opinions. We are deeply indebted to the steering committee, particularly Benny Applebaum, for his invaluable guidance. Special thanks are also due to the previous PC chairs, especially Divesh Aggarwal, for sharing their experience and providing answers to numerous questions. Lastly, we extend our gratitude to all the invited speakers, presenting authors, and participants who devoted their time and energy to ensuring the success of this conference.

Niv Gilboa

■ Steering Committee

- Benny Applebaum (Chair, Tel-Aviv University)
- Ivan Damgård (Aarhus University)
- Yevgeniy Dodis (New York University)
- Yuval Ishai (Technion)
- Ueli Maurer (ETH Zurich)
- Kobbi Nissim (Georgetown)
- Krzysztof Pietrzak (IST Austria)
- Manoj Prabhakaran (IIT Bombay)
- Adam Smith (Boston University)
- Yael Tauman Kalai (MIT and Microsoft Research New England)
- Stefano Tessaro (University of Washington)
- Vinod Vaikuntanathan (MIT)
- Hoeteck Wee (ENS Paris)
- Daniel Wichs (Northeastern University and NTT Research)
- Mary Wootters (Stanford)
- Chaoping Xing (Nanyang Technological University)
- Moti Yung (Google)



Organization

General chair

- Prabhanjan Ananth (University of California, Santa Barbara)

Program chair

- Niv Gilboa (Ben-Gurion University of the Negev)

Program committee

- Akinori Kawachi (Mie University)
- Alex Bredariol Grilo (CNRS/Sorbonne Université)
- Alexander Golovnev (Georgetown University)
- Amos Beimel (Ben-Gurion University of the Negev)
- Antonia Wachter-Zeh (Technical University of Munich)
- Dana Dachman-Soled (University of Maryland)
- Daniele Venturi (Sapienza University of Rome)
- Eylon Yogev (Bar-Ilan University)
- Hemanta Maji (Purdue University)
- Kevin Yeo (Google Research)
- Krzysztof Pietrzak (IST Austria)
- Luisa Siniscalchi (Technical University of Denmark)
- Maciej Obremski (National University of Singapore)
- Mahdi Cheraghchi (University of Michigan Ann Arbor)
- Mohammad Hajiabadi (University of Waterloo)
- Noah Golowich (MIT)
- Or Sheffet (Bar-Ilan University)
- Pasin Manurangsi (Google Research)
- Salim El Rouayheb (Rutgers University)
- Sandro Coretti (IOG)
- Stefano Tessaro (University of Washington)
- Wei-Kai Lin (University of Virginia)

External reviewers

Alper Cakan, Ananta Mukherjee, Anasuya Acharya, Angus Gruen, Changrui Mu, Gal Arnon, Ilan Komargodski, Jihun Hwang, Samuel King, Saswata Mukherjee, Sidhant Saraogi, Xiuyu Ye.



Amortized Locally Decodable Codes for Insertions and Deletions

Jeremiah Blocki 

Department of Computer Science, Purdue University, West Lafayette, IN, USA

Justin Zhang 

Department of Computer Science, Purdue University, West Lafayette, IN, USA

Abstract

Locally Decodable Codes (LDCs) are error correcting codes which permit the recovery of any single message symbol with a low number of queries to the codeword (the locality). Traditional LDC tradeoffs between the rate, locality, and error tolerance are undesirable even in relaxed settings where the encoder/decoder share randomness or where the channel is resource-bounded. Recent work by Blocki and Zhang initiated the study of Hamming *amortized* Locally Decodable Codes (aLDCs), which allow the local decoder to amortize their number of queries over the recovery of a small subset of message symbols. Surprisingly, Blocki and Zhang construct asymptotically ideal (constant rate, constant amortized locality, and constant error tolerance) *Hamming aLDCs* in private-key and resource-bounded settings. While this result overcame previous barriers and impossibility results for Hamming LDCs, it is not clear whether the techniques extend to Insdel LDCs. Constructing Insdel LDCs which are resilient to insertion and/or deletion errors is known to be even more challenging. For example, Gupta (STOC'24) proved that no Insdel LDC with constant rate and error tolerance exists even in relaxed settings.

Our first contribution is to provide a Hamming-to-Insdel compiler which transforms any amortized Hamming LDC that satisfies a particular property (consecutive interval querying) to amortized Insdel LDC while asymptotically preserving the rate, error tolerance and amortized locality. Prior Hamming-to-Insdel compilers of Ostrovsky and Paskin-Cherniavsky (ICITS'15) and Block et al. (FSTTCS'20) worked for arbitrary Hamming LDCs, but incurred an undesirable polylogarithmic blow-up in the locality. Our second contribution is a construction of an ideal amortized Hamming LDC which satisfies our special property (consecutive interval querying) in the relaxed settings where the sender/receiver share randomness or where the channel is resource bounded. Taken together, we obtain ideal Insdel aLDCs in private-key and resource-bounded settings with constant amortized locality, constant rate and constant error tolerance. This result is surprising in light of Gupta's (STOC'24) impossibility result which demonstrates a strong separation between locality and amortized locality for Insdel LDCs.

2012 ACM Subject Classification Mathematics of computing → Coding theory

Keywords and phrases Amortized Locally Decodable Codes, Insertion and Deletion Errors

Digital Object Identifier 10.4230/LIPIcs.ITC.2025.1

Related Version *Full Version*: <https://arxiv.org/abs/2507.03141>

Funding *Jeremiah Blocki*: Supported in part by NSF CAREER Award CNS-2047272.

1 Introduction

Locally Decodable Codes (LDCs) are a variant of error correcting codes which provide single-symbol recovery with highly efficient query complexity over a (possibly corrupted) codeword. Specifically, a LDC over alphabet Σ is defined as a tuple of algorithms the encoding algorithm $\text{Enc} : \Sigma^k \rightarrow \Sigma^n$ and the local decoder $\text{Dec}^{by} : [k] \rightarrow \Sigma$. Here, k denotes the message length, n denotes the codeword length and $\tilde{\mathbf{y}} \in \Sigma^n$ is a (possibly corrupted) codeword — for any message $\mathbf{x} \in \Sigma^k$, its corresponding codeword is $\mathbf{y} = \text{Enc}(\mathbf{x})$ and $\tilde{\mathbf{y}}$ denotes a (possibly



© Jeremiah Blocki and Justin Zhang;
licensed under Creative Commons License CC-BY 4.0
6th Conference on Information-Theoretic Cryptography (ITC 2025).
Editor: Niv Gilboa; Article No. 1; pp. 1:1–1:23



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

corrupted) copy of $\mathbf{y}$. The local decoder $\text{Dec}^{\tilde{\mathbf{y}}}(i)$ is a randomized oracle algorithm that is given an index $i \in [k] := \{1, 2, \dots, k\}$ as input and has oracle access to a (possibly corrupted) codeword $\tilde{\mathbf{y}}$. $\text{Dec}^{\tilde{\mathbf{y}}}(i)$ attempts to output the i 'th symbol of the message $\mathbf{x}$ after making *at most* ℓ queries to the symbols of $\tilde{\mathbf{y}}$. We require that for any index $i \in [k]$ and *any* $\tilde{\mathbf{y}}$ that is not too far from the uncorrupted codeword $\mathbf{y}$ (i.e. $\text{Dist}(\mathbf{y}, \tilde{\mathbf{y}}) \leq \delta|\mathbf{y}|$ for a chosen distance metric Dist) that $\text{Dec}^{\tilde{\mathbf{y}}}(i)$ outputs the correct symbol $\mathbf{x}[i]$ with probability at least $1 - \varepsilon$. Broadly, codes which satisfy the above parameters are called $(\ell, \delta, \varepsilon)$ -LDCs. The main parameters of interest are the *query locality* ℓ , *error tolerance* δ and *rate* k/n . Ideally, we would like an LDC with constant locality, constant error tolerance, and constant rate simultaneously.

Within the classical setting of worst-case Hamming errors, i.e. when the distance metric is chosen to be the Hamming distance metric and error patterns are introduced in a worst-case fashion, the trade-offs between the locality, error tolerance, and rate has been extensively studied. Even so, achievable parameters in this setting remain undesirably sub-optimal. Katz and Trevisan show that any LDC with constant locality $\ell \geq 2$ and constant error tolerance $\delta > 0$ must have non-constant rate [17], immediately ruling out the existence of an ideal LDC for worst case Hamming errors. Moreover, the best known constructions with constant locality and error tolerance have super-polynomial rate [24, 11]. Several relaxations have been made, such as allowing the local decoder to reject corrupted codewords [13, 2, 9, 18, 10], the assumption that the encoder and decoder share a cryptographic key [20, 14, 15] that is unknown to a probabilistic polynomial time channel, or the assumption that the channel is resource-bounded in other ways [1, 6]. Yet, even under these assumptions, we do not have an LDC with constant locality, constant error tolerance, and constant rate. The literature on LDCs and relaxed variants is vast. We provide an expanded discussion of the work related to LDCs in the full version of the paper [8].

Blocki and Zhang [7] introduced the notion of an *amortized* locally decodable code where the total number of queries to the codeword can be amortized by the total number of message bits recovered (see Definition 2). The local decoder for an amortized LDC takes as input a range $[L, R]$ (we will assume $R - L + 1 \geq \kappa$ for a suitable parameter κ) instead of a single index i and attempts to output the entire substring $\mathbf{x}[L, R]$ instead of the single message symbol x_i . The amortized locality is $\ell/(R - L + 1)$ i.e., the total number of queries ℓ divided by the number of message symbols recovered. Surprisingly, they showed how to construct ideal amortized locally decodable codes in relaxed settings where the channel is resource-bounded or where the sender/receiver share a secret key. In particular, given any (sufficiently large) interval $[L, R] \subseteq [k]$ the local decoder can recover all of the corresponding message symbols $x[L], \dots, x[R]$ while making at most $O(R - L)$ queries to the corrupted codeword. They also observed that the Hadamard code can have amortized locality approaching 1. By contrast, Katz and Trevisan show that codes with locality $\ell < 2$ do not exist unless the message length is a constant [17].

Given the surprising success in construction ideal amortized LDCs for Hamming channels, even in relaxed settings, it is natural to wonder whether or not it is possible to obtain similar results for insertions and deletions (*Insdel*) channels.

Is it possible to obtain constant amortized locality, constant error tolerant, and constant rate Insdel LDCs when the channel is oblivious, resource-bounded, or when the sender/receiver share a cryptographic key?

A construction of an ideal amortized LDC for Insdel channels would be especially surprising since when compared to known lowerbounds of Hamming LDCs, the current state of affairs is even worse for Insdel LDCs. Blocki et al. [5] proved the first separation result between Insdel and Hamming LDCs by proving that linear Insdel LDCs with locality $\ell = 2$ do not exist. This

stands in contrast to Hamming LDCs, where constructions do exist, albeit with exponential rate (e.g. the Hadamard code). They additionally provide a exponential rate lowerbound for any constant locality Insdel LDC. Their lower bound holds in both traditional and relaxed (private-key and oblivious channel) settings, making progress towards a conjecture raised by Block et al. [5] that constant locality Insdel LDCs do not exist. Recently, Gupta resolved this conjecture by showing that constant query deletion codes do not exist by providing a randomized, oblivious adversarial deletion strategy [12]. The results of Gupta extend even to relaxed settings where the sender/receiver share random coins or where the channel is resource bounded i.e., even in relaxed settings it is impossible to construct constant query Insdel LDCs. Thus, it would be especially surprising to construct an ideal amortized LDC for Insdel channels even in relaxed settings with shared randomness or resource bounded channels.

On the side of constructions, Ostrovsky and Paskin-Cherniavsky provide the first InsDel LDC via a compilation of a Hamming LDCs into an Insdel LDCs [22] with a polylogarithmic blowup to the query complexity and only a constant blowup to the error-tolerance and rate. Their construction was revisited by Block et al. [4] who reproved and extended their results to construct Locally Correctable Codes (LCCs), a variant of LDCs which considers the local decodability of any codeword symbol. At a high level, their compiler takes a Hamming LDC encoding of the message, partitions it into blocks, and applies an Insdel code to each block. The local decoder then simulate the underlying Hamming decoder's queries by retrieving and decoding the corresponding Insdel-encoded block to recover the queried Hamming code symbol, where each answered Hamming query incurs a polylogarithmic query blowup.

Unfortunately, a black box approach of instantiating their Insdel LDC constructions with an amortizable Hamming LDC does not preserve the amortized locality. Even if all symbols recovered by the decoder are used, the amortized locality will still suffer a polylogarithmic blowup from the Hamming decoder simulation. Thus, it is unclear whether the Insdel LDC constructions of Ostrovsky and Paskin-Cherniavsky or Block et al. are amortization friendly.

1.1 Our Contributions

We provide the first framework for constructing amortized Insdel LDCs by modifying the compiler of Block et al. [4] to be amenable to amortization. Our resulting compiler takes in an amortized Hamming LDC, whose decoder satisfies a properties we define as *consecutive interval querying*, and outputs an Insdel amortized LDC with asymptotically equivalent parameters (amortized locality, rate, and error tolerance). More specifically, given an amortized Hamming LDC with constant rate, constant error tolerance, and constant amortized locality, whose decoder queries contiguous blocks of polylogarithmic size instead of individual message symbols, our compiler will yield an amortized Insdel LDC with constant rate, constant error tolerance, and constant amortized locality. While prior Insdel LDC compilers suffer from a polylogarithmic blow-up in query complexity of the underlying Hamming LDC, our amortized compiler preserves amortized locality as well as rate and error tolerance (up to constant factors). Thus, given any ideal Hamming amortized LDC which satisfies our consecutive interval querying property, we obtain an ideal Insdel amortized LDC achieving constant rate, error tolerance, and amortized locality.

► **Theorem** (Informal, see Corollary 13). *If there exists ideal (constant rate, constant error tolerance, and constant amortized locality) Hamming amortized LDCs whose decoder only queries consecutive blocks of size $\Omega(\text{polylog } n)$, then there exists ideal Insdel amortized LDCs (with asymptotically equivalent parameters).*

Furthermore, in the private-key setting, we construct the first consecutive interval querying Hamming LDC with constant rate, error tolerance, and constant *amortized* locality. We leave it as an open question whether one can construct a consecutive interval querying Hamming LDC with constant *amortized* locality in the information theoretic setting. However, using our updated Hamming-to-Insdel compiler we obtain an ideal amortized Insdel LDC construction (constant rate, constant error tolerance and constant amortized locality) in the private-key setting. Specifically, the local decoder $\text{Dec}^{\tilde{\mathbf{Y}}}(a, b)$ makes at most $O(b - a + \text{polylog}(n))$ queries to the corrupted codeword¹ $\tilde{\mathbf{Y}}$ and (whp) outputs all of the correct symbols $x[a], x[a + 1], \dots, x[b]$ in the interval $[a, b]$. As long as $b - a \geq \text{polylog}(n)$ the amortized locality is $O(1)$ per symbol recovered.

► **Theorem** (Informal, see Theorem 16 and Corollary 19). *For any $t, n \in \mathbb{N}$ such that $t \mid n$, there exists ideal private-key Hamming amortized LDCs with codeword length n whose decoder only queries consecutive blocks of size t and has constant amortized locality whenever decoding any consecutive interval of size at least $\omega(t \log n)$. This implies that there exists ideal private-key Insdel amortized LDCs.*

We show further that our private-key construction implies an ideal amortized Insdel LDC in resource-bounded settings i.e. when the channel has a bounded amount of some resource (e.g. space, circuit-depth, cumulative memory, etc.).

► **Theorem** (Informal, see Corollary 24). *If there exists ideal private-key Insdel amortized LDCs, there exists ideal resource-bounded Insdel amortized LDCs (with asymptotically equivalent parameters).*

Our results demonstrate definitively strong separations between locality and amortized locality for Insdel LDCs. In particular, Gupta [12] rules out the existence of *any* Insdel LDC with constant locality and error tolerance, even in relaxed settings and even with exponential rate.

1.2 Technical Overview

Recalling the BBGKZ Compiler

Block et al. give a construction of a compiler taking any Hamming LDC and compiling it into an Insdel LDC[4]. We refer to their construction as the BBGKZ compiler. In the compiled encoding procedure, the message $\mathbf{x}$ is first encoded under the Hamming LDC as codeword $\mathbf{y}$, where $\mathbf{y}$ is then partitioned into size τ *Hamming* blocks $\mathbf{y} = \mathbf{w}_1 \circ \dots \circ \mathbf{w}_B$. Each Hamming block $\mathbf{w}_j$ along with its index j is encoded using a constant rate Insdel code with the special property stating that the relative Hamming weight of every logarithmic-sized interval is at most $2/5$ (The Schulman-Zuckerman-code [23] satisfies this property). Lastly, each block is pre/postpended with a zero buffer $0^{\phi\tau}$ of length $\phi\tau$, where $\phi \in (0, 1)$, resulting in the *Insdel* block $\mathbf{w}'_j$ and the Insdel codeword $\mathbf{Y} = \mathbf{w}'_1 \circ \dots \circ \mathbf{w}'_B$. Intuitively, the zero buffers $0^{\phi\tau}$ help to identify the beginning/end of each block after insertions/deletions.

Then, the compiled Insdel local decoding procedure, when given oracle access to the (possibly corrupted) Insdel codeword $\tilde{\mathbf{Y}}$, attempts to simulate the Hamming decoder with access to a corrupted codeword $\tilde{\mathbf{y}}$ that is close to the Hamming Codeword $\mathbf{y}$ in Hamming Distance. Specifically, when the Hamming decoder queries symbol i of its expected oracle

¹ $\tilde{\mathbf{Y}}$ is the corrupted codeword output by the PPT-bounded channel who does not have the secret key under the constraint that the edit distance fraction between $\mathbf{Y}$ and $\tilde{\mathbf{Y}}$ is at most 2δ .

access to (possibly corrupted) codeword $\tilde{\mathbf{y}}$, the Insdel decoding procedure will identify the block $\mathbf{w}_j$ which contains $\mathbf{y}[i]$, (attempt to) locate the corresponding padded Insdel block $\mathbf{w}'_j$, undo the padding, run the Insdel decoder to recover the pre-compiled block $\mathbf{w}_j$, and return the corresponding symbol $\tilde{\mathbf{y}}[i]$. This query simulation process is facilitated by a subroutine we refer to as the **RecoverBlock** procedure which (attempts to) locate a particular Insdel block $\mathbf{w}'_j$ in the presence of corruptions using a noisy binary search procedure.

Challenge 1: RecoverBlock is not Amortizable

Our first challenge in constructing amortizable Insdel LDCs is the observation that **RecoverBlock** performs asymptotically strictly more queries than the symbols it recovers. Since the codeword $\mathbf{Y}$ may contain insertions and deletions, in order for **RecoverBlock** to locate the Hamming decoder's queries, it performs a *noisy binary search* procedure and makes $O(\tau + \tau \text{polylog}(n))$ queries to recover the block corresponding to each query made by the Hamming decoder. The Hamming decoder only uses one of these symbols, but even if the decoder utilized all τ symbols from the recovered symbols, the amortized locality is still *at least* $O(\text{polylog}(n))$ per symbol recovered. Thus, we cannot achieve constant amortized locality by using the BBGKZ compiler as a black box.

Our insight to address this challenge is that **RecoverBlock** can be modified to recover a *consecutive interval* of blocks rather than just a single block. We refer to this modified procedure as **RecoverBlocks**, where on input block interval $[a, b] := \{a, \dots, b\}$, **RecoverBlocks** will recover pre-compiled blocks $\mathbf{w}_a, \dots, \mathbf{w}_b$ and make at most $O((b - a + 1)\tau + \tau \text{polylog}(n))$ queries. Hence, as long as $b - a + 1 \geq \text{polylog}(n)$, the ratio of recovered symbols to queried symbols will be constant.

Challenge 2: Proving Correctness of RecoverBlocks

We show that the modified procedure **RecoverBlocks** preserves correctness. That is, a majority of the (corrupted) Insdel blocks $\tilde{\mathbf{w}}'_j$ satisfy the property running the search **RecoverBlocks** with any interval $[a, b]$ that contains $j \in [a, b]$ will yield the original Hamming block $\mathbf{w}_j$ except with negligible probability. Block et al. [4] proved that the procedure **RecoverBlock** will correctly find any *local-good block* and that most blocks will be *local-good* as long as the number of corruptions is bounded. The main technical challenge results from an inherent mismatch for **RecoverBlocks** because an interval $[a, b]$ may contain a mixture of blocks that are (resp. *are not*) locally-good. We show that **RecoverBlocks** successfully recovers any *local-good blocks* in its interval except with negligible probability.

We prove that correctness in a similar manner to the analysis of Block et al. [4]. We also utilize the notion of *local-good blocks*, where if block $\mathbf{w}'_j$ is (θ, γ) -local-good, it will have at most $\gamma\tau$ corruption with the additional desirable property that any block interval $[a, b]$ surrounding the block, i.e. $j \in [a, b]$, has at least $(1 - \theta) \times |b - a + 1|$ blocks that also have at most $\gamma\tau$ corruption. Utilizing analysis from [4] we can pick our parameters to ensure that at least $(1 - \delta_h)$ fraction of our blocks are (θ, γ) -local good. What we prove is that if block $\mathbf{w}'_j$ is (θ, γ) -local good and if $j \in [a, b]$ then calling **RecoverBlocks** for the interval $[a, b]$ will recover the original (uncorrupted) $\mathbf{w}_j$ with high probability along with any other (θ, γ) -good block in the interval $[a, b]$. Thus, for any partition $[a_1, b_1], \dots, [a_z, b_z]$ of $[B]$ calling **RecoverBlocks** with each interval $[a_i, b_i]$ would recover the corrupt (uncorrupted) $\mathbf{w}_j$ for at least $(1 - \delta_h)B$ blocks. We will use this observation to reason about the correctness of our decoding procedure. Intuitively, we can view the compiled Insdel decoder as simulating the Hamming decoder with a corrupted codeword whose fractional hamming distance is at most δ_h from the original codeword.

Challenge 3: Consecutive Hamming Queries Needed For Amortized Decoding

The next challenge is utilizing the **RecoverBlocks** procedure to obtain amortized locality in the Insdel local decoding procedure. Similar to the BBGKZ compiler, our Insdel decoder with oracle access to (possibly corrupted) codeword $\tilde{\mathbf{Y}}$ simulates the compiled Hamming decoder's oracle access to (possibly corrupted) codeword $\tilde{\mathbf{y}}$. Our hope is that if the underlying Hamming LDC is amortizable then the compiled Insdel LDC will also be amortizable. In particular, we hope to group the Hamming decoder's queries $q_1, \dots, q_\ell$ together into corresponding (disjoint) index intervals $[a_1, b_1], \dots, [a_p, b_p]$ such that (1) $\sum_i |b_i + 1 - a_i| = \Theta(\ell)$, (2) $\{q_1, \dots, q_\ell\} \subseteq \bigcup_i [a_i, b_i]$, and (3) the average size $\frac{1}{p} \sum_i |b_i + 1 - a_i|$ of each interval is large enough we can amortize over the calls to **RecoverBlocks** e.g., we need to ensure that the average interval size is at least $\frac{1}{p} \sum_i |b_i + 1 - a_i| = \Omega(\text{polylog} n)$. Unfortunately, in general, we cannot place any such structural guarantees on the queries made by an (amortizable) Hamming LDC.

We address this problem by introducing the notion of a *t-consecutive interval querying* Hamming LDCdecoder which is amortization-friendly under our Insdel compiler. Intuitively, the local decoder for a *t-consecutive interval querying* Hamming LDC accesses the (possibly corrupted) codeword $\tilde{\mathbf{y}}$ by querying for *t*-sized intervals instead of individual message symbols i.e., the decoder may submit the query $[i, i + t - 1]$ and the output will be $\tilde{\mathbf{y}}[i, i + t - 1]$. In more detail the Hamming decoder takes as input an interval $[L, R]$ of message symbols we would like to recover, makes queries for *t*-sized intervals of the (possibly corrupted) codeword $\tilde{\mathbf{y}}$ and then attempts to output $\mathbf{x}[L, R]$. Note that if the Hamming decoder makes ℓ/t interval queries to $\tilde{\mathbf{y}}$ this still corresponds to locality ℓ as the total number of codeword symbols accessed is $t \times (\ell/t) = \ell$.

Correspondingly, when the Hamming decoder is *t-consecutive interval querying*, our Insdel decoder will make at most $2\lceil \ell/t \rceil$ calls to **RecoverBlocks** on intervals of size *t*. The resulting query complexity is roughly $\ell \times \left(\frac{\tau \log^3 n}{t} + O(1) \right)$, and so, in other words, the query complexity blow-up is by a factor of $O\left(\frac{\tau \log^3 n}{t} \right)$. This implies that if there exists a *t-consecutive interval querying* Hamming decoder for $t = \Omega(\tau \log^3 n)$ with constant amortized locality, our compiled Insdel construction will have constant amortized locality. Thus, the primary question is whether or not it is possible to construct *t-consecutive interval querying* Hamming LDCs with constant amortized locality. We leave this as an open question for worst-case errors, but show that it is possible in settings where the channel is computationally/resource restricted.

Challenge 4: A *t-consecutive interval querying* aLDC in Private-Key Settings

We present a *t-consecutive interval querying* private-key amortized Hamming LDC (paLDC) construction with constant rate, constant amortized locality, and constant error tolerance i.e. ideal parameters. As a starting point we use the one-time private-key Hamming LDC constructed by Ostrovsky et al. [21]. Blocki and Zhang show that their construction is amortizable and that their construction may be extended to multi-round secret-key reuse with cryptographic building blocks [7]. For simplicity, we focus on the single round construction, which is information-theoretically secure as long as the channel is computationally bounded.

The single round construction first generates the secret key as a randomly drawn permutation π and a random string $\mathbf{r}$. To encode the message $\mathbf{x}$, it is partitioned into blocks $\mathbf{x} = \mathbf{e}_1 \circ \dots \circ \mathbf{e}_B$ and each block $\mathbf{e}_j$ is encoded as $\mathbf{e}'_j = \text{Enc}(\mathbf{e}_j)$ by a constant rate, constant error tolerance Hamming code **Enc**. Lastly, the bits of the encoded blocks are permuted by π and the random string $\mathbf{r}$ is xor'd i.e. the resulting codeword is $\mathbf{y} = \pi(\mathbf{e}'_1 \circ \dots \circ \mathbf{e}'_B) \oplus \mathbf{r}$. The local decoder can recover any block $\mathbf{e}_j$ by finding the indices of its corresponding encoded

$$\begin{array}{ccccc}
x = & e_1 & \circ \cdots \circ & e_B \\
& \downarrow \text{Enc} & & \downarrow \text{Enc} & \downarrow \text{Enc} \\
& e'_1 & \circ \cdots \circ & e'_B \\
& \parallel & & \parallel & \parallel \\
& (\epsilon_1 \circ \cdots \circ \epsilon_\beta) & \circ \cdots \circ & (\epsilon_{(B-1)\beta+1} \circ \cdots \circ \epsilon_{B\beta}) \\
& \downarrow \pi(\cdot) \oplus r & & \downarrow \pi(\cdot) \oplus r & \downarrow \pi(\cdot) \oplus r \\
y = & \left(\epsilon_{\pi(1)} \circ \cdots \circ \epsilon_{\pi(\beta)} \right. & \circ \cdots \circ & \left. \epsilon_{\pi((B-1)\beta+1)} \circ \cdots \circ \epsilon_{\pi(B\beta)} \right) & \oplus r
\end{array}$$

■ **Figure 1** An overview of our private-key Hamming aLDC decoder, satisfying correct decoding (of any e_j Hamming block) and consecutive interval querying (of any $\epsilon_{\pi(r)}$ sub-block) simultaneously.

block e'_j , reversing the permutation and random masking, and running the Hamming code decoder Dec to recover e_j . Unfortunately, the local decoder is not t -consecutive interval querying since its queries are not in a contiguous interval. The main issue is that the application of the permutation π removes any pre-existing block structure in the codeword y i.e., if the local decoder wants to query for consecutive (pre-permutation) code symbols $(e'_1 \circ \cdots \circ e'_B)[s+1, s+\ell]$, it would have to query $y[\pi(s+1)], \dots, y[\pi(s+\ell)]$, which are no longer consecutive with high probability.

A first attempt to remedy this issue is to permute at the block level instead of the bit level i.e. we draw permutation π over $[B]$ and set the codeword instead as $y = (e'_{\pi(1)} \circ \cdots \circ e'_{\pi(B)}) \oplus r$. While this admits contiguous access when recovering any block e_j it defeats the original purpose of the permutation. The permutation π (and the one-time pad r) ensure that (whp) an adversarial channel cannot produce too many corruptions in any individual block. If we permute at the block level then it is trivial for an attacker to corrupt entire blocks. In particular, the adversary could focus their entire error budget to corrupt symbols at the start of the codeword. This simple strategy will always corrupt a constant fraction of blocks e'_j . Thus, the proposed code will not satisfy correct decoding.

Our primary insight is that we can apply the permutation at an intermediate “sub-block” granularity to ensure correctness and simultaneously support consecutive interval queries. In particular, each encoded block e'_j , for $j \in [B]$, is further partitioned into β sub-blocks $e'_j = (\epsilon_{(j-1)\beta+1} \circ \cdots \circ \epsilon_{j\beta})$ and the permutation is applied over sub-blocks ϵ_r , for $r \in [B\beta]$, rather than the Hamming blocks e'_j themselves i.e. the codeword is

$$y = (\epsilon_{\pi(1)} \circ \cdots \circ \epsilon_{\pi(\beta)} \circ \cdots \circ \epsilon_{\pi(B\beta)}) \oplus r.$$

Intuitively, while a constant fraction of the sub-blocks ϵ_r can still be corrupted with non-negligible probability, as long as the parameter β is suitably large (e.g., $\beta = O(\text{polylog} n)$), we can ensure that, except with negligible probability, the overall error in each encoded block e'_j will still be at most a δ -fraction. Thus, except with negligible probability we can recover e_j (the uncorrupted content in block j) by making β consecutive interval queries to obtain ϵ_r for each $r \in [(j-1)\beta+1, j\beta]$. This is exactly what we needed to achieve amortization with our Ideal Insdel compiler.

The above construction assumes that the sender and receiver share a secret key and only allows the sender to transmit one encoded message. However, if the channel is probabilistic polynomial time (PPT) then the construction can be extended allow for the sender to transmit

multiple messages using standard cryptographic tools e.g., pseudo-random functions[21, 7]. Furthermore, if we assume stronger resource bounds on the channel (space, sequential depth etc...) then one can use cryptographic puzzles to completely eliminate the requirement for a secret key using ideas from prior works e.g., [1, 6, 3].

Putting it All Together: Ideal Insdel aLDCs in Relaxed Settings

With the building blocks established prior, the aLDC Hamming-to-Insdel compiler and the consecutive interval querying, ideal Hamming paLDC, we construct the first ideal Insdel aLDC within relaxed settings (private-key and resource-bounded). Naturally, we first construct a private-key, ideal Insdel aLDC by directly using our compiler on the t -consecutive interval querying, ideal Hamming paLDC. For compiler block size τ , our construction achieves constant amortized locality by setting $t = \Omega(\tau \log^3 n)$ as discussed previously.

The remaining challenge lies in showing that the compiler's correctness is preserved in private-key settings. Intuitively, we argue via a reduction to the security of the underlying Hamming paLDC. Suppose there exists a probabilistic polynomial time adversary $\mathcal{A}$ who introduces an Insdel corruption pattern into compiled codeword $\mathbf{Y}$ of their chosen message $\mathbf{x}$ that causes a decoding error with non-negligible probability. Then, we can construct a probabilistic polynomial time adversary $\mathcal{B}$ who introduces a Hamming error pattern into (pre-compiled) codeword $\mathbf{y}$ by (1) using our compiler procedure to transform Hamming codeword $\mathbf{y}$ into the compiled Insdel codeword $\mathbf{Y}$, (2) Giving compiled Insdel codeword $\mathbf{Y}$ to adversary $\mathcal{A}$ and receiving corrupted Insdel codeword $\tilde{\mathbf{Y}}$, and finally, (3) constructing the pre-compiled codeword $\tilde{\mathbf{y}}$ by the same simulation process done by the compiled decoder via the `RecoverBlocks` procedure. To streamline the reduction argument, we draw inspiration from Block and Blocki [3] and repackage our compiler into a convenient form for our reduction. All together, we have an Ideal Insdel paLDC.

Next, we generalize the paLDC construction to construct an aLDC for resource-bounded settings (raLDC). Similarly, our first step is constructing a t -consecutive interval querying Hamming raLDC. To do this, we make use of the Hamming private-key-to-resource-bounded LDC compiler by Ameri et al. [1], which was observed by Blocki and Zhang to be amortizable with only a constant blow-up to the amortized locality [7]. At a high level, their compiler makes use of a cryptographic primitive known as a *cryptographic puzzle* to embed the secret key within the codeword.

More specifically, cryptographic puzzles consists of two algorithms `PuzzGen` and `PuzzSolve`. On input seed $\mathbf{c}$, `PuzzGen` outputs a puzzle $\mathbf{Z}$ with solution is $\mathbf{c}$ i.e. $\text{PuzzSolve}(\mathbf{Z}) = \mathbf{c}$. The security requirement states that adversary $\mathcal{A}$ in a defined algorithm class $\mathbb{R}$, cannot solve the puzzle $\mathbf{Z}$ and cannot even distinguish between tuples $(\mathbf{Z}_0, \mathbf{c}_0, \mathbf{c}_1)$ and $(\mathbf{Z}_1, \mathbf{c}_0, \mathbf{c}_1)$ for random $\mathbf{c}_i$ and $\mathbf{Z}_i = \text{PuzzGen}(\mathbf{c}_i)$.

Then, the Hamming raLDC compiler takes a paLDC $(\text{Gen}_p, \text{Enc}_p, \text{Dec}_p)$, and on on message $\mathbf{x}$ will (1) generate the secret key $\text{sk} \leftarrow \text{Gen}_p(1^\lambda; \mathbf{c})$ with some random coins $\mathbf{c}$ (2) Compute the paLDC encoding of the message $\mathbf{y}_p \leftarrow \text{Enc}_{p, \text{sk}}(\mathbf{x})$ (3) Compute an encoding $\mathbf{y}_*$ of the puzzle $\mathbf{Z}$ using a code with low locality that recovers the entire message (known as an LDC*) and (4) output $\mathbf{y} = \mathbf{y}_p \circ \mathbf{y}_*$. Intuitively, a resource-bounded adversary will not be able to solve the puzzle, while the raLDC decoder `Dec` will have sufficient resources to solve the puzzle, recover the secret key sk , and run the amortized local decoder $\text{Dec}_{p, \text{sk}}$.

We make an additional observation that if the compiler is instantiated with a t -consecutive interval querying paLDC, then the resulting raLDC will also be t -consecutive interval querying. Thus, when applied to our amortized Hamming-to-Insdel compiler with $t = \Omega(\tau \log^3 n)$, we achieve constant amortized locality. Lastly, the same reduction argument as in the ideal

Insdel paLDC construction may be used, with an additional subtle detail of allowing sufficient resources for the reduction. That is, for the class of resource-bounded channels/adversaries $\mathbb{R}$ that our Insdel aLDC is secure against, the underlying Hamming aLDC must be secure against a wider class of adversaries $\overline{\mathbb{R}}$, where $\overline{\mathbb{R}}$ include adversaries $\mathcal{A} \in \mathbb{R}$ whom can additionally compute the simulation process in the private-key reduction.

1.3 Preliminaries

Notation

Let $x \leftarrow y$ represent a variable assignment of x as y and $x \xleftarrow{\$} X$ denote a uniform random assignment of x from space X . Let $\circ$ denote the concatenation function. For $l \leq r$, $[l, r] := \{l, l+1, \dots, r\}$ denotes an interval from l to r inclusive of its ends points. Similarly, $[l, r) := \{l, l+1, \dots, r-1\}$ denotes an interval from l to r exclusive of the right endpoint. Over an alphabet Σ , bolded notations $\mathbf{x}$ denote vectors/words while non-bolded $x \in \Sigma$ denote single symbols. For a word $\mathbf{x}$, $x[i]$ denotes the symbol of $\mathbf{x}$ at index i , while $\mathbf{x}[l, r] := (x[l], x[l+1], \dots, x[r])$ denotes the subword of $\mathbf{x}$ projected to indices in interval $[l, r]$. It will also be useful to define a short-hand for retrieving symbols from “blocks” of a word $\mathbf{x}$: $\mathbf{x}\langle i \rangle_b := \mathbf{x}[(i-1)b+1, ib]$ for any block size b and index i .

For any words $\mathbf{x}, \mathbf{y} \in \Sigma^n$, $\text{Ham}(\mathbf{x}, \mathbf{y})$ denotes the hamming distance between $\mathbf{x}$ and $\mathbf{y}$ i.e. $|\{i \in [n] : x_i \neq y_i\}|$. Further, for $\mathbf{z} \in \Sigma^*$, $\text{ED}(\mathbf{x}, \mathbf{z})$ denotes the edit distance between $\mathbf{x}$ and $\mathbf{z}$ i.e. the number of symbol insertions and deletions to transform $\mathbf{x}$ into $\mathbf{z}$. We say $\mathbf{x}$ and $\mathbf{y}$ (resp. $\mathbf{x}$ and $\mathbf{z}$) are δ -hamming-close (resp. δ -edit-close) when $\text{Ham}(\mathbf{x}, \mathbf{y}) \leq \delta|\mathbf{x}|$ (resp. $\text{ED}(\mathbf{x}, \mathbf{z}) \leq 2\delta|\mathbf{x}|$).

We recall the formal definitions of an LDC and an amortized LDC.

► **Definition 1 (LDC).** A (n, k) -code $\mathcal{C} = (\text{Enc}, \text{Dec})$ (over Σ) is a (ℓ, δ, ϵ) -locally decodable code (LDC) for hamming errors (resp. insDel errors) if for every $\mathbf{x} \in \Sigma^k$, $\tilde{\mathbf{y}} \in \Sigma^*$ such that $\tilde{\mathbf{y}}$ is δ -hamming-close (resp. δ -edit-close) to $\text{Enc}(\mathbf{x})$, and every index $i \in [k]$, we have $\Pr[\text{Dec}^{\tilde{\mathbf{y}}}(i) = x[i]] \geq 1 - \epsilon$, and $\text{Dec}^{\tilde{\mathbf{y}}}$ makes at most ℓ queries to $\tilde{\mathbf{y}}$.

► **Definition 2 (aLDC [7]).** A (n, k) -code $\mathcal{C} = (\text{Enc}, \text{Dec})$ (over Σ) is a $(\alpha, \kappa, \delta, \epsilon)$ -amortizeable LDC (aLDC) for hamming errors (resp. insDel errors) if for every $\mathbf{x} \in \Sigma^k$, $\tilde{\mathbf{y}} \in \Sigma^*$ such that $\tilde{\mathbf{y}}$ is δ -hamming-close (resp. δ -edit-close) to $\text{Enc}(\mathbf{x})$, and every interval $[L, R] \subseteq [k]$ or size $R - L + 1 \geq \kappa$ we have $\Pr[\text{Dec}^{\tilde{\mathbf{y}}}(L, R) = (x_i : i \in [L, R])] \geq 1 - \epsilon$, and $\text{Dec}^{\tilde{\mathbf{y}}}$ makes at most $\alpha \times (R - L + 1)$ queries to $\tilde{\mathbf{y}}$.

Throughout the paper, it will be assumed codes are over the binary alphabet i.e. $\Sigma = \{0, 1\}$ unless specified otherwise. The primary metrics of interest for Hamming/Insdel aLDCs are the rate k/n , amortized locality α , and the error/edit tolerance δ .

1.4 Organization

We start with reintroducing and modifying the encoding and decoding procedures of the BBGKZ compiler in Section 2. The modification will take various steps organized into the following subsections: in Subsection 2.1, the notion of good blocks and intervals are introduced to analyze our new **RecoverBlocks** procedure in Subsection 2.2. We then present the overall construction of our Hamming-to-Insdel aLDC compiler in Subsection 2.3.

In Section 3, we present our construction of an ideal Insdel aLDCs in settings where the encoder/decoder share a secret-key (paLDC). More specifically, this construction first relies on a modified Hamming paLDC construction in Subsection 3.1, where the local decoder

performs consecutive interval queries. The ideal Hamming paLDC is then compiled into our ideal Insdel paLDC in Subsection 3.2. Lastly, in Section 4, we present our construction of an ideal Insdel aLDCs in settings where the channel is resource-bounded (raLDC). Section 4 follows a symmetric structure to the previous section: we present a consecutive interval querying Hamming raLDC construction in Subsection 4.1, which is then compiled into an ideal Insdel raLDC construction in Subsection 4.2.

2 The Hamming aLDC to Insdel aLDC Compiler

Our results crucially rely on a modified BBGKZ compiler suited for amortized local decoding. We start with the encoding procedure. Interestingly, we will not need to make any changes to the encoding procedure of the BBGKZ compiler, which takes as parameters the block size $\tau \in \mathbb{N}$ and padding rate $\phi \in (0, 1)$: the message $\mathbf{x}$ will first be encoded into a codeword $\mathbf{y}$ using a Hamming LDC. Next, the codeword $\mathbf{y}$ is broken up into blocks of equal size τ as $\mathbf{y} = \mathbf{w}_1 \circ \dots \circ \mathbf{w}_B$, where each block and its index ($i \circ \mathbf{w}_i$ is 1) encoded with a specifically chosen Insdel code 2) pre-and-post-pended with $\phi\tau$ many 0s. The result is the overall codeword $\mathbf{Y}$. Formally, we describe the encoder Enc as first computing the Hamming codeword $\mathbf{y}$, then applying a Hamming-to-Insdel compiler EncCompile to transform it into an Insdel codeword $\mathbf{Y}$. The encoder Enc and compiler EncCompile are formally described below.

Let $\mathcal{C}_h = (\text{Enc}_h, \text{Dec}_h)$ be a Hamming LDC. Let $\mathcal{C}_{\text{insdel}} = (\text{Enc}_{\text{insdel}}, \text{Dec}_{\text{insdel}})$ be an Insdel binary code.

EncCompile $_\phi(\mathbf{y})$

Input: Hamming codeword $\mathbf{y} \in \{0, 1\}^m$, padding rate $\phi \in (0, 1)$.

Output: Insdel codeword $\mathbf{Y} \in \{0, 1\}^n$.

1. Parse $\mathbf{y} = \mathbf{w}_1 \circ \dots \circ \mathbf{w}_{|\mathbf{y}|/\tau}$ where $\mathbf{w}_j \in \{0, 1\}^\tau$ for all $j \in [|\mathbf{y}|/\tau]$.
2. Compute $\mathbf{w}'_j \leftarrow 0^{\phi\tau} \circ \text{Enc}_{\text{insdel}}(j \circ \mathbf{w}_j) \circ 0^{\phi\tau}$ for all $j \in [|\mathbf{y}|/\tau]$.
3. Output $\mathbf{Y} = \mathbf{w}'_1 \circ \dots \circ \mathbf{w}'_{|\mathbf{y}|/\tau}$.

Enc($\mathbf{x}$)

Parameters: Block size $\tau \in \mathbb{N}$, Padding rate $\phi \in (0, 1)$.

Input: message $\mathbf{x} \in \{0, 1\}^k$.

Output: Insdel codeword $\mathbf{Y} \in \{0, 1\}^n$.

1. Compute $\mathbf{y} \leftarrow \text{Enc}_h(\mathbf{x})$.
2. Output $\mathbf{Y} \leftarrow \text{EncCompile}(\mathbf{y})$.

Our main modifications to the BBGKZ compiler will be in the construction of the decoder Dec . Recall that the original BBGKZ decoder simulated oracle access to the Hamming LDC decoder Dec_h by recovering the Hamming blocks corresponding to the requested queries of Dec_h . Since insertions and deletions in the corrupted codeword $\tilde{\mathbf{Y}}$ modify the structure of the blocks, this simulation is underpinned by a *noisy binary search* procedure we denote as the **RecoverBlock** procedure. More specifically, the **RecoverBlock** procedure on input block index j will iteratively cut an initial search radius $[1, \tilde{n}]$ by a fractional amount, until the search radius is a constant size larger than the block size τ . To decide how to cut the search radius on each iteration, the procedure samples $N = \theta(\log^2 n)$ blocks indices by decoding N noisy $\mathbf{w}'_{j_1}, \dots, \mathbf{w}'_{j_b}$ blocks within the middle of the search radius to $(j_b \circ \mathbf{w}_{j_b})$. On the block indices $j_1, \dots, j_b$, their median j_{med} is compared to the desired block index j : depending if $j < j_{\text{med}}$, **RecoverBlock** will cut off a fraction of the front or the back of the search radius.

Since the search radius decreases by a constant fraction each iteration, there are $O(\log n)$ iterations, where on each iteration, $\theta(\log^2 n)$ blocks are read of $O(\tau)$ size. Thus, the query complexity to answer a single query of Dec_h is $O(\tau \log^3 n)$.

Unfortunately, amortization is not feasible with the current Hamming decoder simulation method. Intuitively, since the **RecoverBlock** procedure must query a multiplicative polylog n factor to recover τ symbols, the amortized locality will be at least $\Omega(\text{polylog } n)$.

Our solution will recover multiple blocks at once using the observation that the **RecoverBlocks** procedure can be extended from recovering a single block to an interval of contiguous blocks. That is, we will modify the noisy binary search procedure to take in a block interval $a \leq b \in [B]$, and we change the stopping condition to be linear to the interval size $b - a + 1$. Further, the consideration on how to shrink the search radius will take into account the respective endpoints a and b of the interval rather than a single block j .

While this change is relatively straight forward on paper, the analysis for the noisy binary search procedure BBGKZ compiler does not immediately follow for our modified procedure **RecoverBlocks**. Thus, to prove correctness for the modified **RecoverBlocks** procedure, we reintroduce the good block analysis from Block et al. [4], and modify their proof structure to take into account the recovery of an interval of blocks instead of a single block.

2.1 Good Blocks and Intervals

In this subsection, we introduce good block notation to analyze the modifications we will make to the BBGKZ decoder and its sub-routine **RecoverBlock**.

Additional Notation

First, we fix notation consistent with the encoder definition above for the rest of the paper. We will refer to the Hamming encoding $\mathbf{y}$ in step 1 as the *Hamming encoding* and the final codeword $\mathbf{Y}$ as the *Insdel encoding* or simply as the *encoding*. Let $\mathbf{x} \in \{0, 1\}^k$ be the message of size k and let $\mathbf{y} = \text{Enc}_h(\mathbf{x}) \in \{0, 1\}^m$ be the Hamming encoding with size m . Let $B = m/\tau$ be the number of blocks. Define β such that the rate of the overall encoding is $1/\beta$. Similarly, define β_{insdel} such that the rate of the Insdel code $\mathcal{C}_{\text{insdel}}$ is $1/\beta_{\text{insdel}}$. Then, the rate of the Hamming encoding is $(\beta_{\text{insdel}} \log B + 2\phi)/\beta$. Let δ_h and δ_{insdel} be the error tolerances of the Hamming encoding and the Insdel code respectively. Let each $\mathbf{w}_j = \mathbf{y}_{\langle j \rangle_\tau} \in \{0, 1\}^\tau$ denote *Hamming block* j . Similarly, each $\mathbf{w}'_j \in \{0, 1\}^{\beta\tau}$ denotes the *Insdel block/ block* j . Then, the overall codeword is $\mathbf{Y} \in \{0, 1\}^n$ has length $n = \beta m$. We consider any corrupted codeword $\tilde{\mathbf{Y}} \in \{0, 1\}^{\tilde{n}}$ such that $\text{ED}(\mathbf{Y}, \tilde{\mathbf{Y}}) \leq 2\delta n$.

We define the *block decomposition* as a mapping from codeword symbols to their blocks post corruption.

► **Definition 3.** A *block decomposition* of $\tilde{\mathbf{Y}}$ is a non-decreasing mapping $\phi : [\tilde{n}] \rightarrow [B]$.

By definition, the pre-image of each block decomposition defines a partition of $[\tilde{n}]$ into B intervals $\{\phi^{-1}(j) : j \in [B]\}$. Each of these intervals define the block structure, where the j 'th interval defines the j 'th block in the corrupted codeword. In other words, for each $j \in [B]$, $\phi^{-1}(j)$ are all the indices corresponding to block j .

As in the BBGKZ compiler, our modified decoder will rely on a noisy recovery process, where the decoder will need to locate blocks without knowing the block boundaries that have been modified by insdel errors. For any search interval $[l, r] \subseteq [\tilde{n}]$, it will be helpful to consider its corresponding block interval i.e. the smallest interval $[L, R - 1] \subseteq [B]$ such that $[l, r] \subseteq [\tau L, \tau(R - 1)]$. Block intervals are defined formally below.

► **Definition 4** (Block Interval). *The block interval of an interval $I = [l, r] \subseteq [\tilde{n}]$ is defined as $\bigcup_{i=l}^{r-1} \phi^{-1}(\phi(i)) \subseteq [\tilde{n}]$. An interval I is a block interval if the block interval of I is itself. Equivalently, every block interval has the form $\mathcal{BI}[a, b] := \bigcup_{i=a}^b \phi^{-1}(j)$ for some $a, b \in [B]$.*

We say a block is *good* if it does not have too many edit corruptions. Intuitively, good blocks will correspond to Insdel blocks that are correctly decodable, except with negligible probability.

► **Definition 5** (γ -Good Block). *For $\gamma \in (0, 1)$ and $j \in [B]$, a block j is γ -good (with respect to $\tilde{Y}$) if $\text{ED}(\tilde{Y}[\phi^{-1}(j)], \mathbf{w}'_j) \leq \gamma\tau$. Otherwise, block j is γ -bad.*

Good blocks may be naturally extended to *good intervals*, where the summed edit corruptions of each block within the interval is not too much. Additionally, we constrain the number of bad blocks in any good interval.

► **Definition 6** ((θ, γ) -Good Interval). *A block interval $\mathcal{BI}[a, b]$ is (θ, γ) -good (with respect to $\tilde{Y}$) if the following hold:*

1. $\sum_{j=a}^b \text{ED}(\tilde{Y}[\phi^{-1}(j)], \mathbf{w}'_j) \leq \gamma\tau(b - a + 1)$.
 2. *The number of γ -bad blocks in the block interval $\mathcal{BI}[a, b]$ is at most $\theta \times (b - a + 1)$.*
- Otherwise, the interval is (θ, γ) -bad.*

Lastly, we define the notion of a locally good block, which captures the idea that any interval including such a block must also be good.

► **Definition 7** ((θ, γ) -Local Good Block). *For $\theta, \gamma \in (0, 1)$, block $j \in [B]$ is (θ, γ) -local good (with respect to $\tilde{Y}$) if for every $a, b \in [B]$ such that $a \leq j \leq b$, the interval $\mathcal{BI}[a, b]$ is (θ, γ) -good. Otherwise, block j is (θ, γ) -locally bad.*

2.2 Extending RecoverBlock to RecoverBlocks

We start with recalling specific details of the **RecoverBlock** procedure. The **RecoverBlock** procedure for recovering a single block takes as input the desired block index $j \in [B]$ and the initial search radius $[1, \tilde{n}]$. The procedure will iteratively split the search radius into three contiguous, linear-sized segments, a beginning, a middle, and an end segment, where it will choose to either cut the beginning or the end segment. To decide which, it randomly samples $N = \theta(\log^2 n)$ indices $i_1, \dots, i_N$ within the middle segment and recover their respective block indices $j_1, \dots, j_N$. More specifically, for each index i_b , a subroutine **Block-Decode** is called which queries a $O(\tau)$ interval around i_b to decode the noisy block containing $(j \circ \mathbf{w}_j)$. The median j_{med} of these retrieved indices is then calculated, and, depending if $j < j_{\text{med}}$ or $j \geq j_{\text{med}}$, cuts either the end or beginning segment from the search interval respectively. The binary search continues until the search radius is linear in the size of the block for a sufficiently small constant.

Lastly, an interval decoding procedure is then run over the symbols in the final search interval, where it is guaranteed to recover any local-good block located within the interval. The subroutines used, **Block-Decode** and the final interval decoding, are based on local and global variations of the SZ-code decoding algorithm, finding the pre/post-pended 0's on the Insdel blocks to locate the blocks in between.

Intuitively, we may modify **RecoverBlock** to recover an interval $[a, b] \subseteq [B]$ of contiguous Hamming blocks instead of a single block by stopping the search once its search radius is linear to the *size of the interval* versus the size of a single block. Lastly, we run a procedure we call **Sim-RecoverBlock** to recover each block $j \in [a, b]$ within the final search

radius $[l, r]$. If we naively call $\text{RecoverBlock}^{\tilde{Y}}(l, r, q)$ repeatedly for every block $q \in [a, b]$ then each call to RecoverBlock will execute its own noisy binary search procedure in the interval $[l, r]$ resulting in redundant queries which would blow up our amortized locality i.e., the total number of queries would be too large i.e. $\Omega((l - r)\text{polylog}(l - r))$. Instead, we simply make $l - r + 1$ queries to $\tilde{Y}$ to recover the entire substring $\tilde{Y}[l, r]$ and then run $\text{Sim-RecoverBlock}(\tilde{Y}[l, r], l, r, q)$ for each $q \in [a, b]$ where $\text{Sim-RecoverBlock}(\tilde{Y}[l, r], l, r, q)$ uses the hardcoded string $\tilde{Y}[l, r]$ to simulate the execution of $\text{RecoverBlock}^{\tilde{Y}}(l, r, q)$. We call this overall extended procedure RecoverBlocks , and present the construction below (modifications to the RecoverBlock procedure are in blue).

$\text{RecoverBlocks}^{\tilde{Y}}(l, r, a, b)$

Parameters: Block size $\tau \in \mathbb{N}, \gamma \in (0, 1)$
Input: Search range $l < r \in [\tilde{n} + 1]$, **block range** $a \leq b \in [B]$
Output: pre-compiled blocks $\mathbf{s} \in \{0, 1\}^{(b-a+1)\tau}$
 Let $c = 36(\beta - \gamma)$.
 Let $\rho = \min \left\{ \frac{1}{4} \times \frac{\beta - \gamma}{\beta + \gamma}, 1 - \frac{3}{4} \times \frac{\beta + \gamma}{\beta - \gamma} \right\}$.

1. While $r - l > c(b - a + 1)\tau$:
 - a. Let $m_1 \leftarrow (1 - \rho)l + \rho r$, and $m_2 \leftarrow \rho l + (1 - \rho)r$.
 - b. For $p = 1, \dots, N = \theta(\log^2 n)$,
 - i. randomly sample $i_p \xleftarrow{\$} [m_1, m_2]$
 - ii. set $S \leftarrow \text{Block-Decode}^{\tilde{Y}}(i_p)$
 - iii. If $S = \perp$, ignore and continue
 - iv. Else, parse $(j_p, \mathbf{w}_{j_p}) \leftarrow S$
 - c. Let $j_{\text{med}} \leftarrow \text{median}(j_1, \dots, j_N)$ (ignore $\perp$)
 - d. If $a > j_{\text{med}}$, set $l = m_1$. Otherwise, set $r = m_2$.
2. Make $(l - r + 1)$ queries to recover $\tilde{Y}[l, r]$.
3. Output $[\text{Sim-RecoverBlock}(\tilde{Y}[l, r], l, r, q) : q \in [a, b]]$.

The first step is to prove that RecoverBlocks on input $(1, \tilde{n} + 1, a, b)$ for any block interval $[a, b] \subseteq [B]$ will return all Hamming blocks $\mathbf{w}_j$ within that interval, given that the blocks are good. We will then argue that by a careful setting of parameters, the number of correctly recovered blocks is sufficient for an overall correct decoding process. We refer to the full version [8] for the proof of the following theorem and lemma on the correctness and query complexity of the RecoverBlocks procedure.

► **Theorem 8.** Let $\mathcal{P} = \{[a_i, b_i] : 1 \leq a_i \leq b_i < a_{i+1} \leq B\}$ be any partition of $[B]$. Then, let $\mathbf{s}_j^{[a_i, b_i]}$ be the random variable defined as the output of $\text{RecoverBlocks}^{\tilde{Y}}(1, \tilde{n} + 1, a_i, b_i) \langle j - a_i + 1 \rangle_\tau$. If $\delta = \frac{\delta_h \phi \gamma}{8\beta(1+1/\theta)}$, then

$$\Pr \left[\sum_{[a_i, b_i] \in \mathcal{P}} \sum_{j=1}^{b_i - a_i + 1} 1 \left(\mathbf{s}_j^{[a_i, b_i]} \neq \mathbf{w}_j \right) \geq \delta_h B \right] \leq \text{negl}(n),$$

where the probability is taken over the joint distribution $\left\{ \mathbf{s}_j^{[a_i, b_i]} : [a_i, b_i] \in \mathcal{P}, j \in [b_i - a_i + 1] \right\}$.

► **Lemma 9.** For any word $\tilde{Y} \in \{0, 1\}^{\tilde{n}}$, interval $[a, b] \subseteq [B]$, and constants $\beta, \gamma > 0$, on input $(1, \tilde{n} + 1, a, b)$, RecoverBlocks has query complexity $O((b - a)\tau + \tau \log^3 n)$.

2.3 The Amortized Insdel Decoder

We construct an amortized Insdel local decoder Dec utilizing **RecoverBlocks** to simulate oracle access for the underlying Hamming decoder Dec_h . Our Insdel decoder Dec follows a similar procedure to the BBGKZ compiler decoder. Dec calls the underlying Hamming decoder Dec_h , and on its queries $i_1, \dots, i_q \in [m]$ to the (possibly corrupted) Hamming codeword $\tilde{\mathbf{y}}$, will respond by calling **RecoverBlocks** to recover the corresponding Hamming blocks and queried symbols. Correctness of our procedure will follow from Theorem 8, which roughly states that the symbols returned by **RecoverBlocks** are equivalent to what Dec_h expects from true oracle access to the (possibly corrupted) Hamming codeword $\tilde{\mathbf{y}}$.

Our next step is to ensure that the queries made by our Insdel decoder amortize over the **RecoverBlocks** calls by restricting our Hamming decoder to only make contiguous queries, where each contiguous interval is of a fixed size. Suppose the Hamming decoder Dec_h is guaranteed to make queries in contiguous intervals of size t , say $[u_1 + 1, u_1 + t], \dots, [u_p + 1, u_p + t] \subseteq [m]$ for some t . Intuitively, when the Insdel decoder Dec is simulating the Hamming decoder's queries with **RecoverBlocks**, it can then make $O(p)$ calls to **RecoverBlocks** to recover codeword intervals of size $O(t)$. For large enough t , the number of **RecoverBlocks** calls $O(p)$ decreases and the Insdel decoder's queries will amortize.

We formally describe a Hamming decoder with this desirable property t -consecutive interval querying Hamming decoder.

► **Definition 10** (Consecutive Interval Querying). *Consider any (m, k) -code $\mathcal{C} = (\text{Enc}, \text{Dec})$ that is a $(\alpha, \kappa, \delta, \varepsilon)$ -aLDC. For any word $\tilde{\mathbf{y}}$, interval $[L, R] \subseteq [k]$ with $R - L + 1 \geq \kappa$, and random coins r , let $\text{Query}(\tilde{\mathbf{y}}, [L, R], r) := \{i_1, \dots, i_q\} \subseteq [\tilde{n}]$ denote the codeword indices queries when $\text{Dec}^{\tilde{\mathbf{y}}}(L, R)$ is ran with randomness r . Code $\mathcal{C}$ and decoder Dec are t -consecutive interval querying if the set $\text{Query}(\tilde{\mathbf{y}}, [L, R], r)$ can be partitioned into q/t disjoint intervals $[u_1, v_1], \dots, [u_{q/t}, v_{q/t}]$ of size t i.e.,*

1. $v_j - u_j + 1 = t$ for all $j \leq q/t$,
2. $\text{Query}(\tilde{\mathbf{y}}, [L, R], r) = \bigcup_{j \leq q/t} [u_j, v_j]$, and
3. $[u_{j_1}, v_{j_1}] \cap [u_{j_2}, v_{j_2}] = \emptyset$ for all $j_1 \neq j_2$.

We now show how to construct the amortized local decoder Dec using **RecoverBlocks** and a t -consecutive interval querying Hamming aLDC decoder Dec_h for a value of t to be specified in the analysis. We choose the SZ-code as the Insdel code $\mathcal{C}_{\text{insdel}}$, which has constant rate $1/\beta_{\text{insdel}} = \Omega(1)$ and constant error-tolerance $\delta_{\text{insdel}} = \theta(1)$.

► **Construction 11.** Let $\mathcal{C}_h = (\text{Enc}_h, \text{Dec}_h)$ be a t -consecutive interval querying Hamming LDC such that τ divides t . Let $\mathcal{C}_{\text{insdel}} = (\text{Enc}_{\text{insdel}}, \text{Dec}_{\text{insdel}})$ be an Insdel binary code.

EncCompile $_{\phi}(\mathbf{y})$

Input: Hamming codeword $\mathbf{y} \in \{0, 1\}^m$, padding rate $\phi \in (0, 1)$.

Output: Insdel codeword $\mathbf{Y} \in \{0, 1\}^n$.

1. Parse $\mathbf{y} = \mathbf{w}_1 \circ \dots \circ \mathbf{w}_{|\mathbf{y}|/\tau}$ where $\mathbf{w}_j \in \{0, 1\}^{\tau}$ for all $j \in [|\mathbf{y}|/\tau]$.
2. Compute $\mathbf{w}'_j \leftarrow 0^{\phi\tau} \circ \text{Enc}_{\text{insdel}}(j \circ \mathbf{w}_j) \circ 0^{\phi\tau}$ for all $j \in [|\mathbf{y}|/\tau]$.
3. Output $\mathbf{Y} = \mathbf{w}'_1 \circ \dots \circ \mathbf{w}'_{|\mathbf{y}|/\tau}$.

Enc($\mathbf{x}$)

Parameters: Block size $\tau \in \mathbb{N}$, Padding rate $\phi \in (0, 1)$.

Input: message $\mathbf{x} \in \{0, 1\}^k$.

Output: Insdel codeword $\mathbf{Y} \in \{0, 1\}^n$.

1. Compute $\mathbf{y} \leftarrow \text{Enc}_h(\mathbf{x})$.
2. Output $\mathbf{Y} \leftarrow \text{EncCompile}(\mathbf{y})$.

Dec $^{\tilde{\mathbf{Y}}}(L, R)$

Parameters: Block size $\tau \in \mathbb{N}$, Padding rate $\phi \in (0, 1)$.

Input: $1 \leq L \leq R \leq \tilde{n}$

Output: word $\tilde{\mathbf{s}} \in \{0, 1\}^{(R-L+1)\tau}$

1. Suppose $\text{Dec}_h(L, R)$ queries disjoint intervals $[u_1, v_1], \dots, [u_p, v_p] \subseteq [m]$ of size t .
2. For each $r \in [p]$,
 - a. Compute $j \in [\lfloor n/t \rfloor]$ such that $[u_r, v_r] \subset [(j-1)t+1, (j+1)t]$.
 - b. Let $t' = \lceil t/\tau \rceil$ and compute

$$\mathbf{s}^{(j-1)} \leftarrow \text{RecoverBlocks}^{\tilde{\mathbf{Y}}}(1, \tilde{n}+1, (j-1)t'+1, jt')$$

$$\mathbf{s}^{(j)} \leftarrow \text{RecoverBlocks}^{\tilde{\mathbf{Y}}}(1, \tilde{n}+1, jt'+1, (j+1)t').$$

3. From $\{\mathbf{s}^{(j)}\}$, send Dec_h the bits corresponding to intervals $[u_1, v_1], \dots, [u_p, v_p]$.
4. Output the output of Dec_h .

Our Insdel decoder Dec processes the t -sized intervals $[u_1, v_1], \dots, [u_p, v_p] \subseteq [m]$ queried by the Hamming decoder Dec_h into a corresponding $t' = \lceil t/\tau \rceil$ -sized block interval inputs for RecoverBlocks (step 2.a). Note that for ease of presentation, the decoder always computes RecoverBlocks on two t' sized block intervals (step 2.b). The amortized locality can be optimized by a constant factor by calling RecoverBlocks on the exact block intervals queried by the Hamming decoder Dec_h . However, it will be convenient to assume the Insdel decoder Dec calls RecoverBlocks in a predictable manner for all inputs when proving correctness i.e. the input interval to RecoverBlocks is always $((j-1)t'+1, jt')$ for some j , and asymptotically, there is no change to the amortized locality.

► **Theorem 12.** *Let (m, k) -code $\mathcal{C}_h = (\text{Enc}_h, \text{Dec}_h)$ be a t -consecutive interval querying Hamming $(\alpha_h, \kappa_h, \delta_h, \varepsilon_h)$ -aLDC. Then, for any block size $\tau \in \Omega(\log n)$ such that $\tau \mid n$, $\mathcal{C} = (\text{Enc}, \text{Dec})$ in Construction 11 is a $(\alpha, \kappa, \delta, \varepsilon)$ -aLDC for $\alpha = O\left(\frac{\alpha_h \tau \log^3 n}{t}\right)$, $\kappa = \kappa_h$, $\delta = \Omega(\delta_h)$, and $\varepsilon = \varepsilon_h + \text{negl}(n)$.*

Proof. Suppose $\text{Dec}_h^{\tilde{\mathbf{Y}}}(L, R)$ queries disjoint intervals $[u_1, v_1], \dots, [u_p, v_p] \subseteq [m]$. For each $r \in [p]$, we compute RecoverBlocks on block intervals $[(j-1)t'+1, jt']$ and $[jt'+1, (j+1)t']$ such that $[u_r, v_r] \subset [(j-1)t+1, (j+1)t]$. Let $\mathcal{P} = \{[1, t'], [t'+1, 2t'], \dots, [B-t'+1, B]\}$ be a partition of $[B]$ in equal t' -sized intervals. Let $\mathbf{s}^{(j)}$ be the random variable defined as the output of $\text{RecoverBlocks}^{\tilde{\mathbf{Y}}}(1, \tilde{n}+1, (j-1)t'+1, jt')$. Define $\tilde{\mathbf{y}}$ as the random string defined by $\tilde{\mathbf{y}}_{(j)t'} = \mathbf{s}^{(j)}$ for each $j \in [\lceil n/t' \rceil]$. Then, since $\text{Ham}(\tilde{\mathbf{y}}_u, \mathbf{y}) \leq \delta_h m$ is implied

by the event $\sum_{[(j-1)t'+1, jt'] \in \mathcal{P}} \sum_{r=1}^t 1 \left(\mathbf{s}_r^{(j)} \neq \mathbf{w}_j \right) \leq \delta_h B$, by Theorem 8, $\Pr[\text{Ham}(\tilde{\mathbf{y}}, \mathbf{y}) \leq \delta_h m] \geq 1 - \text{negl}(n)$. Thus, from the view of Dec_h , it is interacting with $\tilde{\mathbf{y}}$ over partition $\mathcal{P}$, and so by definition, for any $R - L + 1 \geq \kappa_h$,

$$\Pr \left[\text{Dec}_h^{\tilde{\mathbf{y}}}(L, R) = (x_i : i \in [L, R]) \mid \text{Ham}(\tilde{\mathbf{y}}, \mathbf{y}) \leq \delta_h m \right] \geq 1 - \varepsilon_u.$$

By construction of Dec , for any $R - L + 1 \geq \kappa_h$,

$$\begin{aligned} \Pr \left[\text{Dec}^{\tilde{\mathbf{Y}}}(L, R) = (x_i : i \in [L, R]) \right] \\ \geq \Pr[\text{Ham}(\tilde{\mathbf{y}}_u, \mathbf{y}) \leq \delta_h m] \Pr \left[\text{Dec}_h^{\tilde{\mathbf{y}}_u}(L, R) = (x_i : i \in [L, R]) \mid \text{Ham}(\tilde{\mathbf{y}}_u, \mathbf{y}) \leq \delta_h m \right] \\ \geq (1 - \text{negl}(n)) \times (1 - \varepsilon_h) \geq 1 - \varepsilon_h - \text{negl}(n). \end{aligned}$$

Since $\mathcal{C}_h$ is t -consecutive interval querying which makes at most $\alpha_h(R - L + 1)$ queries, the decoder Dec calls **RecoverBlocks** at most $\frac{2\alpha_h(R-L+1)}{t}$ times on block intervals of size $t' = \lceil t/\tau \rceil$. By Lemma 9, the query complexity of Dec is

$$\alpha(R-L+1) = \frac{2\alpha_h(R-L+1)}{t} \times O\left(\left(\lceil t/\tau \rceil \times \tau\right) + \tau \log^3 n\right) \text{ and so } \alpha = O\left(\frac{\alpha_h \tau \log^3 n}{t}\right). \blacktriangleleft$$

If a t -consecutive interval querying, ideal Hamming aLDC does exist, we obtain the following corollary.

► **Corollary 13.** *If there exists a t -consecutive interval querying, ideal Hamming aLDC for $t = \Omega(\tau \log^3 n)$, then there exists an ideal Insdel aLDC.*

We construct t -consecutive interval querying, ideal Hamming aLDCs in private-key and resource-bounded settings and leave a construction for worst-case Hamming errors as an open question.

3 Ideal Insdel aLDCs in Private-key Settings

Given the compiler and resulting Theorem 12 in the previous section converting an ideal Hamming aLDC $\mathcal{C}_h$ to ideal Insdel aLDC $\mathcal{C}$ whenever $\mathcal{C}_h$ is $\Omega(\tau \log^3 n)$ -consecutive interval querying, we show that such an aLDC exist with private-key assumptions. We start by recalling the definition of a private-key aLDC (paLDC).

► **Definition 14** (paLDC [7]). *Let λ be the security parameter. A triple of probabilistic polynomial time algorithms $(\text{Gen}, \text{Enc}, \text{Dec})$ over Σ is a private $(\alpha, \kappa, \delta, \epsilon, q)$ -amortizeable LDC (paLDC) for Hamming errors (resp. Insdel errors) if*

- *for all keys $\text{sk} \in \text{Range}(\text{Gen}(1^\lambda))$ the pair $(\text{Enc}_{\text{sk}}, \text{Dec}_{\text{sk}})$ is a (n, k) code $\mathcal{C}$ over Σ , and*
- *for all $\tilde{\mathbf{y}} \in \Sigma^n$ and all $L, R \in [k]$ with $R - L + 1 \geq \kappa$ the local decoding algorithm $\text{Dec}_{\text{sk}}^{\tilde{\mathbf{y}}^{(h)}}(L, R)$ makes at most $(R - L + 1)\alpha$ queries to $\tilde{\mathbf{y}}$*
- *for all probabilistic polynomial time algorithms $\mathcal{A}$ there is a negligible function μ such that $\Pr[\text{paLDC-Sec-Game}(\mathcal{A}, \lambda, \alpha, \kappa, \delta, \epsilon, q) = 1] \leq \mu(\lambda)$, where the probability is taken over the randomness of $\mathcal{A}$, Gen , and **paLDC-Sec-Game**. The experiment **paLDC-Sec-Game** is defined as follows:*

paLDC-Sec-Game($\mathcal{A}, \lambda, \alpha, \kappa, \delta, q$)

The challenger generates secret key $\text{sk} \leftarrow \text{Gen}(1^\lambda)$. For q rounds, on iteration h , the challenger and adversary $\mathcal{A}$ interact as follows:

1. The adversary $\mathcal{A}$ chooses a message $\mathbf{x}^{(h)} \in \Sigma^k$ and sends it to the challenger.
2. The challenger sends $\mathbf{y}^{(h)} \leftarrow \text{Enc}_{\text{sk}}(\mathbf{x}^{(h)})$ to the adversary.
3. The adversary outputs $\tilde{\mathbf{y}}^{(h)} \in \Sigma^*$ that is δ -Hamming-close (resp. δ -edit-close) distance to $\mathbf{y}^{(h)}$.
4. If there exists $L^{(h)}, R^{(h)} \in [k]$ such that $R^{(h)} - L^{(h)} + 1 \geq \kappa$ and $\Pr \left[\text{Dec}_{\text{sk}}^{\tilde{\mathbf{y}}^{(h)}}(L^{(h)}, R^{(h)}) \neq \mathbf{x}[L^{(h)}, R^{(h)}] \right] > \varepsilon(\lambda)$, then this experiment immediately terminates and outputs 1.

If the experiment did not output 1 on any iteration h , then output 0.

Note that for paLDCs, we assume that the local decoder takes in a consecutive interval $[L, R]$ rather than an arbitrary subset Q as input. Blocki and Zhang show that explicit, ideal Hamming paLDC constructions exist for such decoders, and the existence of ideal paLDCs whose local decoders take in arbitrary subsets Q is left as an open problem.

3.1 A consecutive interval querying, Ideal Hamming paLDC

We construct a t -consecutive interval querying, ideal Hamming paLDC by modifying of the private-key LDC construction by Ostrovsky et al. [21]. Recall that the encoding procedure on input message $\mathbf{x}$ first partitions it into equal-sized blocks $\mathbf{x} = \mathbf{e}_1 \circ \dots \circ \mathbf{e}_B$, for all $j \in [B]$. Each block is then individually encoded as $\mathbf{e}'_j = \text{Enc}(\mathbf{e}_j)$ for each $j \in [B]$ by a constant rate, constant error tolerant, and constant alphabet size Hamming code encoding Enc (e.g. the Justesen code) to form the encoded message $\mathbf{x}' = \mathbf{e}'_1 \circ \dots \circ \mathbf{e}'_B$. Lastly, an additional secret-key permutation π and random mask $\mathbf{r}$ are applied i.e. the codeword is $\mathbf{y} = \pi(\mathbf{x}') \oplus \mathbf{r}$, which effectively makes the codeword look random from the view of a probabilistic polynomial time channel.

Blocki and Zhang [7] observe that the encoding procedure by Ostrovsky et al. is amenable to amortization when the recovered symbols are in a consecutive interval $[L, R]$. Their amortized local decoder, with access to the secret key, undoes the permutation π and random mask $\mathbf{r}$, recovers the contiguous blocks $\mathbf{e}'_{s+1}, \dots, \mathbf{e}'_{s+\ell}$ corresponding to requested $[L, R]$, decodes each block, and returns the corresponding queried symbols. Unfortunately, since we applied a permutation π in the encoding procedure, the amortized local decoder does not make consecutive interval queries to the (corrupted) codeword $\tilde{\mathbf{y}}$.

To add t -consecutive interval querying to the current decoder, we modify the encoding scheme permutation step to permute t -sized *sub-blocks* of the blocks $\mathbf{e}'_j$ in encoded message $\mathbf{x}' = \mathbf{e}'_1 \circ \dots \circ \mathbf{e}'_B$, rather than permuting individual bits. We highlight in blue significant changes made from the original paLDC presented by Blocki and Zhang.

► **Construction 15.** Suppose that $\mathcal{C} = (\text{Enc}_c, \text{Dec}_c)$ is a Hamming (A, a) -code over an alphabet Σ with rate R . Let $c = \log |\Sigma|$. Define $\mathcal{C}_p = (\text{Gen}_p, \text{Enc}_p, \text{Dec}_p)$ with message length k , codeword length $m = \frac{k}{R}$, block size cA , and sub-block size t (dividing cA) as follows:

$\text{Gen}(1^\lambda)$

1. Generate $\mathbf{r} \xleftarrow{\$} \{0, 1\}^m$ and uniformly random permutation $\pi : [m/t] \rightarrow [m/t]$.
2. Output $\text{sk} \leftarrow (\mathbf{r}, \pi)$.

$\text{Enc}_{\text{sk}}(\mathbf{x})$

Input: Message $\mathbf{x} \in \{0, 1\}^k$

Output: Codeword $\mathbf{y} \in \{0, 1\}^m$

1. Parse $(\mathbf{r}, \pi) \leftarrow \text{sk}$.
2. Let $\mathbf{x} \leftarrow \mathbf{e}_1 \circ \mathbf{e}_2 \circ \dots \circ \mathbf{e}_B$ where each $\mathbf{e}_s \in \Sigma^a$ (and $B = k/ca$).
3. For each $s \in [B]$:
 - a. set $\mathbf{e}'_s \leftarrow \text{Enc}_{\mathcal{C}}(\mathbf{e}_s)$, and
 - b. let $\mathbf{e}'_s = \epsilon_{(s-1)\beta+1} \circ \dots \circ \epsilon_{s\beta}$ where each $\epsilon_{s'} \in \{0, 1\}^t$ (and $\beta = cA/t$).
4. Output $\mathbf{y} = (\epsilon_{\pi(1)} \circ \dots \circ \epsilon_{\pi(B\beta)}) \oplus \mathbf{r}$.

$\text{Dec}_{\text{sk}}^{\tilde{\mathbf{y}}}(L, R)$

Input: Interval $[L, R] \subseteq [k]$

Output: word $\hat{\mathbf{x}}[L, R] \in \{0, 1\}^{R-L+1}$.

1. Parse $(\mathbf{r}, \pi) \leftarrow \text{sk}$ and interpret $\tilde{\mathbf{y}} = \tilde{\epsilon}_{\pi(1)} \circ \dots \circ \tilde{\epsilon}_{\pi(B\beta)}$ where each $\tilde{\epsilon}_j \in \{0, 1\}^t$.
2. Let the bits of $\mathbf{x}[L, R]$ lie in blocks $\mathbf{e}_{s+1}, \dots, \mathbf{e}_{s+\ell}$. For each $j = s+1, \dots, s+\ell$:
 - a. Compute $s'_1, \dots, s'_\beta$ such that $\pi(s'_r) = (s+j-1)\beta + r$ for all $r \in [\beta]$.
 - b. Query $\tilde{\mathbf{y}}$ to compute $\tilde{\epsilon}'_j \leftarrow \tilde{\epsilon}_{\pi(s'_1)} \circ \dots \circ \tilde{\epsilon}_{\pi(s'_\beta)}$.
 - c. Compute $\tilde{\mathbf{e}}_j \leftarrow \text{Dec}_{\mathcal{C}}(\tilde{\epsilon}'_j)$.

From $\tilde{\mathbf{e}}_{s+1}, \dots, \tilde{\mathbf{e}}_{s+\ell}$ output bits corresponding to interval $[L, R]$.

Any queries made by the decoder Dec are to the t -sized (corrupted) sub-blocks $\tilde{\epsilon}_{\pi(r)}$, so Construction 15 is t -consecutive interval querying. The main challenge is choosing the overall block size cA such that decoding error remains negligible. We show that by increasing the original setting of block size τ in the work of Blocki and Zhang [7] by a multiplicative t -factor, negligible decoding error follows.

► **Theorem 16.** *Let $\lambda \in \mathbb{N}$. In Construction 15, suppose the (A, a) -code $\mathcal{C}$ has (constant) rate R , (constant) error tolerance δ_p , and (constant) alphabet Σ with $c = \log |\Sigma|$. Then, for any $t \in \mathbb{N}$ such that $t \mid cA$ and $k \in \text{poly}(\lambda)$, $\mathcal{C}_p = (\text{Gen}_p, \text{Enc}_p, \text{Dec}_p)$ is a t -consecutive interval querying $(2/R, O(a), \theta(\delta_p), \text{negl}(\lambda), 1)$ -paLDC when $a = \omega(t \log \lambda)$.*

Proof. We show that the probability of an incorrect decoding $\Pr[\text{paLDC-Sec-Game}(\mathcal{A}, \lambda, \alpha, \kappa, \delta, 1) = 1]$ is negligible for $\delta = \theta(\delta_p)$ and any probabilistic polynomial time adversary $\mathcal{A}$. Define the event $\mathbf{Bad} = \bigcup_{j \in [k/ca]} \mathbf{Bad}_j$, where $\mathbf{Bad}_j$ is the event that $\mathbf{e}'_j$ has more than a δ_p fraction of errors. We show that the probability $\Pr[\mathbf{Bad}]$ is negligible.

Since the t -sized sub-blocks of the codeword are permuted and a random mask is applied, any errors added by a probabilistic polynomial time adversary $\mathcal{A}$ to the corrupted codeword $\tilde{\mathbf{y}}$ are added uniformly at random over t -bit intervals. This follows from generalizing the argument given by Lipton [19], where we interpret each sub-block as a t -bit symbol and the observation that the probability that event $\mathbf{Bad}$ occurs given that $\mathcal{A}$ does not apply errors in a t -bit interval is at most the probability that event $\mathbf{Bad}$ occurs given that $\mathcal{A}$ does apply errors in a t -bit interval.

Then, the number of errors in any given block $\mathbf{e}'_j$ follow a Hypergeometric($m/t, \delta m/t, cA/t$), which by the CDF bound of [15, 16], we have $\Pr[\mathbf{BAD}_j] < \exp\left(\frac{-2((\delta_p - \delta)(cA/t))^2 - 1}{(cA/t) + 1}\right)$. Thus, for $\delta_p > \delta$ and $a/t \in \omega(\log n)$, this probability is negligible. By a union bound, $\Pr[\mathbf{Bad}]$ is also negligible, so $\varepsilon \leq \Pr[\mathbf{Bad}] < \text{negl}(\lambda)$.

The proof of the query parameters $\alpha = 2/R$ and $\kappa = a$, is the same as in Theorem 16 in the work of Zhang and Blocki [7] (see full version [8]). ◀

We note that the poly-round paLDC construction of Blocki and Zhang [4] can also be made t -consecutive interval querying by the same technique of applying a higher-order permutation. We omit the construction and proof from this work since it will follow an almost-identical modification.

3.2 The Ideal Insdel paLDC Construction

We compile the t -consecutive interval querying, ideal Hamming paLDC in the prior subsection into an ideal Insdel paLDC using the Hamming-to-Insdel compiler **EncCompile** in Section 2.

► **Construction 17.** Suppose (m, k) -code $\mathcal{C}_p = (\text{Gen}_p, \text{Enc}_p, \text{Dec}_p)$ is a t -consecutive interval querying Hamming $(\alpha_p, \kappa_p, \delta_p, \varepsilon_p, q)$ -paLDC. Then define $\mathcal{C} = (\text{Gen}, \text{Enc}, \text{Dec})$ as $\text{Gen}(1^\lambda) := \text{Gen}_p(1^\lambda)$, $\text{Enc}_{\text{sk}}(\mathbf{x}) := \text{EncCompile}(\text{Enc}_{p,\text{sk}}(\mathbf{x}))$, and $\text{Dec}_{\text{sk}}^{\tilde{\mathbf{y}}}(L, R) := \text{Dec}_{p,\text{sk}}^{\tilde{\mathbf{y}}}(L, R)$.

We will need to show that the Hamming-to-Insdel compiler **EncCompile**, when used within the private-key setting, retains correctness. See the full version [8] for the reduction argument proving the following theorem.

► **Theorem 18.** Let $\mathcal{C}_p = (\text{Gen}_p, \text{Enc}_p, \text{Dec}_p)$ be a (m, k) -code that is a t -consecutive interval querying Hamming $(\alpha_p, \kappa_p, \delta_p, \varepsilon_p)$ -paLDC. Then, $\mathcal{C} = (\text{Gen}, \text{Enc}, \text{Dec})$ in Construction 17 with is a (n, k) -code that is an Insdel $(\alpha, \kappa, \delta, \varepsilon)$ -paLDC with $\alpha = O\left(\frac{\alpha_p \tau \log^3 n}{t}\right)$, $\kappa = \kappa_p$, $\delta = \Omega(\delta_p)$, and $\varepsilon = \varepsilon_p + \text{negl}(n)$.

By instantiating our private-key Insdel compiler with our t -querying Hamming paLDC, we construct an ideal Insdel paLDC.

► **Corollary 19.** If $\mathcal{C}_p$ of Construction 15 instantiated with a constant rate, constant error tolerant, and constant size alphabet code (e.g. a Justesen code), $t = \theta(\tau \log^3 n)$ and $\tau = \theta(\log n)$, then code $\mathcal{C}$ in Construction 17 is an (Ideal) Insdel $(O(1), O(\log^5 n), O(1), \text{negl}(n))$ -paLDC with constant rate.

4 Ideal Insdel aLDCs for Resource-bounded Channels

In this section, we present an ideal insdel aLDC in settings where the channel is bounded for some resource, such as parallel time or circuit depth. As in the construction of an ideal Insdel paLDC, our construction for resource-bounded channels will rely on a construction of a consecutive interval querying Hamming aLDC that will allow the compiler to amortize over its queries. Then, we formally prove the Compiler we constructed in Section 2 is secure in resource-bounded settings, which results in our ideal Insdel construction for resource-bounded channels.

We start by recalling the definition of an aLDC for resource-bounded channels (raLDC).

► **Definition 20 (raLDC [1]).** A (n, k) code $\mathcal{C} = (\text{Enc}, \text{Dec})$ is a $(\alpha, \kappa, \delta, \epsilon, \mathbb{R})$ -resource-bounded amortizeable LDC (raLDC) for hamming errors (resp. insDel errors) if for all $L, R \in [k]$ with $R - L + 1 \geq \kappa$ the local decoding algorithm $\text{Dec}_{\text{sk}}^{\tilde{\mathbf{y}}^{(h)}}(\cdot)$ makes at most $(R - L + 1)\alpha$ queries to $\tilde{\mathbf{y}} \text{Dec}^{\tilde{\mathbf{y}}}$ and for any $\tilde{\mathbf{y}} \in \Sigma^*$ and for any resource bounded algorithm $\mathcal{A} \in \mathbb{R}$, there is a negligible function μ such that $\Pr[\text{raLDC-Game}(\mathcal{A}, \lambda, \delta, \kappa) = 1] \leq \mu(\lambda)$, where the probability is taken over the randomness of $\mathcal{A}$, Gen , and paLDC-Sec-Game. The experiment **raLDC-Game** is defined as follows:

raLDC-Game($\mathcal{A}, \lambda, \delta, \kappa$)

1. The adversary $\mathcal{A}$ chooses a message $\mathbf{x} \in \{0, 1\}^k$ and sends it to the encoder.
2. The encoder sends $\mathbf{y} \leftarrow \text{Enc}_{\text{sk}}(\mathbf{x})$ to the adversary.
3. The adversary outputs $\tilde{\mathbf{y}} \in \{0, 1\}^*$ that is δ -hamming-close (resp. δ -edit-close) distance to $\mathbf{y}$.
4. If there exists $L, R \in [k]$ such that $\Pr [\text{Dec}^{\tilde{\mathbf{y}}}(L, R) \neq \mathbf{x}[L, R]] > \varepsilon(\lambda)$ and $R - L + 1 \geq \kappa$, then this experiment outputs 1. Otherwise, output 0.

4.1 A consecutive interval querying, Ideal raLDC

To construct a t -consecutive interval querying, ideal raLDC, we employ the Hamming paLDC to raLDC compiler of Blocki and Zhang [7], which was a modification of the original compiler by Ameri et al. [1]. The construction makes use of two other building blocks: the first is cryptographic puzzles, consisting of two algorithms **PuzzGen** and **PuzzSolve**. On input seed $\mathbf{c}$, **PuzzGen** outputs a puzzle $\mathbf{Z}$ with solution is $\mathbf{c}$ i.e. $\text{PuzzSolve}(\mathbf{Z}) = \mathbf{c}$. The security requirement states that adversary $\mathcal{A}$ in a defined algorithm class $\mathbb{R}$, cannot solve the puzzle $\mathbf{Z}$ and cannot even distinguish between tuples $(\mathbf{Z}_0, \mathbf{c}_0, \mathbf{c}_1)$ and $(\mathbf{Z}_1, \mathbf{c}_0, \mathbf{c}_1)$ for random $\mathbf{c}_i$ and $\mathbf{Z}_i = \text{PuzzGen}(\mathbf{c}_i)$. Ameri et al. are able to construct *memory-hard* puzzles, i.e. cryptographic puzzles unsolvable by algorithm class $\mathbb{R}_{\text{cmc}}$ of algorithms with bounded cumulative memory complexity, under standard cryptographic assumptions. We generalize their definition for any class of algorithms $\mathbb{R}$, where if a cryptographic puzzle is unsolvable by any algorithm in $\mathbb{R}$, we say the puzzle is $\mathbb{R}$ -hard.

The second building block is a variant of a locally decodable code, referred to as LDC^* , which recovers the entire (short) encoded message $\mathbf{s}$ with locality only scaling linearly with the message length. The original LDC^* construction given by Blocki et al. [6] is a repetition code of a constant rate, constant error tolerance code (e.g. the Justesen code), where Dec^* queries a constant number of repeated codewords and performs a majority vote decoding. On message length k , their LDC^* decoder makes $\theta(k)$ queries, where the codeword length $n \gg k$ can be arbitrarily large.

Given a paLDC code $\mathcal{C}_p = (\text{Gen}_p, \text{Enc}_p, \text{Dec}_p)$ and a LDC^* code $\mathcal{C}_* = (\text{Enc}_*, \text{Dec}_*)$, the constructed Hamming raLDC encoder **Enc**, on message $\mathbf{x}$ will (1) generate the secret key $\text{sk} \leftarrow \text{Gen}_p(1^\lambda; \mathbf{c})$ with some random coins $\mathbf{c}$ (2) Compute the paLDC encoding of the message $\mathbf{y}_p \leftarrow \text{Enc}_{p, \text{sk}}(\mathbf{x})$ (3) Compute the LDC^* encoding of the puzzle $\mathbf{y}_* \leftarrow \text{Enc}_*(\mathbf{Z})$, where the LDC^* encoding has codeword length $\theta(|\mathbf{y}_p|)$ (4) and finally, output $\mathbf{y} = \mathbf{y}_p \circ \mathbf{y}_*$. Intuitively, a resource-bounded channel will not be able to solve the puzzle to retrieve the secret key from $\mathbf{y}_*$. Hence, the message encoding $\mathbf{y}_p$ looks random to the channel, reducing the resource-bounded channel's view of the codeword to the view a computationally-bound channel in the private-key setting. On the other hand, the raLDC decoder **Dec** will have sufficient resources to solve the puzzle and locally decode the codeword on input interval $[L, R]$ i.e. **Dec** computes $\tilde{\mathbf{c}} \leftarrow \text{PuzzSolve}(\text{Dec}_*(\tilde{\mathbf{y}}_*))$, $\tilde{\text{sk}} \leftarrow \text{Gen}_p(1^\lambda; \tilde{\mathbf{c}})$, and outputs $\text{Dec}_{\text{psk}}^{\mathbf{y}_p}(L, R)$. By choosing an ideal paLDC code $\mathcal{C}_p$ and an appropriate LDC^* code $\mathcal{C}_*$, the resulting compiled code (Enc, Dec) is an ideal raLDC.

We observe that the compiled raLDC decoder **Dec** satisfies t -consecutive interval query when instantiated with a t -consecutive interval querying paLDC and the original LDC^* construction by Blocki et al. [6]. First, observe that the decoder **Dec** queries the message encoding $\mathbf{y}_p$ and the secret-key randomness encoding $\mathbf{y}_*$ disjointly using Dec_p and Dec_* respectively. Second, Dec_* is also t -consecutive interval querying as long as the encoding used in the repetition encoding has codeword length t . Thus, the overall raLDC decoder **Dec** is t -consecutive interval querying. We summarize this result formally below.

► **Theorem 21.** Let $\lambda \in \mathbb{N}$. Let $\mathcal{C}_p = (\text{Gen}_p, \text{Enc}_p, \text{Dec}_p)$ be a (n_p, k_p) -code that is a t -consecutive interval querying Hamming $(\alpha_p, \kappa_p, \delta_p, \varepsilon_p)$ -paLDC. For any algorithm class $\mathbb{R}$ such that there exists $\mathbb{R}$ -hard cryptographic puzzles, there exists a (n_r, k_r) -code $\mathcal{C}_r = (\text{Enc}_r, \text{Dec}_r)$ that is a t -consecutive interval querying Hamming $(\alpha_r, \kappa_r, \delta_r, \varepsilon_r, \mathbb{R})$ -raLDC for $n_r = \theta(n_p)$, $k_r = k_p = \text{poly}(\lambda)$, $\alpha_r = \theta(\alpha_p)$, $\kappa_r = \kappa_p$, $\delta_r = \theta(\delta_p)$, and $\varepsilon_r = \varepsilon_p + \text{negl}(n)$.

4.2 The Ideal Insdel raLDC Construction

We present our final construction of an ideal Insdel raLDC. We will proceed similarly to our construction of an ideal Insdel paLDC in Section 3, where we use our Hamming-to-Insdel compiler EncCompile with the ideal, t -consecutive interval querying Hamming raLDC constructed in the prior subsection.

► **Construction 22.** Suppose (m, k) -code $\mathcal{C}_r = (\text{Enc}_r, \text{Dec}_r)$ is a t -consecutive interval querying Hamming $(\alpha_r, \kappa_r, \delta_r, \varepsilon_r, \mathbb{R})$ -raLDC. Then define $\mathcal{C} = (\text{Enc}, \text{Dec})$ as $\text{Enc}(x) := \text{EncCompile}(\text{Enc}_r(x))$ and $\text{Dec}^Y(L, R) := \text{Dec}_r^Y(L, R)$.

Just as before, we will need to show that security holds when we use the Hamming-to-Insdel compiler in a resource-bounded setting by giving a reduction argument. One additional nuance we will need to handle is to allow any adversary for the Hamming code sufficient resources to perform the reduction. Let $T(n)$ (resp. $S(n)$) be the running time (resp. space usage) of the computation of $\text{EncCompile}(y)$ and $\tilde{y}_{\text{sim}}$. Let $\mathcal{B} = \text{reduce}_n(\mathcal{A})$ be a reduction from algorithm $\mathcal{A}$ to $\mathcal{B}$ in time $T(n)$ with space usage $S(n)$. Then, for any class of algorithms $\mathbb{R}(n)$, denote its closure class $\overline{\mathbb{R}}(n)$ with respect to reduce_n defined as the minimum class of algorithms such that $\text{reduce}_n(\mathcal{A}) \in \overline{\mathbb{R}}(n)$ for all $\mathcal{A} \in \mathbb{R}(n)$. See the full version [8] for the detailed proof.

► **Theorem 23.** Let $\mathcal{C}_r = (\text{Enc}_r, \text{Dec}_r)$ be a (m, k) -code that is a t -consecutive interval querying Hamming $(\alpha_r, \kappa_r, \delta_r, \varepsilon_r, \mathbb{R}(n))$ -raLDC. Then, $\mathcal{C} = (\text{Enc}, \text{Dec})$ in Construction 22 is a (n, k) -code that is an Insdel $(\alpha, \kappa, \delta, \varepsilon, \overline{\mathbb{R}}(n))$ -raLDC with $\alpha = O\left(\frac{\alpha_r \tau \log^3 n}{t}\right)$, $\kappa = \kappa_r$, $\delta = \Omega(\delta_r)$, and $\varepsilon = \varepsilon_r + \text{negl}(n)$.

► **Corollary 24.** Let $t = \theta(\tau \log^3 n)$, $\tau = \theta(\log n)$, and define any algorithm class $\mathbb{R}$ such that there exists $\mathbb{R}$ -hard cryptographic puzzles. Then by Theorem 21, code $\mathcal{C}$ in Construction 22 is an Insdel $(O(1), O(\log^5 n), O(1), \text{negl}(n), \mathbb{R})$ -raLDC with constant rate.

References

- 1 Mohammad Hassan Ameri, Alexander R. Block, and Jeremiah Blocki. Memory-hard puzzles in the standard model with applications to memory-hard functions and resource-bounded locally decodable codes. In Clemente Galdi and Stanislaw Jarecki, editors, *SCN 22: 13th International Conference on Security in Communication Networks*, volume 13409 of *Lecture Notes in Computer Science*, pages 45–68, Amalfi, Italy, september 12–14 2022. Springer, Cham, Switzerland. doi:10.1007/978-3-031-14791-3_3.
- 2 Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. In László Babai, editor, *36th Annual ACM Symposium on Theory of Computing*, pages 1–10, Chicago, IL, USA, June 13–16 2004. ACM Press. doi:10.1145/1007352.1007361.
- 3 Alexander R. Block and Jeremiah Blocki. Private and resource-bounded locally decodable codes for insertions and deletions. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 1841–1846, 2021. doi:10.1109/ISIT45174.2021.9518249.

- 4 Alexander R Block, Jeremiah Blocki, Elena Grigorescu, Shubhang Kulkarni, and Minshen Zhu. Locally decodable/correctable codes for insertions and deletions. *arXiv preprint arXiv:2010.11989*, 2020. [arXiv:2010.11989](#).
- 5 Jeremiah Blocki, Kuan Cheng, Elena Grigorescu, Xin Li, Yu Zheng, and Minshen Zhu. Exponential lower bounds for locally decodable and correctable codes for insertions and deletions. In *62nd Annual Symposium on Foundations of Computer Science*, pages 739–750, Denver, CO, USA, february 7–10 2022. IEEE Computer Society Press. [doi:10.1109/FOCS52979.2021.00077](#).
- 6 Jeremiah Blocki, Shubhang Kulkarni, and Samson Zhou. On locally decodable codes in resource bounded channels. In Yael Tauman Kalai, Adam D. Smith, and Daniel Wichs, editors, *ITC 2020: 1st Conference on Information-Theoretic Cryptography*, volume 163 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:23, Boston, MA, USA, june 17–19 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. [doi:10.4230/LIPIcs.ITC.2020.16](#).
- 7 Jeremiah Blocki and Justin Zhang. Amortized locally decodable codes, 2025. [doi:10.48550/arXiv.2502.10538](#).
- 8 Jeremiah Blocki and Justin Zhang. Amortized locally decodable codes for insertions and deletions. *arXiv preprint*, 2025. [doi:10.48550/arXiv.2507.03141](#).
- 9 Alessandro Chiesa, Tom Gur, and Igor Shinkar. Relaxed locally correctable codes with nearly-linear block length and constant query complexity. In Shuchi Chawla, editor, *31st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1395–1411, Salt Lake City, UT, USA, january 5–8 2020. ACM-SIAM. [doi:10.1137/1.9781611975994.84](#).
- 10 Gil Cohen and Tal Yankovitz. Asymptotically-good rlccs with $(\log n)^{2+o(1)}$ queries. In *39th Computational Complexity Conference (CCC 2024)*, pages 8:1–8:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. [doi:10.4230/LIPIcs.CCC.2024.8](#).
- 11 Klim Efremenko. 3-query locally decodable codes of subexponential length. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 39–44, Bethesda, MD, USA, May 31 – june 2 2009. ACM Press. [doi:10.1145/1536414.1536422](#).
- 12 Meghal Gupta. Constant query local decoding against deletions is impossible. In Bojan Mohar, Igor Shinkar, and Ryan O’Donnell, editors, *56th Annual ACM Symposium on Theory of Computing*, pages 752–763, Vancouver, BC, Canada, june 24–28 2024. ACM Press. [doi:10.1145/3618260.3649655](#).
- 13 Tom Gur, Govind Ramnarayan, and Ron D. Rothblum. Relaxed locally correctable codes. In Anna R. Karlin, editor, *ITCS 2018: 9th Innovations in Theoretical Computer Science Conference*, volume 94, pages 27:1–27:11, Cambridge, MA, USA, january 11–14 2018. Leibniz International Proceedings in Informatics (LIPIcs). [doi:10.4230/LIPIcs.ITCS.2018.27](#).
- 14 Brett Hemenway and Rafail Ostrovsky. Public-key locally-decodable codes. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 126–143, Santa Barbara, CA, USA, august 17–21 2008. Springer Berlin Heidelberg, Germany. [doi:10.1007/978-3-540-85174-5_8](#).
- 15 Brett Hemenway, Rafail Ostrovsky, Martin J Strauss, and Mary Wootters. Public key locally decodable codes with short keys. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 605–615. Springer, 2011. [doi:10.1007/978-3-642-22935-0_51](#).
- 16 Don Hush and Clint Scovel. Concentration of the hypergeometric distribution. *Statistics & Probability Letters*, 75(2):127–132, November 2005. [doi:10.1016/j.spl.2005.05.019](#).
- 17 Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *32nd Annual ACM Symposium on Theory of Computing*, pages 80–86, Portland, OR, USA, May 21–23 2000. ACM Press. [doi:10.1145/335305.335315](#).
- 18 Vinayak M. Kumar and Geoffrey Mon. Relaxed local correctability from local testing. In Bojan Mohar, Igor Shinkar, and Ryan O’Donnell, editors, *56th Annual ACM Symposium on Theory of Computing*, pages 1585–1593, Vancouver, BC, Canada, june 24–28 2024. ACM Press. [doi:10.1145/3618260.3649611](#).

- 19 Richard J. Lipton. A new approach to information theory. In *Proceedings of the 11th Annual Symposium on Theoretical Aspects of Computer Science*, STACS '94, pages 699–708, Berlin, Heidelberg, 1994. Springer-Verlag. doi:10.1007/3-540-57785-8_183.
- 20 Rafail Ostrovsky, Omkant Pandey, and Amit Sahai. Private locally decodable codes. Cryptology ePrint Archive, Report 2007/025, 2007. URL: <https://eprint.iacr.org/2007/025>.
- 21 Rafail Ostrovsky, Omkant Pandey, and Amit Sahai. Private locally decodable codes. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *ICALP 2007: 34th International Colloquium on Automata, Languages and Programming*, volume 4596 of *Lecture Notes in Computer Science*, pages 387–398, Wroclaw, Poland, july 9–13 2007. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-540-73420-8_35.
- 22 Rafail Ostrovsky and Anat Paskin-Cherniavsky. Locally decodable codes for edit distance. In Anja Lehmann and Stefan Wolf, editors, *ICITS 15: 8th International Conference on Information Theoretic Security*, volume 9063 of *Lecture Notes in Computer Science*, pages 236–249, Lugano, Switzerland, May 2–5 2015. Springer, Cham, Switzerland. doi:10.1007/978-3-319-17470-9_14.
- 23 L.J. Schulman and D. Zuckerman. Asymptotically good codes correcting insertions, deletions, and transpositions. *IEEE Transactions on Information Theory*, 45(7):2552–2557, 1999. doi:10.1109/18.796406.
- 24 Sergey Yekhanin et al. Locally decodable codes. *Foundations and Trends® in Theoretical Computer Science*, 6(3):139–255, 2012. doi:10.1561/04000000030.

Powerful Primitives in the Bounded Quantum Storage Model

Mohammed Barhoush ✉

Université de Montréal (DIRO), Canada

Louis Salvail ✉

Université de Montréal (DIRO), Canada

Abstract

The bounded quantum storage model aims to achieve security against computationally unbounded adversaries that are restricted only with respect to their quantum memories. In this work, we provide the following contributions in this model:

1. We build one-time programs and utilize them to construct CCA1-secure symmetric key encryption and message authentication codes. These schemes require no quantum memory from honest users, yet they provide information-theoretic security against adversaries with arbitrarily large quantum memories, as long as the transmission length is suitably large.
2. We introduce the notion of k -time *program broadcast* which is a form of program encryption that allows multiple users to each learn a single evaluation of the encrypted program, while preventing any one user from learning more than k evaluations of the program. We build this primitive unconditionally and employ it to construct CCA1-secure asymmetric key encryption, encryption tokens, signatures, and signature tokens. All these schemes are information-theoretically secure against adversaries with roughly $e^{\sqrt{m}}$ quantum memory where m is the quantum memory required for the honest user.

All of the constructions additionally satisfy disappearing security, essentially preventing an adversary from storing and using a transmission later on.

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques

Keywords and phrases Quantum Cryptography, Bounded Quantum Storage Model, Information-Theoretic Security

Digital Object Identifier 10.4230/LIPIcs.ITC.2025.2

Related Version *Full Version*: <https://arxiv.org/abs/2302.05724>

1 Introduction

Most of the interesting cryptographic concepts of security are unattainable when dealing with completely unrestricted adversaries. The conventional approach to resolve this conundrum is to focus solely on adversaries operating within polynomial time. Nevertheless, given our current understanding of complexity theory, security in this setting can only be guaranteed under the assumed hardness of solving certain problems, such as factoring, computing the discrete log, or learning with errors. As a consequence, security in the *computational model* is usually conditional.

An alternative approach is to constrain the quantum memory (qmemory) available to the adversary. This model is called the *Bounded Quantum Storage Model* (BQSM) and was first introduced in 2005 by Damgård, Fehr, Salvail, and Schaffner [9, 30]. They demonstrated that non-interactive oblivious transfer and bit-commitment can be achieved information-theoretically in this model! Surprisingly, these schemes demand no qmemory from honest participants and can be made secure against adversaries with arbitrarily large qmemories by



© Mohammed Barhoush and Louis Salvail;
licensed under Creative Commons License CC-BY 4.0
6th Conference on Information-Theoretic Cryptography (ITC 2025).
Editor: Niv Gilboa; Article No. 2; pp. 2:1–2:20



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

sufficiently extending the transmission length. Subsequently, this oblivious transfer scheme was later adapted to the more useful variant of non-interactive 1-2 oblivious transfer¹ [10]. Henceforth, we denote this 1-2 oblivious transfer scheme as *BQS-OT*.

Concretely, in the BQSM, an adversary $\mathcal{A}_s$ has access to unlimited resources at all times except at certain points. At these points, we say *the memory bound applies*, and the adversary is forced to reduce its stored state to s -qubits. We emphasize that the adversary is again unrestricted with respect to its qmemory after the bound applies and is never restricted with respect to its computational power or classical memory.

In practice, the point when the memory bound applies can be implemented by pausing and delaying further transmissions which enforces the bound given the technological difficulties of maintaining a quantum state. Even in the foreseeable future, qmemory is not only expected to be very expensive but also very limited in size to allow for good enough fidelity. Consequently, this model appears to provide an apt characterization of real-world adversaries.

1.1 Our Results

In this paper, we delve deeper into the BQSM. We first adapt conventional definitions, such as those pertaining to encryption and one-time programs, to the BQSM. We then proceed to construct a variety of cryptographic primitives as elaborated on below.

We first build information-theoretic secure one-time programs² through utilizing the BQS-OT scheme. We then leverage one-time programs to construct information-theoretic secure CCA1-symmetric encryption. All these schemes are secure against any computationally unbounded adversary with s qmemory where s can be any fixed polynomial (in the security parameter) whereas honest users do not need any qmemory.

Next, we address the challenges associated with information-theoretic *asymmetric* key cryptography, particularly the task of hiding the secret key from the public key without relying on computational assumptions. We propose the novel primitive termed program broadcast as a natural solution.

Roughly speaking, a k -time program broadcast of a function P allows arbitrarily many users to each (1) obtain a single evaluation of the program, and, at the same time, (2) cannot be used by any user to learn more than k evaluations of the program. We proceed to construct an information-theoretically secure program broadcast and employ it to build CCA1-secure asymmetric key encryption. In the full paper, we also use program broadcast to build information-theoretically secure signatures, signature tokens, and encryption tokens. A signature token, as first defined in [5], can be used to sign a single message (without the signing key) and then self-destructs. We similarly define decryption tokens – each such token allows its holder to decrypt a single ciphertext and then self-destructs.

The schemes built from program broadcast are secure against any computationally unbounded adversary with s qmemory where s can be any fixed polynomial in the security parameter but require $\lg^k(\lambda)$ qmemory for the receiver where $k \in \mathbb{R}$ can take any value larger than 2. This implies that the gap between the qmemory required of the honest user and the needed qmemory to break security approaches $\lg^2(\lambda)$ vs. $\text{poly}(\lambda)$ which translates to a gap of m vs. $e^{\sqrt{m}}$ by setting $m = \lg^2(\lambda)$. While we cannot assert that the required qmemory for

¹ 1-2 *oblivious transfer* is a primitive allowing a sender to transmit two strings so that (1) a receiver can choose to receive anyone of the two strings without learning anything about the other one and (2) the sender is oblivious of the receiver's choice.

² A one-time program (as introduced in [15]) (1) can be used to obtain a single evaluation of the program on any input chosen by the user, and at the same time, (2) (black-box obfuscation) cannot be used to obtain any other information about the program (see Definition 11).

the honest receiver in the asymmetric setting is optimal, we show in the full paper that it is not possible to achieve asymmetric key cryptography without requiring any qmemory for the honest user. Given that our qmemory requirements are already quite minimal, there is little room for improvement. For simplicity, in the rest of the paper we work with $k = 3$, however, our results easily apply to any value $k > 2$.

Prior to this work, none of the aforementioned primitives have been achieved in the BQSM and all are impossible information-theoretically in the plain model.

Our constructions additionally satisfy disappearing security which shall be formally defined in Sec. 5 but we provide a brief overview here. A transmission, say a ciphertext or signature, is deemed disappearing if it can no longer be used after a certain point [21]. In the encryption setting, an adversary cannot decrypt a ciphertext even if the private key is revealed afterward. While in the authentication setting, an adversary cannot forge a signature of a message μ even if it has received a signature of μ earlier! Such disappearing properties are impossible in the plain model for obvious reasons, but in the BQSM an adversary cannot necessarily store a ciphertext or signature for later use.

1.2 Notes on Feasibility

The qmemory requirements for the receiver in our program broadcast, asymmetric key encryption, and signature schemes make these constructions difficult to actualize with current technology given the difficulty of storing quantum states. It is worth noting, however, that the users only need a small amount of qmemory and only need it at the start in order to process the public keys. Therefore, we hope that these schemes will become feasible with further advancements.

Meanwhile, our one-time programs, symmetric key encryption, and message authentication codes can be realized with current technology as they only require the ability to prepare, send, and measure a single qubit in the BB84 basis. Specifically, these constructions can be implemented on hardware designed to run quantum key distribution (QKD), albeit with a caveat. Indeed, popular *weak coherent pulse sources* require interaction as the receiver needs to tell the sender which pulses were received. However, other technologies can enable non-interactive quantum transmissions to allow for non-interactive primitives such as one-time programs, encryption, and signature schemes. For instance, *heralded photon sources* can, in principle, remove the need for interaction.

Note also that our constructions can be modified to allow for users with imperfect apparatus to perform non-interactive error correction without compromising the security of the scheme as we briefly discuss in Sec. 9. With all that being said, in this work, we focus on building the theory and leave it as an open research direction to properly prepare the theoretical constructions for real-world applications.

1.3 Related Work

We compare our results with similar contributions in alternative models.

1.3.1 Plain Model

One-time programs in the plain model are impossible for the simple reason that any software can be stored and then reused to obtain multiple evaluations. Broadbent, Gutoski, and Stebila [7] extended this idea to the quantum realm and showed that quantum one-time programs are also impossible. These results do not apply to the BQSM as one-time programs are indeed feasible.

That being said, the works [15, 18] constructed one-time programs by leveraging one-time memory devices. These hardware devices essentially implement 1-2 oblivious transfer – each device holds two secrets and allows the device holder to choose and learn one and only one secret. Essentially, a circuit is encrypted and sent along with appropriate one-time memory devices. A receiver can only obtain enough keys from the devices to learn one evaluation from the encrypted circuit. Our one-time program construction involves replacing the one-time memory devices with the BQS-OT scheme, which serves the same purpose, although our proof requires novel analysis.

1.3.2 Bounded Classical Storage Model (BCSM)

Memory limitations were first considered in the classical setting. Specifically, in 1992, Maurer [26] introduced the *bounded classical storage model* (BCSM) where adversaries are restricted only with respect to the size of their classical memory. A cipher was shown to guarantee privacy unconditionally, even against adversaries having access to a larger memory than what is needed for honest players. Subsequent works [20, 11, 28, 12] have achieved information-theoretic (interactive) oblivious transfer, key agreement, and symmetric key encryption and signatures in this model. Furthermore, Guan, Wichs, and Zhandry [19] recently provided constructions for disappearing ciphertext and signature schemes based on public-key cryptography and one-way functions, respectively.

Overall, the BQSM seems to provide stronger security guarantees when compared to the BCSM for several reasons. First of all, in the BCSM, the gap between the required memory to run many of these schemes securely and the size of the memory needed to break for various primitives, including asymmetric encryption, is typically on the order of m vs. m^2 , which is optimal [12]. This memory gap is a notable vulnerability as if it is feasible for honest users to store m bits, it does not seem too difficult for a determined adversary to store m^2 bits. In contrast, the memory gap of m vs. $e^{\sqrt{m}}$ for primitives in the BQSM is far more significant. Moreover, certain primitives achievable in the BQSM are completely unattainable in the BCSM. For instance, non-interactive oblivious transfer was shown to be impossible in the BCSM [11] which implies the impossibility of the stronger one-time program primitive.

1.3.3 Noisy Quantum Storage Model

In the *Noisy Quantum Storage Model*, which was first introduced in [31], the qmemory of the adversary is not limited in size but is intrinsically noisy. The BQS-OT schemes proposed in [9, 10] can, essentially as such, be shown secure against adversaries with noisy qmemories. In this work, we only study the BQSM and an interesting research direction is to generalize our results to the Noisy Quantum Storage Model.

2 Technical Overview

We now discuss each of our results in more detail. We first describe how to construct one-time programs and then, how to use them to build symmetric key encryption. Next, we explain why we need a new tool to tackle asymmetric cryptography. Correspondingly, we introduce the notion of program broadcast and describe how to build it in the BQSM. We then show how program broadcast can be used to build asymmetric encryption. Finally, we discuss two impossibility results related to the qmemory requirements and disappearing properties of the schemes.

2.1 One-Time Programs

We now present a quantum algorithm $\mathcal{O}$ that converts any polynomial classical circuit to a one-time program in the BQSM (Theorem 17). Fortunately, the BQSM dodges all the impossibility proofs of one-time programs. The work showing the impossibility of quantum one-time programs [7] does not apply to our setting since our programs cannot be stored by the adversary to be reused. Moreover, the proof of the impossibility of quantum obfuscation [7] does not apply since it requires the adversary to run the quantum circuit homomorphically. Adversaries in our setting are forced to continuously perform measurements on quantum states (due to the memory bound) which interrupts a quantum homomorphic evaluation.

Before describing the construction, we discuss its intuition. A natural first attempt is to apply previous one-time program constructions [15, 18] based on one-time memory devices but replace the devices with BQS-OT transmissions. The problem is that simulating an adversary requires determining which 1-out-of-2 strings are received from the oblivious transfer transmissions. This information allows the simulator to deduce which evaluation the adversary has learned from the program. In earlier constructions, this is not a problem since the adversary must *explicitly* ask which string it wishes to receive from the one-time memory devices. Note that Kilian [23] builds a similar notion known as *oblivious circuit evaluation* from OT but the simulator is also assumed to be given this information. In these works, the simulator can simply run the adversary and “know” which strings are requested. However, in our case, it is not clear this extraction is always possible – a complex adversary might not know itself which 1-out-of-2 strings it has received! To make matters worse, our adversary is a quantum and computationally unbounded algorithm. Note that the proof of security for BQS-OT does not provide a method to extract the receiver’s chosen bit but rather just shows that such a bit must exist.

To solve this issue, we construct a simulator that (inefficiently) analyzes the circuit of the adversary gate-by-gate to extract the chosen bit. In particular, the simulator takes the circuit of the adversary and runs it an exponential number of times on all possible BQS-OT transmissions. This allows the simulator to approximately determine the distribution of the measurement results obtained by the adversary given the transmission provided to the adversary. This then allows the simulator to determine the distribution of the BQS-OT transmission from the perspective of the adversary given the measurement results obtained. In other words, the simulator can infer which part of the transmission the adversary is uncertain about which can be used to determine which string the adversary is oblivious to.

The rest of the construction involves adapting Kilian’s approach [23] for building one-time programs from oblivious transfer to the quantum realm. Readers are referred to Sec. 6 for the formal statement of our result. However, the construction and proof are given in the full paper as they are quite long and detailed.

2.2 Symmetric Key Encryption

We use one-time programs to build a symmetric key encryption scheme that satisfies indistinguishability under (lunchtime) chosen ciphertext attack (IND-CCA1) [8] and with disappearing ciphertexts. IND-CCA1 tests security against an adversary that can query the encryption oracle at any point in the experiment and can query the decryption oracle at any point before the challenge ciphertext is received. To satisfy *disappearing* security under CCA1, we require that such an adversary cannot win the IND-CCA1 experiment even if the private key is revealed after the challenge ciphertext is given.

To achieve IND-CPA (where the adversary is not given access to a decryption oracle), the ciphertext can simply consist of a one-time program that evaluates to $\perp$ on every input except on the private key, where it outputs the message. By the security of one-time programs, an adversary has a negligible chance of guessing the key and learning the message. Upgrading to CCA1-security turns out to be far more involved as we now discuss.

First of all, it is trivial to show that we must rely on quantum ciphertexts for unconditional security in the BQSM. Now, traditionally, a CPA-secure scheme can be upgraded to a CCA-secure one by simply authenticating ciphertexts using message-authentication codes or a signature scheme. However, it seems difficult to authenticate a quantum ciphertext in a way that allows verification without any qmemory. Alternatively, we could authenticate the output of the one-time programs in the CPA construction. Specifically, instead of sending programs that map the secret key to the message, we send programs that map the key to the message along with a tag or signature of the message.

This solution fails due to the following attack: an adversary queries the encryption oracle on an arbitrary message μ , receives a one-time program, modifies the one-time program so that it outputs $\perp$ on any input starting with 0, and forwards the modified one-time program to the decryption oracle. Depending on the response received from the oracle, the adversary can successfully determine the first bit of the secret key. This attack can be repeated a polynomial number of times to deduce the entire secret key. It turns out an adversary can perform a variety of attacks of this sort due to a fundamental problem with the BQSM: this model provides security by ensuring that users only receive partial information from a transmission. Hence, it is difficult to detect whether an adversary has tampered with a transmission.

To thwart such attacks, we rely on a construction where a ciphertext consists of two interdependent one-time programs. The first program initiates some randomness and this randomness is used to ensure that the second one-time program is evaluated on a seemingly random input during decryption. To prevent the adversary from choosing the first one-time program, we authenticate the output of the first program in the second program using the secret key. We show that it is difficult to modify both programs simultaneously in a meaningful way without being detected. Proving this rigorously is somewhat technical as the adversary can still perform various modifications successfully – the reader is referred to Theorem 22 for the formal proof.

2.3 The Obstacle to Asymmetric Cryptography

Asymmetric cryptography utilizes a pair of related keys, a secret and a public key, to enable versatile cryptographic applications. Specifically, our goal is to build information-theoretic secure asymmetric key encryption and digital signatures in the BQSM. However, the task of concealing the secret key from the public key without relying on computational assumptions poses a significant challenge. Such assumptions should be avoided in the BQSM given the goal of this model is to base security purely on the memory limitations of the players.

This implies we need to impose the memory bound in some way during the public key transmission in order to hide the secret key. In the classical bounded storage model, this can be done by announcing a large classical public key [12]. However, in our scenario, imposing the memory bound necessitates the use of quantum public keys. Unfortunately, due to no-cloning, we need to create and distribute multiple copies of the quantum key – this is the general approach taken in the computational setting as well [17, 13, 22]. Here we face a critical problem: if a computationally unbounded adversary gains access to multiple copies

of the key, it can repeatedly reuse its quantum memory to process multiple keys. Such an adversary could gradually extract classical information from each key copy until it has learned the entire quantum public key.

To prevent this attack, it is imperative that every public key copy needs *additional* qmemory to process, preventing an adversary from processing an unlimited number of keys. In other words, we need to distribute keys in a way so that all users must process the keys simultaneously or in parallel. More abstractly, we want to distribute quantum information in a way that allows every user to learn some information while preventing a computationally unbounded adversary from learning too much information. This goal is a recurring theme in various settings, hence, we introduce a new notion which we term *program broadcast* that formalizes and captures this objective. We build this primitive unconditionally in the BQSM and employ it to build asymmetric cryptography.

All in all, this is the first work to achieve information-theoretic asymmetric key encryption in any of the bounded (classical or quantum) storage models. This is the main focus and contribution of this paper. In the following, we first discuss program broadcast, which may be of independent interest, and then how this can be used to realize asymmetric cryptography.

2.4 Program Broadcast

In many situations, we would like to send multiple one-time programs to multiple users while ensuring that no adversary can take advantage of all the information to learn the program. This is the goal of program broadcast. Recall that a k -time program broadcast of a function P allows an unbounded polynomial number of users to each (1) obtain at least a single evaluation of the program, and, at the same time, (2) cannot be used by any user to learn more than k evaluations of the program. Essentially, an adversary with access to the broadcast can be simulated with access to k queries to an oracle for P .

In Sec. 7, we present a scheme for an information-theoretically secure program broadcast in the BQSM. The idea is to distribute a polynomial number of augmented one-time programs of P in a fixed time period. Unlike the one-time programs discussed in the previous section, these augmented versions require a small amount of qmemory to process.

More rigorously, there is a set time where the sender distributes one-time programs of the form $\mathcal{O}(P \oplus c_i)$ where $\mathcal{O}$ is our one-time compiler. In each one-time program, the output of P is padded with a different value c_i . A quantum ciphertext is generated which encrypts the value c_i and is sent with the corresponding one-time program. Users can evaluate the one-time program but need to store the ciphertext states until the encryption key is revealed to make use of their evaluation. The key is revealed at a set time after all users have received a one-time program copy. By revealing this classical key at the end, we are essentially forcing users to process the programs in a parallel fashion, limiting the number of evaluations that can be learned. In other words, an adversary with bounded qmemory can store the ciphertext states for a limited number of one-time programs and thus can only learn a limited number of evaluations of P .

To prove this rigorously, we need a new min-entropy lower bound (Lemma 7) which roughly states that if $H_\infty(X_0 X_1 \dots X_{p-1} | Q) \geq \alpha$, where $X_i \in \{0, 1\}^n$ and Q is some quantum information, then there exists $i \in [p]$ and negligible ϵ such that roughly $H_\infty^\epsilon(X_i) \geq \alpha/p$. König and Renner [24] established such a bound but only in the case when n is sufficiently large with respect to p . Unfortunately, in our applications, n is very small with respect to p . Informally, we resolve this issue by using their bound when p is small and then use this as a base case to an inductive argument for larger values of p achieving the required result. This result is of independent interest in the field of information theory.

Note that while one-time programs can be built from one-time memory devices in the plain model, this is insufficient to build program broadcast in the plain model. Specifically, we need to use the qmemory bound outside the one-time programs as well in order to construct this primitive. Hence, program broadcast acts as a more pronounced advantage of the BQSM.

2.5 Asymmetric Key Encryption

We now discuss how we upgrade our symmetric key encryption scheme to a CCA1-secure asymmetric key scheme using program broadcast. We let the private key be a large program P while the public key is a quantum program broadcast of P . A user can use the broadcast to learn a single evaluation of P which can be used to encrypt messages in the same way as in the private key setting. However, an adversary can only learn a limited number of evaluations by the security of the program broadcast. If P is large enough then this knowledge is not sufficient to predict the output of P on a random input. In other words, the adversary cannot predict the encryption key of another user, so security is reduced to the symmetric key setting.

It is not difficult to show that the public keys need to be mixed-state for information-theoretic security in the BQSM. This might seem unfortunate, but we believe that it is not very relevant whether keys are pure or mixed-state. Some works, such as [4], require that the quantum public keys be pure since this would provide some level of certification by allowing users to compare keys with a SWAP test. However, this test is not always feasible to perform in the BQSM given that keys cannot necessarily be stored. Instead, we provide a more secure way to certify mixed-state quantum keys without establishing authenticated quantum channels in a companion work [2]. That being said, in this work, we do not certify public keys and it is always assumed that honest users receive authentic quantum public keys. This is a common assumption for the quantum public key schemes, such as those in [17, 13, 22], as a quantum state cannot be signed [3].

2.6 Impossibility Results

We briefly discuss two interesting impossibility results that we present in the full paper. First, we show that it is impossible to implement asymmetric schemes (with unbounded public key copies) in the BQSM without requiring qmemory from the honest user. The fundamental idea is that if the public keys require no qmemory to process then an adversary can request and process an unbounded number of key copies which would reveal information about the secret key. The technical argument is more involved since the public keys may be mixed-state which could aid in hiding information.

The second impossibility result is concerned with the disappearing property of our constructions. The ciphertexts and one-time programs satisfy disappearing security providing interesting applications as we discussed earlier. However, this disappearing property can also be disadvantageous in some scenarios. For instance, sometimes, it is more beneficial to have an obfuscation that can be stored and reused multiple times instead of a disappearing one-time program. The final contribution of this paper is to provide a negative result showing that non-disappearing obfuscation and one-time programs are impossible in the BQSM.

The proof relies on a new approach since our simulator is computationally unbounded, which nullifies adversarial attacks used in standard impossibility proofs [1, 14, 6, 7]. Essentially, we show that if a computationally unbounded adversary can perform a single evaluation after the qmemory bound applies then it can rewind the program and perform another evaluation. By the gentle measurement lemma [32], this rewinding introduces only a negligible error each time which allows the adversary to learn a super-polynomial number of evaluations

with non-negligible probability. These evaluations can be used to learn more information regarding the hidden program than any simulator can learn using only a polynomial number of oracle queries. We refer the reader to the full paper for more details.

3 Preliminaries

3.1 Notation

In the following, we denote Hilbert spaces by calligraphic letters: $\mathcal{H}$, $\mathcal{V}$, etc... The set of positive semi-definite operators in Hilbert space $\mathcal{H}$ are denoted $\text{Pos}(\mathcal{H})$. When Hilbert space $\mathcal{X}$ holds a classical value, we denote the random variable for this value by X .

We denote the density matrix of a quantum state in a register E as ρ_E . If a quantum state depends on a classical variable X , then we denote ρ_E^x as the density matrix of the state given $X = x$. Meanwhile, to an observer who does not know the value of X , the state is given by $\rho_E := \sum_x P(X = x) \rho_E^x$ and the joint state by $\rho_{XE} := \sum_x P(X = x) |x\rangle\langle x| \otimes \rho_E^x$. Such states having a quantum part depending on a classical value are called *cq-states*. The random variable X then corresponds to the classical state $\rho_X := \sum_x P(X = x) |x\rangle\langle x|$.

We denote the trace distance between two density matrices as $\delta(\rho, \sigma)$ and the trace norm as $\|\rho\|_1$.

Notice that $\rho_{XE} = \rho_X \otimes \rho_E$ if and only if ρ_E is independent of X (i.e. $\rho_E^x = \rho_E$) which means that no information can be learned regarding ρ_E from X and vice versa. More generally, if ρ_{XE} is ϵ -close to $\rho_E \otimes \rho_X$ in terms of trace distance denoted as $\delta(\rho_{XE}, \rho_X \otimes \rho_E) \leq \epsilon$ or equivalently $\rho_{XE} \approx_\epsilon \rho_X \otimes \rho_E$, then no observer can distinguish between the two systems with advantage greater than ϵ . We write $\langle \rho_A, \rho_B \rangle$ to denote a sequential transmission where register A is sent and then register B .

The rectilinear or $+$ basis for the complex space $\mathbb{C}^2$ is the pair $\{|0\rangle, |1\rangle\}$ while the diagonal or $\times$ basis is the pair $\{|0\rangle_\times, |1\rangle_\times\}$ where $|0\rangle_\times := \frac{|0\rangle + |1\rangle}{\sqrt{2}}$ and $|1\rangle_\times := \frac{|0\rangle - |1\rangle}{\sqrt{2}}$. To choose between the $+$ and $\times$ basis depending on a bit $b \in \{0, 1\}$, we write $\{+, \times\}_b$.

We say $x \in_R X$ if x is chosen uniformly at random from the values in X . We let $\mathbb{1}_k$ denote the density matrix of a uniformly distributed variable on k elements. Also, let $[n] := [0, 1, \dots, n-1]$ and let $\text{negl}(n)$ be denoting any function that is smaller than the inverse of any polynomial for large enough n .

We say an algorithm A is *QPT* if it is a quantum polynomial-time algorithm. We let A^P denote an algorithm with access to a polynomial number of classical oracle queries to the program P and let A^{qP} denote access to q classical oracle queries.

Recall a Pauli pad [27] information-theoretically hides a n -qubit state using a $2n$ -bit key k . We denote $\text{PP}_k(\rho)$ to be the Pauli pad of the state with density matrix ρ using key k .

By abuse of notation, if x is a binary string and M is a matrix then, for simplification, we let $M \cdot x$ denote the string representation of the vector $M \cdot \vec{x}$.

3.2 Rényi Entropy

We remind the reader of the notion of quantum Rényi Entropy and its variants. See [25] for more detailed definitions.

► **Definition 1 (Conditional Min-Entropy).** Let $\rho_{XB} \in \text{Pos}(\mathcal{H}_X \otimes \mathcal{H}_B)$ be classical on $\mathcal{H}_X$. The min-entropy of ρ_{XB} given $\mathcal{H}_B$ is defined as

$$H_\infty(X|B)_\rho := -\lg(p_{\text{guess}}(X|B)_\rho)$$

where $p_{\text{guess}}(X|B)_\rho$ is the maximal probability to decode X from B with a POVM on $\mathcal{H}_B$.

► **Definition 2** (Smooth Min-Entropy). *Let $\epsilon \geq 0$ and $\rho_{XB} \in \text{Pos}(\mathcal{H}_X \otimes \mathcal{H}_B)$. The ϵ -smooth min-entropy of ρ_{XB} given $\mathcal{H}_B$ is defined as*

$$H_\infty^\epsilon(X|B)_\rho := \sup_{\bar{\rho}} H_\infty(X|B)_{\bar{\rho}}$$

where the supremum is taken over all density operators $\bar{\rho}_{XB}$ acting on $\mathcal{H}_X \otimes \mathcal{H}_B$ such that $\delta(\bar{\rho}_{XB}, \rho_{XB}) \leq \epsilon$.

3.3 Uncertainty Relations

This section discusses lower bounds on the average min-entropy and presents a new generalized min-entropy splitting lemma conditioned on quantum states. This will be fundamental in our program broadcast construction.

The min-entropy satisfies the following chain rule.

► **Lemma 3** (Chain Rule for Min-Entropy [29]). *Let $\epsilon \geq 0$ and $\rho_{XUE} \in \text{Pos}(\mathcal{H}_X \otimes \mathcal{H}_U \otimes \mathcal{H}_E)$ where register E has size m . Then,*

$$H_\infty^\epsilon(X|UE)_\rho \geq H_\infty^\epsilon(XE|U)_\rho - m.$$

► **Lemma 4** (Uncertainty Relation [9]). *Let $\rho \in \mathcal{P}(\mathcal{H}_2^{\otimes n})$ be an arbitrary n -qubit quantum state. Let $\Theta \in_R \{+, \times\}^n$ and let X be the outcome when ρ is measured in basis Θ . Then for any $0 < \gamma < \frac{1}{2}$,*

$$H_\infty^\epsilon(X|\Theta)_\rho \geq \left(\frac{1}{2} - 2\gamma\right)n,$$

where $\epsilon = \exp(-\frac{\gamma^2 n}{32(2-\lg(\gamma))^2})$.

If $X = X_0 X_1$ has high entropy, then it was shown in [10] that, in a randomized sense, one of X_0 or X_1 has high entropy.

► **Lemma 5** (Conditional Min-Entropy Splitting Lemma [10]). *Let $\epsilon \geq 0$ and let X_0, X_1 and Z be random variables. If $H_\infty^\epsilon(X_0 X_1|Z) \geq \alpha$, then there exists a binary random variable C such that $H_\infty^{\epsilon+\epsilon'}(X_{1-C}|ZC) \geq \alpha/2 - 1 - \log(1/\epsilon')$ for any $\epsilon' > 0$.*

More generally, min-entropy sampling gives lower bounds on the min-entropy of a subset of a string given the min-entropy of the entire string. In [24], it was shown how to sample min-entropy relative to quantum knowledge which allows us to give a lower bound on min-entropy conditioned on a quantum state.

► **Lemma 6** (Quantum Conditional Min-Entropy Splitting Lemma [24]). *Let $\epsilon \geq 0$ and let $\rho_{X_0 X_1 B}$ be a quantum ccq-state where X_0 and X_1 are classical random variables on alphabet $\mathcal{X}$ of dimension $d = \lg|\mathcal{X}| > 14$. Then, there exists a binary random variable C such that for all $\tau \geq 0$,*

$$H_\infty^{\epsilon+\tau}(X_C|CB)_\rho \geq \frac{H_\infty^\tau(X_0 X_1|B)_\rho}{2} - 2$$

where $\epsilon := 2^{-\frac{3d}{2}+1}$.

Proof. Notice that any distribution P_C over $\{0, 1\}$ is a $(2, 3/4, 0)$ -sampler (Definition 2.1 [24]). Corollary 6.19 in [24] gives the result. ◀

In Lemma 6 the size of the sampled string X_C is half the size of the original string X . The results in [24] do not give strong lower bounds when the sampled string is small relative to the size of the original string. Hence, we present the following generalization of Lemma 6 which will be used in constructing program broadcast and asymmetric key encryption.

► **Lemma 7** (Generalized Min-Entropy Splitting Lemma). *Let $\epsilon \geq 0$. Let ρ_{XB} be a quantum cq-state where $X := X_0X_1\dots X_\ell$ and each X_i is a classical random variable over alphabet $\mathcal{X}$ of dimension $d > 14$. There exists a random variable C (with $\lceil \lg \ell \rceil$ bits) such that for all $\tau \geq 0$,*

$$H_\infty^{\epsilon+\tau}(X_C|CB)_\rho \geq \frac{H_\infty^\tau(X|B)_\rho}{\ell} - 4$$

where $\epsilon := 2^{-\frac{3d}{2}+2}$.

Proof. We only consider the case $\ell = 2^k$, however, the same argument works for the general case. We prove the following slightly stronger bound using induction on k :

$$H_\infty^{\epsilon_k+\tau}(X_C|CB) \geq \frac{H_\infty^\tau(X|B)}{2^k} - 4\left(1 - \frac{1}{2^k}\right).$$

where $\epsilon_k := \sum_{i=0}^{k-1} 2^{-2^i d+1}$. The base case $k = 1$ follows trivially from the Quantum Conditional Min-Entropy Splitting Lemma 6. Assume the claim holds for $k = n$. For $k = n + 1$, we apply Lemma 6 on the two variables $Y_0 := X_1\dots X_{2^n}$ and $Y_1 := X_{2^n+1}\dots X_{2^{n+1}}$ to deduce that there exists a binary variable C_1 such that $H_\infty^{\epsilon'+\tau}(Y_{C_1}|BC_1) \geq \frac{H_\infty^\tau(X|B)}{2} - 2$ where $\epsilon' := 2^{-2^n d+1}$. By the inductive hypothesis, there exists C' such that:

$$\begin{aligned} H_\infty^{\epsilon_n+\epsilon'+\tau}(X_{C_1C'}|BC_1C') &\geq \frac{H_\infty^{\epsilon'+\tau}(Y_{C_1}|BC_1)}{2^n} - 4\left(1 - \frac{1}{2^n}\right) \geq \\ \frac{\frac{H_\infty^\tau(X|B)}{2} - 2}{2^n} - 4\left(1 - \frac{1}{2^n}\right) &= \frac{H_\infty^\tau(X|B)}{2^{n+1}} - 4\left(1 - \frac{1}{2^{n+1}}\right). \end{aligned} \quad \blacktriangleleft$$

3.4 Privacy Amplification

For the rest of this work, we let $H_{m,\ell}$ denote a two-universal class of hash functions from $\{0,1\}^m$ to $\{0,1\}^\ell$. Remember that a class of hash functions is *two-universal* if for every distinct $x, x' \in \{0,1\}^m$ and for $F \in_R H_{m,\ell}$, we have $\Pr[F(x) = F(x')] \leq \frac{1}{2^\ell}$.

► **Theorem 8** (Privacy Amplification [29]). *Let $\epsilon \geq 0$. Let $\rho_{XB} \in \text{Pos}(\mathcal{H}_X \otimes \mathcal{H}_B)$ be a cq-state where X is classical and takes values in $\{0,1\}^m$. Let $F \in_R H_{m,\ell}$ be the random variable for a function chosen uniformly at random in a two-universal class of hash functions $H_{m,\ell}$. Then,*

$$\delta(\rho_{F(X)FB}, \mathbb{1} \otimes \rho_{FB}) \leq \frac{1}{2} 2^{-\frac{1}{2}(H_\infty^\epsilon(X|B)_\rho - \ell)} + \epsilon.$$

3.5 Algebra

The following result shows that it is difficult to learn a large matrix using a small number of samples and is proven in the full paper.

► **Theorem 9.** *Let M be an arbitrary $\ell \times n$ binary matrix. Let A be an algorithm that is given as input: $(a_1, b_1), \dots, (a_m, b_m)$ and $(\hat{a}_1, \hat{b}_1), \dots, (\hat{a}_p, \hat{b}_p)$ where $a_i, \hat{a}_i \in \{0,1\}^n$, $b_i, \hat{b}_i \in \{0,1\}^\ell$, $m < n$, p is a polynomial in ℓ , $b_i = M \cdot a_i$ and $\hat{b}_i \neq M \cdot \hat{a}_i$. Then the following statements hold:*

1. For any vector $a \in \{0,1\}^n$ not in the span of $(a_1, \dots, a_m)$, if A outputs a guess b' of $b := M \cdot a$, then $\Pr[b' = b] = O(2^{-\ell})$.
2. Let $x_0, x_1 \in \{0,1\}^n$ be any two distinct vectors not in the span of $(a_1, \dots, a_m)$. Choose $r \in_R \{0,1\}$. If A is additionally given x_0, x_1, y_r where $y_r := M \cdot x_r$ and outputs a guess r' then $\Pr[r' = r] \leq \frac{1}{2} + O(2^{-\ell})$.

4 Definitions: Obfuscation and Variants in the BQSM

In this section, we adapt the notions of obfuscation and one-time programs to the BQSM and introduce a related notion termed program broadcast.

► **Definition 10** (BQS Obfuscation). *A algorithm O is a (r, s) -BQS obfuscator of the class of classical circuits $\mathcal{F}$ if it QPT and satisfies the following:*

1. (functionality) For any circuit $C \in \mathcal{F}$, the circuit described by $O(C)$ can be used to compute C on an input x chosen by the evaluator.
2. For any circuit $C \in \mathcal{F}$, the receiver requires r qmemory to learn an evaluation of C using $O(C)$.
3. (security) For any computationally unbounded adversary $\mathcal{A}_s$ there exists a computationally unbounded simulator $\mathcal{S}_s$ such that for any circuit $C \in \mathcal{F}$,

$$|\Pr[\mathcal{A}_s(O(C)) = 1] - \Pr[\mathcal{S}_s^C(|0\rangle^{\otimes |C|}) = 1]| \leq \text{negl}(|C|).$$

One-time programs, introduced in [7], are similar to obfuscation but can only be used to learn a single evaluation of the program. We adapt this notion to the BQSM.

► **Definition 11** (BQS One-Time Program). *An algorithm O is a (r, s) -BQS one-time compiler for the class of classical circuits $\mathcal{F}$ if it is QPT, satisfies the first three conditions of Definition 10, and the following:*

4. (security) For any computationally unbounded adversary $\mathcal{A}_s$ there exists a computationally unbounded simulator $\mathcal{S}_s$ such that for any circuit $C \in \mathcal{F}$

$$|\Pr[\mathcal{A}_s(O(C)) = 1] - \Pr[\mathcal{S}_s^{1^C}(|0\rangle^{\otimes |C|}) = 1]| \leq \text{negl}(|C|).$$

We introduce the notion of BQS program broadcast which is similar to one-time programs but additionally requires that multiple copies of the encrypted program can only be used to learn a limited number of evaluations. While one-time programs allow for symmetric key cryptography, program broadcast allows for powerful applications such as asymmetric cryptography and tokens.

► **Definition 12** (BQS Program Broadcast). *A (q, s, k) -BQS program broadcast for the class of circuits $\mathcal{C}$ consists of the following QPT algorithms:*

1. $\text{KeyGen}(1^\lambda, t_{\text{end}})$: Outputs a classical key ek .
2. $\text{Br}(s, ek, C)$: Outputs a quantum transmission O_C for the circuit $C \in \mathcal{C}$ during broadcast time (before t_{end}). Outputs ek after broadcast time.
3. $\text{Eval}(\langle O_C, ek \rangle, x)$: Outputs an evaluation y on input x from the transmission $\langle O_C, ek \rangle$ using q qmemory.

Correctness requires that for any circuit $C \in \mathcal{C}$ and input x ,

$$\Pr \left[\text{Eval}(\langle O_C, ek \rangle, x) = C(x) \mid \begin{array}{l} ek \leftarrow \text{KeyGen}(1^\lambda, t_{\text{end}}) \\ O_C \leftarrow \text{Br}(s, ek, C) \end{array} \right] \geq 1 - \text{negl}(\lambda).$$

Security requires that for any computationally unbounded adversary $\mathcal{A}_s$ there exists a computationally unbounded simulator $\mathcal{S}_s$ such that for any circuit $C \in \mathcal{C}$, and $ek \leftarrow \text{KeyGen}(1^\lambda, t_{\text{end}})$,

$$|\Pr[\mathcal{A}_s^{\text{Br}(s, ek, C)}(|0\rangle) = 1] - \Pr[\mathcal{S}_s^{kC}(|0\rangle^{\otimes |C|}) = 1]| \leq \text{negl}(\lambda).$$

5 Definitions: Disappearing Cryptography

In this section, we initiate the study of disappearing cryptography in the BQSM. These concepts were defined earlier in [21, 16] but we adapt the definitions to the BQSM. We define these notions for oblivious transfer, one-time programs, and asymmetric key encryption.

Informally, a state $|\phi\rangle$ is *disappearing* if any adversary that receives $|\phi\rangle$ cannot produce a state with the same “functionality” as $|\phi\rangle$ after a certain point in the transmission. In the BQSM, this point is when the adversary’s qmemory bound applies.

5.1 One-Time Programs

We present the following experiment to define disappearing security for one-time programs. Intuitively, the experiment checks whether an adversary that receives a one-time program can retrieve the evaluation on a random input x that is only revealed after the program. The experiment focuses on matrix branching programs but the same experiment can be applied to any circuit class where sampling can be done efficiently.

Experiment $1\text{TP}_{\Pi, \mathcal{A}}^{\text{Dis}}(m, n, \ell)$: Let $\Pi(m, P)$ be the one-time program protocol with security parameter m on the matrix branching program $P : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$.

1. Sample $x \in_R \{0, 1\}^n$ and a matrix branching program $P : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ uniformly at random.
2. Protocol $\Pi(m, P)$ is executed between the experiment and adversary $\mathcal{A}$.
3. After the execution ends, send x to $\mathcal{A}$.
4. $\mathcal{A}$ outputs a guess p for $P(x)$.
5. The output of the experiment is 1 if $p = P(x)$ and 0 otherwise.

► **Definition 13.** A one-time program protocol Π_s is disappearing in the BQSM if for any adversary $\mathcal{A}_s$,

$$\Pr [1\text{TP}_{\Pi_s, \mathcal{A}_s}^{\text{Dis}}(m, n, \ell) = 1] \leq \frac{1}{2^\ell} + \text{negl}(\min(n, m)).$$

5.2 Encryption

We first recall the definition of a quantum asymmetric (public) key encryption scheme on classical messages. Note that the public key in this setting is quantum so multiple copies must be created and distributed due to the no-cloning theorem. Hence, we add an algorithm **KeySend** that outputs a copy of the quantum public key when queried. In our security experiment, the adversary is allowed to receive a polynomial number of public key copies. We also introduce an algorithm **KeyReceive** which describes how to extract a reusable classical key from a public key copy to use for encryption.

► **Definition 14** (Quantum Asymmetric Key Encryption). A quantum asymmetric key encryption scheme Π over classical message space $\mathcal{M}$ consists of the following QPT algorithms:

- $\text{Gen}(1^\lambda) : \text{Outputs a private key } sk.$
- $\text{KeySend}(sk) : \text{Outputs a quantum public key copy } \rho_{pk}.$
- $\text{KeyReceive}(\rho_{pk}) : \text{Extracts a key } k \text{ from } \rho_{pk}.$
- $\text{Enc}(k, \mu) : \text{Outputs a ciphertext } \rho_{ct} \text{ for } \mu \in \mathcal{M}.$
- $\text{Dec}(sk, \rho_{ct}) : \text{Outputs a message } \mu' \text{ by decrypting } \rho_{ct}.$

► **Definition 15** (Correctness). A quantum asymmetric key encryption scheme Π is correct if for any message $\mu \in \mathcal{M}$,

$$\Pr \left[\text{Dec}(sk, \rho_{ct}) = \mu \mid \begin{array}{l} sk \leftarrow \text{Gen}(1^\lambda) \\ \rho_{pk} \leftarrow \text{KeySend}(sk) \\ k \leftarrow \text{KeyReceive}(\rho_{pk}) \\ \rho_{ct} \leftarrow \text{Enc}(k, \mu) \end{array} \right] \geq 1 - \text{negl}(\lambda).$$

We now present an experiment to test disappearing security under lunchtime chosen ciphertext attack (qDCCA1) in the BQSM. Recall, IND-CCA1 [8] tests security against an adversary that can query the encryption oracle at any point in the experiment and can query the decryption oracles at any point before the challenge ciphertext is received. To satisfy *disappearing* security under CCA1, we require that an adversary cannot win the IND-CCA1 experiment even if the private key is revealed after the challenge ciphertext is given.

Experiment $\text{AsyK}_{\Pi, \mathcal{A}}^{\text{qDCCA1}}(\lambda)$:

1. Sample a private key $sk \leftarrow \text{Gen}(1^\lambda)$ and bit $b \in_R \{0, 1\}$.
2. Generate a public key $\rho_{pk} \leftarrow \text{KeySend}(sk)$.
3. Extract key $k \leftarrow \text{KeyReceive}(\rho_{pk})$.
4. Adversary outputs two messages $m_0, m_1 \leftarrow \mathcal{A}^{\text{KeySend}(sk), \text{Enc}(k, \cdot), \text{Dec}(sk, \cdot)}$.
5. Send $\mathcal{A}^{\text{KeySend}(sk), \text{Enc}(k, \cdot)}$ the ciphertext $\rho_{ct} \leftarrow \text{Enc}(k, m_b)$.
6. Give $\mathcal{A}^{\text{KeySend}(sk), \text{Enc}(k, \cdot)}$ the private key sk .
7. $\mathcal{A}^{\text{KeySend}(sk), \text{Enc}(k, \cdot)}$ outputs a guess b' .
8. The output of the experiment is 1 if $b' = b$ and 0 otherwise.

We can similarly construct disappearing experiment in the symmetric key case, denoted $\text{SymK}_{\Pi, \mathcal{A}}^{\text{qDCCA1}}(\lambda)$ by deleting steps 2 & 3 and replacing k with sk in the asymmetric experiment.

► **Definition 16** (Security). An asymmetric key encryption scheme Π_s satisfies disappearing indistinguishability against chosen ciphertext attacks (qDCCA1) if for any adversary $\mathcal{A}_s$,

$$\Pr [\text{AsyK}_{\Pi_s, \mathcal{A}_s}^{\text{qDCCA1}}(\lambda) = 1] \leq \frac{1}{2} + \text{negl}(\lambda),$$

6 One-Time Programs

In this section, we give the result for an information-theoretic secure one-time program in the BQSM. The construction and proof are provided in the full paper as they are long and technical. Our construction is also disappearing meaning a one-time program cannot be evaluated after the transmission ends.

► **Theorem 17.** *There exists an algorithm $\mathcal{O}_s$ that is a disappearing information-theoretically secure $(0, s)$ one-time compiler for the class of polynomial classical circuits against any computationally unbounded adversary with s qmemory bound.*

7 Program Broadcast

In this section, we construct a BQS program broadcaster as introduced in Definition 12. This will allow us to tackle asymmetric key encryption in later sections. Let $\mathcal{C}_m = \{\mathcal{C}_{n,m}\}_{n \leq 2^m}$ where $\mathcal{C}_{n,m}$ is a set of polynomial-size circuits in n , of input size n , and output size m . Note that any polynomial-size circuits belongs to such a class by adding null outputs to ensure that $n \leq 2^m$.

► **Construction 18** (BQS Program Broadcaster). *The $(12m, s, \frac{s}{2m})$ -BQS program broadcast scheme for the class $\mathcal{C}_m$ until time t_{end} is as follows:*

- **KeyGen** $(1^\lambda, t_{\text{end}})$: *Choose uniformly at random a $(12m) \times (12m)$ binary matrix M and let $F(x) := M \cdot x$ be the corresponding program. Choose a two-universal hash function $H \in_R \mathcal{H}_{12m, m}$. Output $ek := (M, H, t_{\text{end}})$.*
- **Br** (s, ek, P) : *Let $P \in \mathcal{C}_m$. If queried before time t_{end} , then:*
 1. *Randomly choose $\mathbf{r}, \mathbf{x} \in_R \{0, 1\}^{12m}$.*
 2. *Compute $\theta := F(\mathbf{r})$ and $c := H(\mathbf{x})$.*
 3. *Send $|\mathbf{x}\rangle_\theta := |x_1\rangle_{\theta_1} \dots |x_{12m}\rangle_{\theta_{12m}}$.*
 4. *Send $\mathcal{O}_s(P \oplus c)$.*
 5. *Send $\mathbf{r}$.*

The entire transmission is denoted as $O_P^{r, x}$.

If queried after time t_{end} , then apply the qmemory bound and output ek .

- **Eval** $(\langle O_P^{r, x}, ek \rangle, v)$:
 1. *Store $|\mathbf{x}\rangle_\theta$.*
 2. *Evaluate the one-time program $\mathcal{O}_s(P \oplus c)$ on v to obtain $P(v) \oplus c$.*
 3. *After $\mathbf{r}$ and F are received, compute θ .*
 4. *Measure $|\mathbf{x}\rangle_\theta$ in the basis θ to obtain $\mathbf{x}$.*
 5. *Use H to compute $c = H(\mathbf{x})$ and obtain $P(v)$.*

► **Theorem 19** (Security). *Construction 18 is a $(12m, s, \frac{s}{2m})$ -BQS program broadcaster for the class $\mathcal{C}_m$.*

Proof. Let $P \in \mathcal{C}_m$. It is clear that the receiver requires $12m$ qmemory to learn a single evaluation of P from the broadcast.

In terms of security, an adversary $\mathcal{A}_s$ can receive a polynomial number, say p , outputs of the broadcast **Br** (s, ek, P) . From these states, the adversary obtains $(|\mathbf{x}_i\rangle_{\theta_i})_{i \in [p]}$ and one-time programs $(\mathcal{O}_s(P \oplus c_i))_{i \in [p]}$ where $c_i := H(\mathbf{x}_i)$.

$\mathcal{A}_s$ can be simulated by $\mathcal{S}_s$ which has access to a single oracle query to $P \oplus c_i$ for each $i \in [p]$ by the security of one-time protocol (Theorem 17). The lemma below shows that $\mathcal{S}_s$ can learn at most $\frac{s}{2m}$ values in $\{c_i\}_{i \in [p]}$. This implies that $\mathcal{A}_s$ can learn at most $\frac{s}{2m}$ evaluations of P from the broadcast thus achieving program broadcast security.

► **Lemma 20.** *$\mathcal{S}_s$ can distinguish at most $\frac{s}{2m}$ terms in $\{c_i\}_{i \in [p]}$ from random.*

Proof. Assume that there exists $\frac{s}{2m}$ values $\mathbf{x}_{i_1}, \dots, \mathbf{x}_{i_{\frac{s}{2m}}}$ that are distinguishable from random.

Instead of the broadcaster sending $|\mathbf{x}_i\rangle_{\theta_i}$, a standard purification argument shows it is sufficient to show security for the protocol with the following modification. For each qubit supposed to be sent, the broadcaster instead prepares an EPR state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ and sends one half to $\mathcal{S}_s$ and keeps one half. Then the broadcaster measures its halves of the EPR pairs in a random BB84 basis in $\{+, \times\}^{12m}$ after the memory bound of $\mathcal{S}_s$ applies. Let Θ_i and X_i be the random variables representing the broadcaster's choice of measurement basis and outcome. Let $X_I := X_{i_1} X_{i_2} \dots X_{i_{\frac{s}{2m}}}$ and $\Theta_I := \Theta_{i_1} \Theta_{i_2} \dots \Theta_{i_{\frac{s}{2m}}}$. The Uncertainty Relation (Lemma 4) implies that for $\gamma = \frac{1}{12}$, there exists ϵ (negligible in m) such that $H_\infty^\epsilon(X_I | \Theta_I) \geq (\frac{1}{2} - 2\gamma)(6s) = 2s$. Let B be the random s qubit state the adversary stores when the memory bound applies. By the chain rule for min-entropy (Lemma 3),

$$H_\infty^\epsilon(X_I | \Theta_I B) \geq H_\infty^\epsilon(X_I | \Theta_I) - s \geq s.$$

Hence, by the Generalized Min-entropy Splitting Lemma 7, there exists a random variable C and negligible value $\epsilon' > 0$ such that,

$$H_{\infty}^{\epsilon'}(X_{i_C} | \Theta_I CB) \geq 2m - 4.$$

By the Privacy Amplification Theorem 8,

$$\begin{aligned} \delta(\rho_{H(X_{i_C})CH\Theta_I B}, \mathbb{1} \otimes \rho_{CH\Theta_I B}) \\ \leq \frac{1}{2} 2^{-\frac{1}{2}(H_{\infty}^{\epsilon}(X_{i_C} | \Theta_I CB)_{\rho} - m)} + \epsilon = O(2^{-m}) = \text{negl}(n). \end{aligned}$$

The final equality is because $m \geq \lg^2(n)$ which means 2^{-m} is negligible in n . This gives a contradiction so there is less than $\frac{s}{2m}$ values c_i which are distinguishable from random. $\blacktriangleleft$

This completes the proof of Theorem 19. $\blacktriangleleft$

8 Encryption Schemes

We construct information-theoretically secure symmetric encryption in the BQSM and, then, show how to upgrade it to the asymmetric setting.

8.1 Symmetric Key Encryption

In this section, we present a private key encryption scheme that satisfies disappearing indistinguishability under chosen ciphertext attacks (qDCCA1).

► **Construction 21.** *The symmetric key encryption scheme Π_{SymK} against adversaries with q memory bound s is as follows.*

- **Gen(1^λ):** Let $m := \lceil (\lg \lambda)^{3/2} \rceil$. Choose at random strings $q, w \in_R \{0, 1\}^m$, a $(2m) \times (2m)$ invertible binary matrix M and string $z \in_R \{0, 1\}^{2m}$. Define $S(x) := M \cdot x + z$, and $S'(x) := qx + w \pmod{2^m}$. The private key is $k := (q, w, M, z)$.
- **Enc(s, k, μ):** Let $\ell := |\mu|$. Choose strings $a, b \in_R \{0, 1\}^m$ and define $f(x) := ax + b \pmod{2^m}$. Construct the following program:

$$E_{k,f,\mu}(y) = \begin{cases} S'(f(x)) \parallel \mu & \text{if } y = S(x \parallel f(x)) \\ \perp & \text{otherwise.} \end{cases}$$

The encryption is $\rho_{ct} \leftarrow \langle \mathcal{O}_s(f), \mathcal{O}_s(E_{k,f,\mu}) \rangle$ sent in sequence; first $\mathcal{O}_s(f)$ and then $\mathcal{O}_s(E_{k,f,\mu})$.

- **Dec(k, ρ_{ct}):** Check ρ_{ct} has the correct format (see note below). Evaluate the first one-time program on a random input v to obtain $f(v)$. Evaluate the second one-time program on $S(v \parallel f(v))$. If the output is of the form $S'(f(v)) \parallel \hat{\mu}$ (for some string $\hat{\mu}$) then output $\hat{\mu}$ and $\perp$ otherwise.

► **Theorem 22 (Security).** *Construction 21 (Π_{SymK}) satisfies qDCCA1 security against computationally unbounded adversaries with q memory bound s .*

Proof. An adversary $\mathcal{A}_s$ in the qDCCA1-security experiment requests a polynomial number, say Q_e , of encryption queries and a polynomial number, say Q_d , of decryption queries. Denote the ciphertexts produced by all encryption queries to the oracle as $\langle \mathcal{O}(f_i), \mathcal{O}(E_{k,f_i,\mu_i}) \rangle_{i \in [Q_e]}$ and denote the decryption queries submitted to the oracle as $\langle \mathcal{O}(\hat{f}_j), \mathcal{O}(\hat{E}_j) \rangle_{j \in [Q_d]}$ for $\mathbf{NC}^1$ functions $\hat{f}_j : \{0, 1\}^m \rightarrow \{0, 1\}^m$ and $\hat{E}_j : \{0, 1\}^{2m} \rightarrow \{0, 1\}^{m+\ell}$. Any decryption query not of this form is immediately rejected as an invalid ciphertext.

$\mathcal{A}_s$ cannot learn anything from the encryption queries alone as such attacks can be simulated with only a single oracle query to each program in $(E_{k,f_i,\mu_i})_{i \in [Q_e]}$. All such queries will yield $\perp$ except with negligible probability by the security of one-time programs (Theorem 17).

Now, consider what $\mathcal{A}_s$ learns from the first decryption query $\langle \mathcal{O}(\hat{f}_0), \mathcal{O}(\hat{E}_0) \rangle$. Suppose $\mathcal{A}_s$ requested the encryption queries $\langle \mathcal{O}(f_i), \mathcal{O}(E_{k,f_i,\mu_i}) \rangle_{i \in [d_0]}$ prior to requesting the first decryption query. These ciphertexts may be utilized in the construction of the decryption query. Let A_0 be the sub-algorithm of $\mathcal{A}_s$ which produces the first decryption query using these ciphertexts. Hence, we write $\langle \mathcal{O}_s(\hat{f}_0), \mathcal{O}_s(\hat{E}_0) \rangle \leftarrow A_0(\langle \mathcal{O}(f_i), \mathcal{O}(E_{k,f_i,\mu_i}) \rangle_{i \in [d_0]})$.

As a side note, prior to sending $\mathcal{O}_s(\hat{E}_0)$, the memory bound for $(\mathcal{O}_s(f_i))_{i \in [d_0]}$ must have already been applied; thus there exists values $(v_i)_{i \in [d_0]}$ such that $\mathcal{A}_s$ can only distinguish $f_i(x)$ from random on $x = v_i$ for all $i \in [d_0]$. This is because these functions are of the form $f_i(x) = a_i x + b_i \pmod{2^m}$ so learning one evaluation is not sufficient to distinguish the evaluation on another input from random.

When the oracle receives the first decryption query, it evaluates the first one-time program on a random input say $\hat{v}_0$ to get $\hat{f}_0(\hat{v}_0)$. Next, it evaluates the second one-time program on $S(\hat{v}_0 \| \hat{f}_0(\hat{v}_0))$ and gets an output say $\hat{y}_0 \| \hat{\mu}_0$ (or $\perp$). Let Or_0 denote the sub-algorithm of the oracle which performs this evaluation. Then this entire interaction can be described as $\hat{y}_0 \| \hat{\mu}_0 \leftarrow Or_0(A_0(\langle \mathcal{O}(f_i), \mathcal{O}(E_{k,f_i,\mu_i}) \rangle_{i \in [d_0]}))$. Critically, the algorithm A_0 is oblivious of S and Or_0 has access to only a single evaluation of S , namely $S(\hat{v}_0 \| \hat{f}_0(\hat{v}_0))$.

The purpose of this description is to highlight that this entire procedure is essentially performed by an algorithm with access to only a single evaluation of S . By Theorem 9, this algorithm cannot guess $S(x)$ on any input $x \neq \hat{v}_0 \| \hat{f}_0(\hat{v}_0)$. By the security of one-time programs, this algorithm can be simulated with a simulator that has access to only a single query to each program $(E_{k,f_i,\mu_i})_{i \in [d_0]}$ instead of $(\mathcal{O}_s(E_{k,f_i,\mu_i}))_{i \in [d_0]}$. The query to E_{k,f_i,μ_i} will yield $\perp$ except with negligible probability unless $\hat{v}_0 \| f_i(\hat{v}_0) = \hat{v}_0 \| \hat{f}_0(\hat{v}_0)$ since the simulator cannot guess $S(x)$ on $x \neq \hat{v}_0 \| \hat{f}_0(\hat{v}_0)$. If the condition $\hat{v}_0 \| f_i(\hat{v}_0) = \hat{v}_0 \| \hat{f}_0(\hat{v}_0)$ is not satisfied for any $i \in [d_0]$ then the simulator will receive $\perp$ from all its oracle queries except with negligible probability and thus there is a negligible probability that $\hat{y}_0 = S'(\hat{f}_0(\hat{v}_0))$. Note that, there is at most a single value i that satisfies this condition except with negligible probability since the parameters of the functions $(f_i)_{i \in [d_0]}$ are chosen independently and at random.

Assume this condition is satisfied only by the function f_n where $n \in [d_0]$. There is negligible chance that $\hat{v}_0 = v_n$ and so $\mathcal{A}_s$ cannot distinguish $f_n(\hat{v}_0)$ or equivalently $\hat{f}_0(\hat{v}_0)$ from random. The other values used by the oracle in the decryption query are: v , $S(v \| \hat{f}_0(\hat{v}_0))$ and $S'(\hat{f}_0(\hat{v}_0))$. Even if $\mathcal{A}_s$ learns all these values, it does not help in determining functions S and S' since $\hat{f}_0(\hat{v}_0)$ is indistinguishable from random.

At the end of this argument, $\mathcal{A}_s$ cannot distinguish the key k from random. Notice that the only requirement to apply this argument on a decryption query is that $\mathcal{A}$ cannot distinguish k from random when it submits the query. Hence, this argument can be applied to all decryption queries and it can be deduced inductively that $\mathcal{A}_s$ cannot distinguish k from random when it receives the challenge ciphertext. Let $\langle \mathcal{O}_s(f), \mathcal{O}_s(E_{k,f,m_b}) \rangle$ be the challenge ciphertext. By one-time program security, the adversary's access to these programs can be simulated with a single evaluation to each program. Hence, it is clear that the probability that the adversary guesses b is upper bounded by $\frac{1}{2} + \text{negl}(n)$. This still holds if k is later revealed since one-time programs disappear after their transmission ends by Theorem 17. $\blacktriangleleft$

8.2 Asymmetric Key Encryption

We now look into the asymmetric key setting and construct an asymmetric key encryption scheme with information-theoretic disappearing security under chosen-ciphertext attacks in the BQSM. The private key is a large matrix and the public key is a program broadcast of the matrix. Since different users will likely learn different evaluations, an evaluation can be used as a secret key to encrypt messages in the same way as in Construction 21. Note that honest users need a small qmemory to process the broadcast as described in Construction 18. This is in contrast to the private key setting where no qmemory is required.

► **Construction 23.** Let $\bar{\Pi} = (\bar{Gen}, \bar{Enc}, \bar{Dec})$ be given in Construction 21 for symmetric key encryption and let $\Pi_{Br} = (KeyGen, Br, Eval)$ be the algorithms for program broadcast given in Construction 18. The asymmetric key encryption scheme Π_{AsyK} against adversaries with qmemory bound s is as follows:

- **Gen**($1^\lambda, s$): Let $m = 100\lceil(\lg \lambda)^3\rceil$ and $n = \max(\lceil\frac{6s}{m}\rceil + 1, m)$. Choose uniformly at random $m \times n$ binary matrix M and for $x \in \{0, 1\}^m$, let $P(x) := M \cdot x$ be the corresponding program. Generate $ek \leftarrow KeyGen(1^\lambda, t_{end})$. The private key is $sk = (ek, P)$.
- **KeySend**(s, sk): If queried before t_{end} , then output $O_P \leftarrow Br(s, ek, P)$.
If queried after t_{end} , the qmemory bound applies and output ek .
Let $\rho_{pk} \leftarrow \langle O_P, ek \rangle$ denote the public key copy.
- **KeyReceive**(ρ_{pk}): Randomly choose $v \in_R \{0, 1\}^n$. Evaluate the public key $P(v) \leftarrow Eval(\rho_{pk}, v)$ and output the key $k_v = (v, P(v))$.
- **Enc**(s, k_v, μ): Output $\langle v, \bar{Enc}(s, P(v), \mu) \rangle$ ³.
- **Dec**($sk, \langle v, \rho_{ct} \rangle$): Obtain $P(v)$ using sk and v . Output $\bar{Dec}(P(v), \rho_{ct})$.

The algorithm **KeyReceive** requires $100(\lg \lambda)^3$ qmemory to run and the scheme is secure against adversaries with s qmemory, where s can be any polynomial in λ . Hence, by setting $m := 100(\lg \lambda)^3$, the gap between the qmemory of the honest user and the adversary is m vs. $e^{\sqrt[3]{m}}$.

► **Theorem 24 (Security).** Construction 23 (Π_{AsyK}) satisfies qDCCA1 security against computationally unbounded adversaries with qmemory bound s .

Proof. An adversary $\mathcal{A}_s$ in the qDCCA1 security experiment can request a polynomial number of public key copies ρ_{pk} . The proof can be realized in the following steps.

1. By the security of our program broadcast protocol (Theorem 19) $\mathcal{A}_s$ can be simulated with an algorithm $\mathcal{S}_s$ that is given $\frac{6s}{m}$ queries to P .
2. By Theorem 9, since M is a matrix of dimension $\lceil\frac{m}{12}\rceil \times n$ and $n > \frac{6s}{m}$, $\mathcal{S}_s$ cannot guess the output of $P(x) = M \cdot x$ on a random input except with negligible probability.
3. This means that there is negligible chance $\mathcal{S}_s$ can determine the sub-key k_v extracted from the public key ρ_{pk} in the experiment which reduces the proof to the private key setting. Theorem 22 gives the result. ◀

9 Note on Noisy Communication

Our protocols assume that the communication between the sender and receiver is error-free. It is assumed that the sender has a perfect quantum source which when requested will produce one and only one qubit in the correct state. Note if the source accidentally produces

³ The string $P(v)$ is of length $m = 100\lceil(\lg \lambda)^3\rceil$, so it is of sufficient length such that it can be interpreted as the secret key in the symmetric-key encryption scheme $\bar{\Pi}$.

two copies of a qubit then the adversary can measure the qubit in two bases and break the 1-2 oblivious transfer scheme. Furthermore, it requires the honest evaluator to measure all qubits without error.

Fortunately, in [10] the authors used techniques based on information reconciliation to allow an honest evaluator and sender in the quantum 1-2 oblivious transfer scheme with imperfect apparatus to perform error correction without compromising the security of the scheme. The same techniques are applicable to our protocols since the quantum component of our one-time programs consist of oblivious transfer transmissions. However, we leave a more rigorous treatment of these issues as an avenue for future work.

References

- 1 Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. In *Annual international cryptology conference*, pages 1–18. Springer, 2001. doi:10.1007/3-540-44647-8_1.
- 2 Mohammed Barhoush and Louis Salvail. How to sign quantum messages. *arXiv preprint arXiv:2304.06325*, 2023. doi:10.48550/arXiv.2304.06325.
- 3 Howard Barnum, Claude Crépeau, Daniel Gottesman, Adam Smith, and Alain Tapp. Authentication of quantum messages. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 449–458, 2002. doi:10.1109/SFCS.2002.1181969.
- 4 Khashayar Barooti, Giulio Malavolta, and Michael Walter. A simple construction of quantum public-key encryption from quantum-secure one-way functions. *arXiv preprint*, 2023. arXiv:2303.01143.
- 5 Shalev Ben-David and Or Sattath. Quantum tokens for digital signatures. *Quantum*, 7:901, 2023. doi:10.22331/Q-2023-01-19-901.
- 6 Nir Bitansky, Ran Canetti, Henry Cohn, Shafi Goldwasser, Yael Tauman Kalai, Omer Paneth, and Alon Rosen. The impossibility of obfuscation with auxiliary input or a universal simulator. In *Annual Cryptology Conference*, pages 71–89. Springer, 2014. doi:10.1007/978-3-662-44381-1_5.
- 7 Anne Broadbent, Gus Gutoski, and Douglas Stebila. Quantum one-time programs. In *Annual Cryptology Conference*, pages 344–360. Springer, 2013.
- 8 Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Annual international cryptology conference*, pages 13–25. Springer, 1998. doi:10.1007/BFB0055717.
- 9 Ivan Damgård, Serge Fehr, Louis Salvail, and Christian Schaffner. Cryptography in the bounded-quantum-storage model. *SIAM Journal on Computing*, 37(6):1865–1890, 2008. doi:10.1137/060651343.
- 10 Ivan B Damgård, Serge Fehr, Renato Renner, Louis Salvail, and Christian Schaffner. A tight high-order entropic quantum uncertainty relation with applications. In *Advances in Cryptology-CRYPTO 2007: 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007. Proceedings 27*, pages 360–378. Springer, 2007. doi:10.1007/978-3-540-74143-5_20.
- 11 Yevgeniy Dodis, Willy Quach, and Daniel Wichs. Speak much, remember little: Cryptography in the bounded storage model, revisited. *Cryptology ePrint Archive*, 2021. URL: <https://eprint.iacr.org/2021/1270>.
- 12 Yevgeniy Dodis, Willy Quach, and Daniel Wichs. Authentication in the bounded storage model. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 737–766. Springer, 2022. doi:10.1007/978-3-031-07082-2_26.
- 13 Javad Doliskani. Efficient quantum public-key encryption from learning with errors. *arXiv preprint*, 2021. arXiv:2105.12790.

- 14 Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 553–562. IEEE, 2005. doi:10.1109/SFCS.2005.60.
- 15 Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. One-time programs. In *Annual International Cryptology Conference*, pages 39–56. Springer, 2008. doi:10.1007/978-3-540-85174-5_3.
- 16 Daniel Gottesman. Uncloneable encryption. *arXiv preprint*, 2002. arXiv:quant-ph/0210062.
- 17 Daniel Gottesman and Isaac Chuang. Quantum digital signatures. *arXiv preprint*, 2001. arXiv:quant-ph/0105032.
- 18 Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In *Theory of Cryptography: 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings 7*, pages 308–326. Springer, 2010. doi:10.1007/978-3-642-11799-2_19.
- 19 Jiaxin Guan, Daniel Wichs, and Mark Zhandry. Incompressible cryptography. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 700–730. Springer, 2022. doi:10.1007/978-3-031-06944-4_24.
- 20 Jiaxin Guan and Mark Zhandary. Simple schemes in the bounded storage model. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 500–524. Springer, 2019. doi:10.1007/978-3-030-17659-4_17.
- 21 Jiaxin Guan and Mark Zhandry. Disappearing cryptography in the bounded storage model. In *Theory of Cryptography Conference*, pages 365–396. Springer, 2021. doi:10.1007/978-3-030-90453-1_13.
- 22 Akinori Kawachi, Takeshi Koshihara, Harumichi Nishimura, and Tomoyuki Yamakami. Computational indistinguishability between quantum states and its cryptographic application. In *Eurocrypt*, volume 3494, pages 268–284. Springer, 2005. doi:10.1007/11426639_16.
- 23 Joe Kilian. Founding cryptography on oblivious transfer. In *Proc. ACM Symposium on Theory of Computing*, STOC '88, pages 20–31, New York, NY, USA, 1988. ACM. doi:10.1145/62212.62215.
- 24 Robert König and Renato Renner. Sampling of min-entropy relative to quantum knowledge. *IEEE Transactions on Information Theory*, 57(7):4760–4787, 2011. doi:10.1109/TIT.2011.2146730.
- 25 Robert König, Renato Renner, and Christian Schaffner. The operational meaning of min- and max-entropy. *IEEE Transactions on Information theory*, 55(9):4337–4347, 2009. doi:10.1109/TIT.2009.2025545.
- 26 Ueli M Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, 1992. doi:10.1007/BF00191321.
- 27 Michele Mosca, Alain Tapp, and Ronald de Wolf. Private quantum channels and the cost of randomizing quantum information. *arXiv preprint*, 2000. arXiv:quant-ph/0003101.
- 28 Ran Raz. Fast learning requires good memory: A time-space lower bound for parity learning. *Journal of the ACM (JACM)*, 66(1):1–18, 2018. doi:10.1145/3186563.
- 29 Renato Renner and Robert König. Universally composable privacy amplification against quantum adversaries. In *Theory of Cryptography Conference*, pages 407–425. Springer, 2005. doi:10.1007/978-3-540-30576-7_22.
- 30 Christian Schaffner. *Cryptography in the Bounded-Quantum-Storage Model*. PhD thesis, Aarhus Universitet, 2007. arXiv:0709.0289.
- 31 Stephanie Wehner, Christian Schaffner, and Barbara M. Terhal. Cryptography from noisy storage. *Phys. Rev. Lett.*, 100:220502, June 2008. doi:10.1103/PhysRevLett.100.220502.
- 32 Andreas Winter. Coding theorem and strong converse for quantum channels. *IEEE Transactions on Information Theory*, 45(7):2481–2485, 1999. doi:10.1109/18.796385.

Leakage-Resilience of Shamir’s Secret Sharing: Identifying Secure Evaluation Places

Jihun Hwang 

Department of Computer Science, Purdue University, West Lafayette, IN, USA

Hemanta K. Maji 

Department of Computer Science, Purdue University, West Lafayette, IN, USA

Hai H. Nguyen 

Department of Computer Science, ETH, Zürich, Switzerland

Xiuyu Ye 

Department of Computer Science, Purdue University, West Lafayette, IN, USA

Abstract

Can Shamir’s secret-sharing protect its secret even when all shares are partially compromised?

For instance, repairing Reed-Solomon codewords, when possible, recovers the entire secret in the corresponding Shamir’s secret sharing. Yet, Shamir’s secret sharing mitigates various side-channel threats, depending on where its “secret-sharing polynomial” is evaluated. Although most evaluation places yield secure schemes, none are known explicitly; even techniques to identify them are unknown. Our work initiates research into such classifier constructions and derandomization objectives.

In this work, we focus on Shamir’s scheme over prime fields, where every share is required to reconstruct the secret. We investigate the security of these schemes against single-bit probes into shares stored in their native binary representation. Technical analysis is particularly challenging when dealing with Reed-Solomon codewords over prime fields, as observed recently in the code repair literature. Furthermore, ensuring the statistical independence of the leakage from the secret necessitates the elimination of any subtle correlations between them.

In this context, we present:

1. An efficient algorithm to classify evaluation places as secure or vulnerable against the least-significant-bit leakage.
2. Modulus choices where the classifier above extends to any single-bit probe per share.
3. Explicit modulus choices and secure evaluation places for them.

On the way, we discover new bit-probing attacks on Shamir’s scheme, revealing surprising correlations between the leakage and the secret, leading to vulnerabilities when choosing evaluation places naively.

Our results rely on new techniques to analyze the security of secret-sharing schemes against side-channel threats. We connect their leakage resilience to the orthogonality of square wave functions, which, in turn, depends on the 2-adic valuation of rational approximations. These techniques, novel to the security analysis of secret sharings, can potentially be of broader interest.

2012 ACM Subject Classification Theory of computation → Cryptographic primitives; Security and privacy → Cryptanalysis and other attacks

Keywords and phrases Shamir’s secret sharing, leakage resilience, physical bit probing, secure evaluation places, secure modulus choice, square wave families, LLL algorithm, Fourier analysis

Digital Object Identifier 10.4230/LIPIcs.ITC.2025.3

Related Version *Extended version with full proof:* <https://www.cs.purdue.edu/homes/hmaji/papers/HMNY24.pdf> [27]

Funding Hemanta K. Maji: Partly supported by CNS-2055605 and CCF-2327981.

Hai H. Nguyen: Partly supported by the Zurich Information Security & Privacy Center (ZISC), ETH Zurich.



© Jihun Hwang, Hemanta K. Maji, Hai H. Nguyen, and Xiuyu Ye;
licensed under Creative Commons License CC-BY 4.0

6th Conference on Information-Theoretic Cryptography (ITC 2025).

Editor: Niv Gilboa; Article No. 3; pp. 3:1–3:20



Leibniz International Proceedings in Informatics

LIPIcs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Secret-sharing schemes protect their secrets when only a few shares are compromised. *Side-channel attacks* have repeatedly circumvented their security by accumulating partial information from all shares [28, 29, 10]. For instance, *repairing Reed-Solomon codes* [20, 21], when possible, recovers the entire secret in the corresponding *Shamir's secret sharing* [40] by downloading a small amount of information per share. More alarmingly, ingenious side-channel attacks have revealed critical information about cryptographic secrets (without completely recovering them). Securing our secret-sharing schemes against various side-channel threats has become even more compelling due to the ongoing NIST standardization efforts [8], considering their wide use in key distribution [37, 43], masking schemes [10, 18], and other higher-level primitives like secure computation [17].

Local leakage resilience [4, 19] is a security metric for secret sharing against a broad spectrum of side-channel threats that leak from each share independently. Local leakages are surprisingly powerful; even single-bit probes into every share partially reveal an additively shared secret [33, 1, 34, 14]. Shamir's secret-sharing is a more promising alternative – its security depends on where its secret-sharing polynomial is evaluated. Most evaluation places, in particular, ensure that the cumulative leakage from bit probes into shares is statistically independent of the secret [33, 35]. However, not one choice is known explicitly; even techniques to identify them have yet to be discovered. As a result, *NIST can neither recommend evaluation places for Shamir's secret sharing nor certify their security against such attacks*. Towards alleviating this situation, it is natural to wonder:

Question: *Is there an algorithm to determine whether the picked evaluation places yield a locally leakage-resilient Shamir's secret sharing?*

Any meaningful classifier in this context must have the following features.

1. *No false positives.* No evaluation places can be incorrectly determined to be leakage-resilient; otherwise, they could be picked unbeknownst to the honest parties.
2. *A small number of false negatives.* Ideally, the algorithm should correctly identify most (or at least a significant fraction) of the leakage-resilient evaluation places.
3. *Efficiency.* The runtime of the classifier should not be “prohibitively large.”

In fact, *explicitly identifying secure evaluation places* would be ideal. Our work initiates research into such classifier constructions and derandomization objectives.

Summary of our results. We consider Shamir's schemes where *shares of all parties are required to reconstruct the secret* and investigate their security against *arbitrary single bit-probe in each share*. We present such classifiers for *Mersenne* and *Fermat* prime modulus. Our algorithms have $\text{poly}(\log p)$ running time and $\sqrt{p} \cdot \text{poly}(\log p)$ false negatives for prime modulus p . For the two-party case, we present secure evaluation places explicitly. The technical workhorse is our classifier for the specific leakage that obtains each share's *least significant bit* (LSB); this classifier works for *arbitrary prime modulus*. Our classifier is accurate; we present *new bit-probing attacks* on those identified to be insecure.

Summary of our key technical challenge. For an arbitrary prime modulus $p \geq 3$, define the function $\text{LSB}: F_p \rightarrow F_2$ by $\text{LSB}(x) := 0$, for $x \in \{0, 2, \dots, (p-1)\}$; otherwise, $\text{LSB}(x) := 1$. Fix arbitrary elements $\alpha_1, \alpha_2 \in F_p$.

Technical Question: For a uniformly random $X \in F_p$, are the distributions $\text{LSB}(\alpha_1 \cdot X)$ and $\text{LSB}(\alpha_2 \cdot X)$ statistically independent?

Answering this technical question is challenging because $x \mapsto \text{LSB}(x)$ is a *non-linear map*. Linear maps are either (perfectly) independent or (completely) correlated; answering this question for them is easy. Subtle correlations can surreptitiously manifest between non-linear maps, which is the case here. The pattern of (α_1, α_2) resulting in statistically independent distributions is highly non-trivial. We prove that it depends on the *2-adic valuation of their rational approximation*; our classifier algorithm is outlined below.

1. Solve for relatively prime integers $u, v \in \{-\lceil\sqrt{p}\rceil, \dots, 0, \dots, \lceil\sqrt{p}\rceil\}$ such that $u \cdot \alpha_2 = v \cdot \alpha_1 \pmod{p}$.
2. Distributions are independent if (and only if) u/v has non-zero 2-adic valuation; i.e., either u or v is even.
3. Otherwise, for odd u and v , the 2-adic valuation of u/v is 0, and the dependence between these two distributions is $1/|u \cdot v|$. When this dependence is significant, we identify *new side-channel attacks*.

The connection between 2-adic valuations with security of secret sharings is novel and possibly of broader interest. Our work highlights the challenges in determining the leakage resilience of secret sharing. There are several natural follow-up questions; Section 3 presents a few and the hurdles in approaching them.

► **Remark 1 (Recent relevant works).** Maji et al. [35] and, very recently, Nguyen [38] drew inspiration from our approach and constructed such classifiers over *characteristic-2 composite-order fields*. The analogous map in their technical analysis is F_2 -linear, making their technical question approachable via elementary “rank arguments” (a.k.a., dual distance of the concatenated Reed-Solomon codes over the binary alphabet). Analyzing non-linear leakage in the related literature on repairing Reed-Solomon codewords has also been technically challenging; *non-linear repairing* was only recently addressed [13, 12]. We summarize this discussion and other prior relevant works in the full version [27] of the paper.

► **Remark 2 (Other leakage-resilient alternatives).** The additive and Shamir’s schemes are deployed widely. It is crucial to determine their security; this work contributes to this effort. New constructions (like [5, 2, 41, 3, 30, 6, 15, 41, 16, 26, 11, 36, 9]) cannot match their simplicity and high information rate or replace them in security technologies.

1.1 Basic Definitions & Our Formal Problem Statement

Shamir’s secret-sharing scheme. Shamir’s secret-sharing scheme among n parties with reconstruction threshold k over a finite field F and distinct evaluation places $\alpha_1, \alpha_2, \dots, \alpha_n \in F^*$ proceeds as follows. To share a secret $s \in F$, sample a random F -polynomial $P(X)$ such that $\deg P < k$ and $P(0) = s$. Define the shares: $s_1 := P(\alpha_1)$, $s_2 := P(\alpha_2)$, $\dots$, and $s_n := P(\alpha_n)$. Denote this secret-sharing by $\text{ShamirSS}(n, k, \vec{\alpha})$ and the joint distribution of the shares by $\text{Share}(s)$ – other parameters will be clear from the context. *This work only considers $n = k$.*

Representing prime field elements. Consider a prime field F_p of order p , where $2^{\lambda-1} < p < 2^\lambda$ and λ is the security parameter. The elements of F_p are represented as λ -bit binary strings representing the elements $\{0, 1, \dots, (p-1)\}$.

► **Remark 3.** For a Fermat prime $p = 2^\lambda + 1$, elements of F_p require $(\lambda + 1)$ bits in their binary representation. However, only the binary representation of 2^λ has 1 in the most significant bit. For simplicity of presentation, we assume that elements are represented using λ bits only; disregarding the element $2^\lambda \in F_p$ adds only an additive $1/p$ slack to the analysis.

Leakage functions & families. This work studies *physical bit leakage* $\text{PHYS}_i: F_p \rightarrow \{0, 1\}$ that outputs the i -th least significant bit, where $i \in \{0, 1, \dots, \lambda - 1\}$. For example, PHYS_0 (also referred to as LSB) outputs 0 for the elements in $\{0, 2, \dots, (p - 1)\}$, where $p \geq 3$, and PHYS_1 outputs 0 for the elements in $\{0, 1, 4, 5, \dots\}$. For $\mathbf{i} = (i_1, i_2, \dots, i_n) \in \{0, 1, \dots, \lambda - 1\}^n$, the *leakage function* $\text{PHYS}_{\mathbf{i}}: F^n \rightarrow \{0, 1\}^n$ leaks the i_t -th bit of the t -th share, where $t \in \{1, 2, \dots, n\}$. For a secret $s \in F$, the joint distribution of the leakage is $\text{PHYS}_{\mathbf{i}}(\text{Share}(s))$. We consider two *leakage families*.

1. Physical bit leakage family: $\text{PHYS} := \left\{ \text{PHYS}_{\mathbf{i}} : \mathbf{i} = (i_1, \dots, i_n) \in \{0, 1, \dots, \lambda - 1\}^n \right\}$.
2. LSB leakage family: $\text{LSB} := \left\{ \text{PHYS}_{\mathbf{0}} \right\}$, where $\mathbf{0} = (0, 0, \dots, 0)$

Insecurity & randomized construction. Insecurity of $\text{ShamirSS}(n, k, \vec{\alpha})$ against a leakage family $\mathcal{F}$ is:

$$\varepsilon_{\mathcal{F}}(\vec{\alpha}) := \max_{f \in \mathcal{F}} \max_{s \in F^*} \text{SD}(f(\text{Share}(0)), f(\text{Share}(s))). \quad (1)$$

Low insecurity indicates the statistical independence of the leakage from the secret, i.e., the *secret-sharing is locally leakage-resilient* [4, 19]. Recently, Faust et al. [14] connected this definition to practice.

High insecurity indicates a leakage function can distinguish the secret 0 and some $s^* \in F^*$ using the leakage. Maji et al. [33] analyzed the insecurity against the PHYS leakage family when *evaluation places were chosen randomly*. Their result implies the following corollary for prime modulus $p \geq 3$ and $n = k \geq 2$.

For randomly chosen evaluation places $\vec{\alpha} \in (F_p^*)^n$, the insecurity $\varepsilon_{\text{PHYS}}(\vec{\alpha}) \leq p^{-1/2}$ with probability $\geq 1 - p^{-1/2}$.

Recently, [35] extended the randomized construction from prime fields to composite ones.

Our work investigates the security against the leakage family PHYS ; i.e., the adversary obtains arbitrary *one physical bit leakage from each share*. Our research question can be rewritten using these terminologies and notations as follows.

Our Research Question: Given evaluation places $\vec{\alpha}$ and prime modulus p , identify whether (1) $\varepsilon_{\text{PHYS}}(\vec{\alpha}) \leq p^{-1/2}$ or (2) $\varepsilon_{\text{PHYS}}(\vec{\alpha}) > p^{-1/2}$.

If $\varepsilon_{\text{PHYS}}(\vec{\alpha}) > p^{-1/2}$, then output a secret $s^* \in F_p^*$ such that the shares of 0 can be distinguished from the shares of s^* with (roughly) $\varepsilon_{\text{PHYS}}(\vec{\alpha})$ advantage. *All algorithms must be computationally efficient – runtime is a polynomial in λ ; i.e., $\text{poly}(\log p)$.* Furthermore, concrete security analysis (over asymptotic analysis) is prioritized.

1.2 Our Results

Below, for $x, y, z \in \mathbb{R}$, the expression $x = y \pm z$ is a concise representation for “ $x \in [y - z, y + z]$.” For example, “ x is close to y ” is expressed using $x = y \pm \varepsilon$, for a small ε . Section 2 presents a high-level overview of the critical technical ideas underlying our results.

Technical Result: Security against LSB Leakage when $n = 2$

Consider *arbitrary prime* $p \geq 3$ (not just a Mersenne/Fermat prime) and the LSB leakage. The technical workhorse for our results is the classifier for $(n, k) = (2, 2)$; other results bootstrap from it.

1. Figure 1 presents our efficient algorithm to classify $\vec{\alpha}$ as secure or not. If our algorithm classifies $\vec{\alpha}$ as secure, then Corollary 14 and Corollary 15 shows that

$$\varepsilon_{\text{LSB}}(\vec{\alpha}) \leq \frac{1 + 8^{5/4}}{\sqrt{p}} + \frac{13/2}{p} \leq \frac{14.46}{\sqrt{p}},$$

which is exponentially small in the security parameter λ . The number of false negatives is $\mathcal{O}(\sqrt{p} \cdot \log p)$.

2. We present an efficient adversary (Corollary 16) that generates $s^* \in F^*$ such that it distinguishes the secret 0 from s^* by leaking the LS of each share with an advantage

$$\geq \varepsilon_{\text{LSB}}(\vec{\alpha}) - \frac{2 \cdot 8^{5/4}}{\sqrt{p}} - \frac{13}{p} \geq \varepsilon_{\text{LSB}}(\vec{\alpha}) - \frac{26.91}{\sqrt{p}}.$$

Therefore, our efficient leakage attack achieves a comparable distinguishing advantage when the insecurity $\varepsilon_{\text{LSB}}(\vec{\alpha})$ is significant.

Result A: Security against Physical Bit Leakage when $n = 2$

For the $n = k = 2$ case, we analyze a prime field F_p , where p is a Mersenne/Fermat prime – primes of the form $2^\lambda \pm 1$. We reduce arbitrary physical bit leakage to LSB leakage for related evaluation places over these fields. In this context, our work proves the following results.

1. Figure 2 presents our efficient classifier against PHYS leakage. For $\vec{\alpha}$ that is classified secure, Corollary 19 shows that the insecurity is $\varepsilon_{\text{PHYS}}(\vec{\alpha}) \leq 14.46/\sqrt{p}$ and the number of false negatives is $\mathcal{O}(\sqrt{p} \cdot (\log p)^2)$.
2. We present an efficient adversary that generates $(s^*, f) \in F_p^* \times \text{PHYS}$ such that it distinguishes the secret 0 from secret s^* by leaking $f \in \text{PHYS}$ from the shares with an advantage $\geq \varepsilon_{\text{PHYS}}(\vec{\alpha}) - 26.91/\sqrt{p}$. *These are new side-channel attacks; their existence demonstrates the tightness of our analysis and accuracy of our classifier.*

This is direct consequence of the properties of Mersenne and Fermat primes and Corollary 16.

3. We explicitly identify secure evaluation places against PHYS leakage: all (α_1, α_2) satisfying $\alpha_2 \cdot \alpha_1^{-1} \in \{\gamma, -\gamma, \gamma^{-1}, -\gamma^{-1}\}$, where $\gamma = 2^{\lfloor \lambda/2 \rfloor} - 1$. For these evaluation places, we get $\varepsilon_{\text{PHYS}}(\vec{\alpha}) \leq 8.49/\sqrt{p}$, which Corollary 20 and Corollary 21 prove. We provide an example for Mersenne prime $p = 2^{13} - 1$ in the full version of our paper [27].

Result B: Security against Physical Bit Leakage when $n > 2$

Consider a prime field F_p such that $p = 2^\lambda \pm 1$ a Mersenne/Fermat prime. Figure 2 presents an efficient classifier for $\vec{\alpha}$ such that the corresponding ShamirSS($n, n, \vec{\alpha}$) is secure to physical bit probes; the insecurity is at most $1/\sqrt{p}$, as shown in Corollary 23.

Given evaluation places $\vec{\alpha} := (\alpha_1, \alpha_2, \dots, \alpha_n)$ we efficiently compute an appropriate $\vec{\beta} := (\beta_1, \beta_2, \dots, \beta_n)$ (see Equation 3). Corollary 23 proves that if ShamirSS($2, 2, (\beta_1, \beta_2)$) has ε -insecurity against physical bit leakage, then ShamirSS($n, n, \vec{\alpha}$) has 2ε -insecurity against physical bit leakage. Clarifications below highlight the subtlety of this classifier:

► Remark 4 (Clarifications).

1. High insecurity of $\text{ShamirSS}(2, 2, (\beta_1, \beta_2))$ does not imply high insecurity of $\text{ShamirSS}(n, n, \vec{\alpha})$; our result lifts security only in one direction.
2. Can the security of $\text{ShamirSS}(2, 2, (\alpha_i, \alpha_j))$, for all $1 \leq i < j \leq n$, imply the security of $\text{ShamirSS}(n, n, \vec{\alpha})$? This natural classifier has false positives. Consider $n = 3$, a prime $p = 4w^2 + 6w + 9$, and evaluation places $\vec{\alpha} = (1, \sigma, \sigma^2)$, where $w \geq 4$, $w \not\equiv 0 \pmod 3$, and $\sigma = 2w \cdot 3^{-1}$. For example, $p = 97$ and $\sigma = 35$; Bunyakovsky conjecture [7] implies infinitude of such primes. Against LSB leakage, although every $\text{ShamirSS}(2, 2, (\alpha_i, \alpha_j))$ is secure, $\text{ShamirSS}(n, n, \vec{\alpha})$ is $(2/\pi)^3 > 0.25$ insecure [34, 14]; Appendix C presents the details.

So, the following randomized strategy suffices to construct secure schemes: (1) randomly sample $\vec{\alpha}$, (2) compute $\vec{\beta}$ using our map in Figure 3, and (3) test the security of secret sharing scheme $\text{ShamirSS}(2, 2, (\beta_1, \beta_2))$ using Corollary 19.

We can obtain secure evaluation places for $n = k > 2$ case by bootstrapping from the explicit secure evaluation places for $n = k = 2$ case. For example, $\alpha_1 = (n - 1)$, $\alpha_2 = (n - 1) - (1 + \gamma)$, and $\alpha_j = (j - 2) \cdot (\gamma + 1)$, for $j \in \{3, 4, \dots, n\}$, is secure against one physical bit probe per share if $(1, \gamma)$ is secure evaluation place for $n = k = 2$ case. Specifically, $\gamma = \sqrt{(p \pm 1)/2} - 1$ suffices for Mersenne/Fermat prime modulus.

2 Technical Overview

2.1 Technical Result: LSB Leakage $(n, k) = (2, 2)$

For any prime field F_p , we outline our classification algorithm for $(n, k) = (2, 2)$ and, en route, highlight our technical contributions (Figure 1 presents the pseudocode).

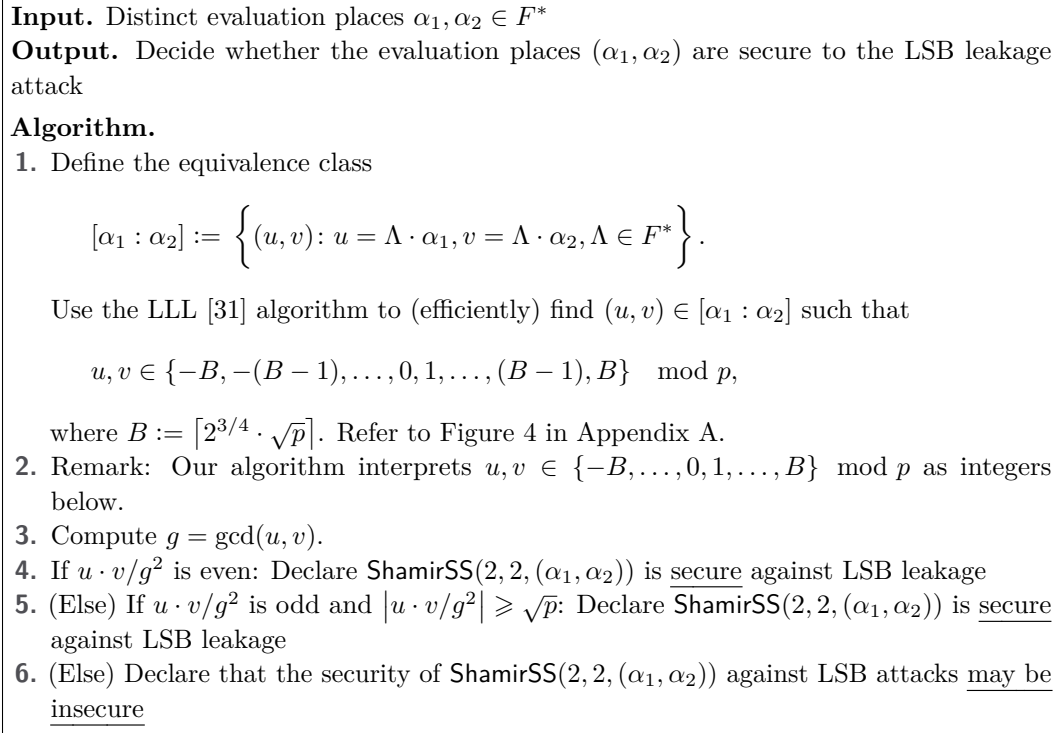
Step 1. The prime modulus p and the distinct evaluation places $\alpha_1, \alpha_2 \in F_p^*$ are inputs to the LSB classification algorithm. The security/vulnerability of evaluation places (α_1, α_2) is identical to any evaluation places (u, v) satisfying $\alpha_1 \cdot \alpha_2^{-1} = u \cdot v^{-1}$ (follows from Generalized Reed-Solomon codes' properties [23]). We find “small norm” $u, v \in \{-\lceil \sqrt{p} \rceil, \dots, 0, 1, \dots, \lceil \sqrt{p} \rceil\}$ with the property mentioned above – a Dirichlet approximation problem. We solve it with a small constant multiplicative slack using the LLL [32] algorithm in $\text{poly}(\lambda)$ runtime, where $\lambda := \lceil \log_2(p + 1) \rceil$. The reasoning for choosing “small norm” u, v will be evident below in Step 3.

Step 2. We proceed to solve the technical question from Page 2: determine whether the bits $\text{LSB}(u \cdot X)$ and $\text{LSB}(v \cdot X)$ are statistically independent, for uniformly random $X \in F_p$.

We will calculate the similarity/dependence between these two distributions, which is equivalent to the bias ε of the distribution $\text{LSB}(u \cdot X) \oplus \text{LSB}(v \cdot X)$. In this context, the bias is the probability that $\text{LSB}(u \cdot X) = \text{LSB}(v \cdot X)$ minus the probability that $\text{LSB}(u \cdot X) \neq \text{LSB}(v \cdot X)$.

Step 3. To develop an *efficient* algorithm to compute ε , we express the quantity ε as the inner product of two oscillatory $\{\pm 1\}$ sequences, approximated by the following integral.

$$\int_0^1 \text{sign} \sin(2\pi|u| \cdot t) \cdot \text{sign} \sin(2\pi|v| \cdot t) \, dt.$$



■ **Figure 1** Identify secure evaluation places for Shamir's secret sharing against LSB leakage.

Here, $\text{sign} \sin(2\pi|u| \cdot t) \in \{\pm 1\}$ is a *square wave* that oscillates $|u|$ times in the domain $[0, 1]$. The integral above measures the similarity/dependence between the two square waves, the first oscillating $|u|$ times and the second oscillating $|v|$ times. The error of our approximation is directly proportional to the total number of oscillations of the square waves. The approximation error is $\leq (|u| + |v|)/p = \mathcal{O}(1/\sqrt{p})$, exponentially small in λ , for small norm u, v . For simplicity, the presentation below ignores this approximation error.

Step 4. Finally, we present a closed-form expression for the integral; thus computing the bias ε . For $g := \gcd(|u|, |v|)$ and $\rho := |u| \cdot |v| / g^2$, we prove that:

$$\varepsilon = \begin{cases} 0, & \text{if } \rho \text{ is even} \\ 1/2\rho, & \text{if } \rho \text{ is odd.} \end{cases}$$

Step 5. Consider the $\varepsilon = 0$ case. This happens when the highest powers of 2 dividing $|u|$ and $|v|$ differ. In this case, we prove that $\text{LSB}(u \cdot X + s)$ is independent of $\text{LSB}(v \cdot X + s)$, for every secret $s \in F_p$. Technically, we prove the following integral representing the bias for this general case – a phase-shifted integral from Step 2 above – is 0 for all $\delta \in [0, 1)$.

$$\int_0^1 \text{sign} \sin(2\pi|u| \cdot t) \cdot \text{sign} \sin(2\pi|v| \cdot t + 2\pi \cdot \delta) \, dt.$$

Note that the marginal $\text{LSB}(u \cdot X + s)$ is a uniformly random bit, and so is the marginal $\text{LSB}(v \cdot X + s)$. Therefore, these leakage bits are uniformly and independently random. Furthermore, the distribution of $(u \cdot X + s, v \cdot X + s)$, for uniformly random $X \in F_p$, is identical to the distribution of the shares $(s_1, s_2) = (\alpha_1 \cdot X + s, \alpha_2 \cdot X + s)$ by properties of General Reed-Solomon codes [23]. Consequently, Shamir's scheme is secure in this case because all secrets produce identical leakage distribution.

When $\varepsilon \neq 0$, $|u|$ and $|v|$ have the identical highest power of 2 dividing them. Theorem 10 presents a (closed-form) expression for a secret $s^* \in F_p^*$ such that the distributions of LSB leakage for secret 0 and secret s^* are distinguishable with an advantage of ε . We achieve this by giving the formula for the $\delta^* \in \{1/p, 2/p, \dots, (p-1)/p\}$, such that the following integral's value is farthest from ε .

$$\int_0^1 \text{sign} \sin(2\pi|u| \cdot t) \cdot \text{sign} \sin(2\pi|v| \cdot t + 2\pi \cdot \delta^*) \, dt$$

We then reconstruct s^* from this δ^* .

Our classification algorithm for arbitrary physical bit probes will build on the classifier outlined in this section.

► **Remark 5.** Our work connects the security of secret-sharing schemes against leakage attacks with the orthogonality properties of a family of square waves [42, 25, 24]. Various families of square waves, like the ones by Haar [22], Walsh [44], and Rademacher [39], are central to science and engineering. These techniques are new to the security analysis of secret sharings and possibly of broader interest.

2.2 Overview of Result A: Physical Bit Leakage $(n, k) = (2, 2)$

Suppose the evaluation places are $\vec{\alpha} = (\alpha_1, \alpha_2)$. We aim to *determine whether Shamir's secret-sharing scheme with these evaluation places is secure against all physical bit leakage attacks in Mersenne prime fields*. For $i, j \in \{0, 1, \dots, \lambda - 1\}$, consider the physical bit leakage attack $\text{PHYS}_{i,j}$. This leakage attack leaks the i -th LSB of the share s_1 and the j -th LSB of the share s_2 . For a Mersenne prime p and an element $x \in F_p$, the binary representation of $x \cdot 2^{-1}$ is the *right rotation* of the binary representation of x by one position. Therefore, $\text{PHYS}_{i,j}$ leakage with evaluation places (α_1, α_2) is identical to the LSB leakage with evaluation places $(2^{-i} \cdot \alpha_1, 2^{-j} \cdot \alpha_2)$. By Generalized Reed-Solomon codes' properties [23], the leakage is identical to the LSB leakage with evaluation places $(2^{j-i} \cdot \alpha_1, \alpha_2)$. Consequently,

$$\varepsilon_{\text{PHYS}}((\alpha_1, \alpha_2)) = \max \{ \varepsilon_{\text{LSB}}(\alpha_1, \alpha_2), \varepsilon_{\text{PHYS}}(2\alpha_1, \alpha_2), \dots, \varepsilon_{\text{PHYS}}(2^{\lambda-1}\alpha_1, \alpha_2) \}.$$

Thus, security against PHYS leakage reduces to a sequence of LSB security estimations. Figure 2 presents this pseudo-code.

► **Remark 6 (An Edge Case).** The algorithm determining the security of Shamir's secret-sharing scheme to LSB attack requires the evaluation places to be distinct. Even though α_1 and α_2 are distinct, it may be the case that $2^t \alpha_1 = \alpha_2$, for some $t \in \{0, 1, \dots, \lambda - 1\}$. So, the call to the “LSB security check subroutine” with the argument $(2^t \alpha_1, \alpha_2)$ would be invalid. Lemma 18 proves that this edge case is insecure. This case captures why evaluation places $(1, 2)$ are insecure against physical bit leakage.

For *Fermat prime* p , $x \in F_p$, and $i \in \{1, 2, \dots, \lambda - 1\}$ we prove following identity.¹

$$\text{PHYS}_{i-1}(x) = \text{PHYS}_i(2x + 1). \quad (2)$$

Therefore, $\text{PHYS}_i(x) = \text{LSB}(2^{-i} \cdot x + 2^{-i} - 1)$. Like the Mersenne prime case above, arbitrary physical bit leakage translates into LSB leakage, except the map here is an *affine map* instead of a *linear map*. As a result, the secret $s^* \in F_p^*$ witnessing the maximum insecurity is different; it is still efficiently computable. See Section 5.1 for details.

► **Remark 7.** Investigating Mersenne prime modulus in the context of Shamir secret-sharing has also been done by Faust et al. [14]; the ideas to analyze Fermat prime modulus are new.

¹ For primes other than Mersenne and Fermat primes, there is no such affine transformation.

Input. Distinct evaluation places $\alpha_1, \alpha_2 \in F_p^*$, and p is a Mersenne/Fermat prime.

Output. Decide whether the evaluation places (α_1, α_2) are secure to an arbitrary single physical bit leakage per share.

Algorithm.

1. If there is $t \in \{0, 1, \dots, \lambda - 1\}$ such that $2^t \alpha_1 = \alpha_2$: Return insecure
2. For $t \in \{0, 1, \dots, \lambda - 1\}$:
 - a. Call the algorithm in Figure 1 with evaluation places $(2^t \alpha_1, \alpha_2)$
 - b. If the algorithm returns “may be insecure,” return may be insecure
3. Declare $\text{ShamirSS}(2, 2, (\alpha_1, \alpha_2))$ is secure against physical bit attacks.

■ **Figure 2** Identify secure evaluation places for Shamir’s secret sharing against physical bit leakage.

2.3 Overview of Result B: Physical Bit Leakage $n = k > 2$

Our objective is to choose n distinct evaluation places $\alpha_1, \alpha_2, \dots, \alpha_n \in F^*$ such that the corresponding $\text{ShamirSS}(n, n, \vec{\alpha})$ is secure against physical bit leakage attacks. We prove a lifting theorem (Theorem 22) that proves the following result. Given evaluation places $\vec{\alpha}$, consider $\vec{\beta}$ related to Lagrange multipliers (where $i \in \{1, 2, \dots, n\}$):

$$\beta_i := \left(\alpha_i \prod_{j \neq i} (\alpha_i - \alpha_j) \right)^{-1}. \quad (3)$$

Now consider the $\text{ShamirSS}(2, 2, (\beta_i, \beta_j))$ secret-sharing scheme for all distinct $i, j \in \{1, \dots, n\}$. Suppose one of these secret-sharing schemes is secure against physical bit leakage. In that case, the $\text{ShamirSS}(n, n, \vec{\alpha})$ secret-sharing scheme is also secure. More concretely, if the insecurity of $\text{ShamirSS}(2, 2, (\beta_i, \beta_j))$ is ε , for some distinct $i, j \in \{1, 2, \dots, n\}$, then the $\text{ShamirSS}(n, n, \vec{\alpha})$ secret-sharing scheme is (at most) 2ε insecure.

Whether the evaluation places of $\text{ShamirSS}(2, 2, (\beta_i, \beta_j))$ is secure or not can be determined efficiently using our algorithm in Figure 2. We can use this algorithm to detect if our chosen $\vec{\alpha}$ has such a secure (β_i, β_j) pair of evaluation places. Corollary 23 formally states this result; its proof is entirely Fourier-analytic.

► **Remark 8.** Analyzing this classifier has some subtleties. The $\vec{\alpha} \mapsto \vec{\beta}$ mapping is not a bijection; few $\vec{\beta}$ have multiple preimages, most have one, and some have none. We prove that (β_1, β_2) are (nearly) independent when $\vec{\alpha}$ is chosen uniformly at random, for $n \geq 3$.

► **Remark 9 (Clarifications).**

1. High insecurity of $\text{ShamirSS}(2, 2, (\beta_1, \beta_2))$ *does not imply* high insecurity of the corresponding $\text{ShamirSS}(n, n, \vec{\alpha})$; *our result lifts security only in one direction.*
2. Can the security of $\text{ShamirSS}(2, 2, (\alpha_i, \alpha_j))$, for all $1 \leq i < j \leq n$, imply the security of $\text{ShamirSS}(n, n, \vec{\alpha})$? *This natural classifier has false positives.* Consider $n = 3$, a prime $p = 4w^2 + 6w + 9$, and evaluation places $\vec{\alpha} = (1, \sigma, \sigma^2)$, where $w \geq 4$, $w \not\equiv 0 \pmod{3}$, and $\sigma = 2w \cdot 3^{-1}$. For example, $p = 97$ and $\sigma = 35$; Bunyakovsky conjecture [7] implies infinitude of such primes. Against LSB leakage, although every $\text{ShamirSS}(2, 2, (\alpha_i, \alpha_j))$ is secure, $\text{ShamirSS}(n, n, \vec{\alpha})$ is $(2/\pi)^3 > 0.25$ insecure [34, 14]; Appendix C presents the details.

Input. Distinct evaluation places $\vec{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_n) \in (F_p^*)^n$, and p is a Mersenne or Fermat prime

Output. Decide whether the evaluation places $\vec{\alpha}$ are secure to all physical bit leakage attacks

Algorithm.

1. For $i \in \{1, 2, \dots, n\}$, compute

$$\beta_i := \left(\alpha_i \prod_{j \neq i} (\alpha_i - \alpha_j) \right)^{-1}.$$

2. If there exist $1 \leq i < j \leq n$ such that $\text{ShamirSS}(2, 2, (\beta_i, \beta_j))$ is secure per the algorithm in Figure 2, then declare that $\text{ShamirSS}(n, n, \vec{\alpha})$ is secure.
3. Otherwise, the algorithm states that $\text{ShamirSS}(n, n, \vec{\alpha})$ may be insecure.

■ **Figure 3** Identify secure evaluation places for $\text{ShamirSS}(n, n)$ against physical bit leakage.

3 Future Research Directions

There are several natural research directions for future work. A few immediate ones and their respective technical hurdles are presented below.

LSB classifier construction for $n > 2$. To illustrate the challenges, consider $n = 3$ and evaluation places $(\alpha_1, \alpha_2, \alpha_3)$. The rational approximation problem will require finding small-norm u, v, w such that $\alpha_1 : \alpha_2 : \alpha_3 = u : v : w$. Dirichlet approximation theorem only guarantees $|u|, |v|, |w| \leq p^{(n-1)/n}$. Therefore, *the accuracy error in estimating the summation by an integral will be $p^{(n-1)/n}/p = p^{-1/n} \gg p^{-1/2}$, for $n \geq 3$.*

Moreover, for $\varphi(x) = \text{sign} \sin(2\pi x)$, *the estimate of the integral below is not known.*

$$\int_0^1 \varphi(ut) \cdot \varphi(vt) \cdot \varphi(wt) dt. \quad (4)$$

Arbitrary physical bit leakage in general prime modulus. Against arbitrary physical bit leakage, extension to general prime modulus seems challenging. For example, when $n = 2$, the technical challenge is to characterize (α_1, α_2) such that the distributions $\text{PHYS}_i(\alpha_1 X)$ is independent of $\text{PHYS}_j(\alpha_2 X)$, where X is chosen uniformly at random. The bottleneck is to establish an integral that estimates this expression for a *general prime modulus*.

More physical probes. Consider $(n, k) = (2, 2)$, evaluation places (α_1, α_2) , a Mersenne prime modulus p , and physical bit leakage probing the first share twice & the second share once. The technical problem is to show the independence of the following three distributions

$$\left(\text{PHYS}_i(\alpha_1 X), \text{PHYS}_j(\alpha_1 X), \text{PHYS}_k(\alpha_2 X) \right),$$

where $X \in F_p$ is chosen uniformly at random. The analysis *reduces to estimating the integral in Equation 4*, where $u = 2^{-i}\alpha_1$, $v = 2^{-j}\alpha_1$, and $w = 2^{-k}\alpha_2$, which is not known.

More general (n, k) . For concreteness, consider $(n, k) = (3, 2)$ and resilience to LSB leakage. This resilience requires t -wise independence of the leakage bits, where $k \leq t \leq n$. The 2-wise independence of leakage bits can be tested using the classifier in Figure 1. The 3-wise

independence test has identical hurdles as the “LSB classifier construction for $n > 2$ ” case discussed above. There are evaluation places where *the LSB leakage is 2-wise independent but 3-wise correlated* for $(n, k) = (3, 2)$. The evaluation places of Appendix C with $(n, k) = (3, 2)$ (instead of $(n, k) = (3, 3)$) have this property.

4 Security against Least Significant Bit Leakage

This section presents our results regarding the security of Shamir’s secret-sharing scheme when $n = k = 2$ against the LSB leakage. We begin with a powerful technical result.

► **Theorem 10** (Technical Result). *Consider the ShamirSS(2, 2, (α_1, α_2)) secret-sharing scheme over a prime field F_p , where $p \geq 3$.*

$$\begin{aligned} & \max_{s \in F} \text{SD} \left(\text{LSB}(\text{Share}(0)), \text{LSB}(\text{Share}(s)) \right) \\ &= \begin{cases} \frac{4(|u| + |v|) - (3/2)}{p}, & \text{if } |u| \cdot |v|/g^2 \text{ is even,} \\ \left(2 - \frac{1}{2p}\right) \cdot \frac{g^2}{|u| \cdot |v|} \pm \frac{4(|u| + |v|) - (3/2)}{p} & \text{if } |u| \cdot |v|/g^2 \text{ is odd,} \end{cases} \end{aligned}$$

where $\alpha_1 \cdot \alpha_2^{-1} = u \cdot v^{-1}$ and $g = \gcd(|u|, |v|)$.

Furthermore, for $s \in F_p^*$ satisfying $(2^{-1} \cdot s) \cdot (u^{-1} - v^{-1}) \in \frac{(2\mathbb{Z}+1) \cdot p \pm 1}{2|u||v|}$, if

$$\text{SD} \left(\text{LSB}(\text{Share}(0)), \text{LSB}(\text{Share}(s)) \right) > \frac{4(|u| + |v|) - (3/2)}{p}$$

then there is an efficient distinguisher to distinguish the secret 0 and s with advantage at least

$$\left(2 - \frac{1}{2p}\right) \cdot \frac{g^2}{|u| \cdot |v|} - \frac{4(|u| + |v|) - (3/2)}{p}$$

using the LSB leakage on the secret shares.

Essentially, this theorem helps estimate the insecurity efficiently. Section 4.1 presents the proof outline for this result and we presents the complete proof in the full version of our paper [27]. With this theorem, we will state and prove the corollaries mentioned in Section 1.2.

4.1 Proof outline of Theorem 10

For any $s \in F^*$, we start by obtaining a closed-form estimate of

$$\text{SD} \left(\text{LSB}(\text{Share}(0)), \text{LSB}(\text{Share}(s)) \right).$$

Then, we can solve for the optimal $s \in F^*$ that maximizes the statistical distance. Below, we present a high-level overview of the proof of Theorem 10.

Step 1. We connect the statistical distance between the leakages to the difference between two sums of oscillatory functions. We define the function $\text{sign}_p: \mathbb{Z} \rightarrow \pm 1$.

$$\text{sign}_p(X) := \begin{cases} +1, & \text{if } X \in \{0, 1, \dots, (p-1)/2\} \pmod{p} \\ -1, & \text{if } X \in \{-(p-1)/2, \dots, -1\} \pmod{p}. \end{cases}$$

3:12 Leakage-Resilience of Shamir's Secret Sharing

For $u, v, \Delta \in F$, we define the following measurement of similarity between two lines uT and $v(T - \Delta)$ on F .

$$\Sigma_{u,v}^{(\Delta)} := \sum_{T \in F} \text{sign}_p(uT) \cdot \text{sign}_p(v(T - \Delta)). \quad (5)$$

► **Lemma 11.** *Consider the ShamirSS(2, 2, (α_1, α_2)) secret-sharing scheme over a prime field F_p . For any secret $s \in F_p$ and $(u, v) \in [\alpha_1 : \alpha_2]$,*

$$\text{SD} \left(\text{L}\vec{\text{S}}\text{B}(\text{Share}(0)), \text{L}\vec{\text{S}}\text{B}(\text{Share}(s)) \right) = \frac{1}{2p} \cdot \left| \Sigma_{u,v}^{(0)} - \Sigma_{u,v}^{(\Delta)} \right|,$$

where $\Delta := (s \cdot 2^{-1}) \cdot (u^{-1} - v^{-1})$, a linear automorphism over F_p .

Step 2. Next, our objective is to estimate the sum $\frac{1}{p} \cdot \Sigma_{u,v}^{(\Delta)}$ using the integral $I_{u,v}^{(\delta)}$ defined as an inner product of two square wave functions as follow.

$$I_{u,v}^{(\delta)} := \int_0^1 \text{sign} \sin(2\pi|u| \cdot t) \cdot \text{sign} \sin(2\pi|v| \cdot (t - \delta)) \, dt.$$

► **Lemma 12.** *For any $u, v, \Delta \in F_p$, and $\delta = \frac{\text{sign}_p(\Delta) \cdot |\Delta|}{p} \in \mathbb{Q}$,*

$$\frac{1}{p} \cdot \Sigma_{u,v}^{(\Delta)} = \text{sign}_p(u) \cdot \text{sign}_p(v) \cdot I_{|u|,|v|}^{(\delta)} + \frac{\text{sign}_p(u\Delta) - \text{sign}_p(v\Delta)}{p} \pm \frac{4(|u| + |v|) - 2}{p}.$$

Step 3. Finally, we compute the value of the integral $I_{u,v}^{(\delta)}$.

► **Lemma 13.** *Let $\Delta: \mathbb{R} \rightarrow [-1, +1]$ be the triangle wave function defined as*

$$\Delta(t) := 4 \cdot \left| t + \frac{1}{2} - \lceil t \rceil \right| - 1.$$

Then, for any $u, v \in \{1, 2, \dots\}$, $\delta \in \mathbb{R}$, and $g = \gcd(u, v)$

$$I_{u,v}^{(\delta)} = \begin{cases} 0, & \text{if } u \cdot v / g^2 \text{ is even} \\ \Delta(uv \cdot \delta) \cdot \frac{g^2}{uv}, & \text{if } u \cdot v / g^2 \text{ is odd.} \end{cases}$$

Intuitively, if the highest power of 2 dividing u is different from the highest power of 2 dividing v , then uv/g^2 is even and $I_{u,v}^{(\delta)} = 0$. If the highest power of 2 dividing u is identical to the highest power of 2 dividing v , then uv/g^2 is odd and $I_{u,v}^{(\delta)} \neq 0$.

Step 4. Sequentially performing the substitutions above, we can estimate the statistical distance using the integrals, which yields Theorem 10 after maximizing over every $s \in F^*$.

Efficient distinguisher construction. We present an efficient maximum likelihood distinguisher in Appendix B.

4.2 Insecurity Estimation: Statement and proof of Corollary 14

Using Theorem 10, we prove that the estimated insecurity achieved by our classifier in Figure 1 is close to the true insecurity.

► **Corollary 14.** *Consider distinct evaluation places $\vec{\alpha} = (\alpha_1, \alpha_2)$ and the corresponding ShamirSS(2, 2, $\vec{\alpha}$) secret-sharing scheme over the prime field F_p , where $p \geq 3$. Let $(u, v) \in [\alpha_1 : \alpha_2]$ such that $|u|, |v| \leq B$, where $B = \lceil 8^{1/4} \cdot \sqrt{p} \rceil$. Let $\Delta: \mathbb{R} \rightarrow [-1, +1]$ be the triangle wave function $\Delta(t) := 4 \cdot |t + \frac{1}{2} - \lceil t \rceil| - 1$. Let $g = \gcd(|u|, |v|)$. Define*

$$\varepsilon_{\text{LSB}}^{(\text{est})}(\vec{\alpha}) := \begin{cases} 0, & \text{if } |u| \cdot |v|/g^2 \text{ is even,} \\ \Delta(|u||v| \cdot \delta) \cdot \frac{g^2}{|u| \cdot |v|}, & \text{if } |u| \cdot |v|/g^2 \text{ is odd.} \end{cases}$$

Then,

$$\varepsilon_{\text{LSB}}^{(\text{est})}(\vec{\alpha}) = \varepsilon_{\text{LSB}}(\vec{\alpha}) \pm \left(\frac{8^{5/4}}{\sqrt{p}} + \frac{13/2}{p} \right).$$

Proof. Use the LLL algorithm [32] to efficiently find $(u, v) \in [\alpha_1 : \alpha_2]$ with properties mentioned in the corollary (see Appendix A for details). Observe that the LHS of the expression in Theorem 10 is identical to $\varepsilon_{\text{LSB}}(\vec{\alpha})$ by our definition in Equation 1. From this observation, the corollary is immediate. ◀

Next, we state the corollaries mentioned in Section 1.2 through this tight estimation.

4.3 Insecurity Identification: Statement of Corollary 15

► **Corollary 15.** *Consider distinct evaluation places $\vec{\alpha} = (\alpha_1, \alpha_2)$ and the corresponding ShamirSS(2, 2, $\vec{\alpha}$) secret-sharing scheme over the prime field F_p , where $p \geq 3$. Suppose the algorithm in Figure 1 determines $\vec{\alpha}$ to be secure. Then,*

$$\varepsilon_{\text{LSB}}(\vec{\alpha}) \leq \frac{1 + 8^{5/4}}{\sqrt{p}} + \frac{13/2}{p}.$$

Among all possible distinct evaluation places $\alpha_1, \alpha_2 \in F_p^*$, the algorithm of Figure 1 determines at least

$$\geq 1 - \frac{\frac{1}{4} \cdot \sqrt{p} \cdot \ln p + \frac{3}{2} \cdot \sqrt{p} + \frac{1}{2}}{p - 2} \geq^{(*)} 1 - \left(\frac{\ln p}{4\sqrt{p}} + \frac{5/2}{\sqrt{p}} \right).$$

fraction of them to be secure. The (*) inequality holds for any prime $p \geq 11$.

4.4 Advantage of Adversary: Statement of Corollary 16

► **Corollary 16.** *Consider distinct evaluation places $\vec{\alpha} = (\alpha_1, \alpha_2)$ and the corresponding ShamirSS(2, 2, $\vec{\alpha}$) secret-sharing scheme over the prime field F_p , where $p \geq 3$. If $\varepsilon_{\text{LSB}}(\vec{\alpha}) > \frac{2 \cdot 8^{5/4}}{\sqrt{p}} + \frac{13}{p}$, then there is an efficient algorithm that generates $s \in F_p^*$ and can distinguish the secret 0 from the secret s with an advantage*

$$\geq \varepsilon_{\text{LSB}}(\vec{\alpha}) - \frac{2 \cdot 8^{5/4}}{\sqrt{p}} - \frac{13}{p}$$

by leaking the LSB of the secret shares.

Consider an efficient adversary outputs the s indicated in Theorem 10. After observing the leakage (ℓ_1, ℓ_2) , the algorithm performs maximum likelihood decoding – computes whether secret 0 or secret s is more likely to have generated the observed leakage. Then, it predicts the most likely of the two events.

5 Security against all Physical Bit Leakage

We consider $\text{ShamirSS}(n = 2, k = 2, (\alpha_1, \alpha_2))$ over prime field F_p of order $p \geq 3$. Let λ be the security parameter. This section considers Mersenne and Fermat primes, i.e., primes of the form $p = 2^\lambda \pm 1$. Some initial Mersenne primes are 3, 7, 31, 127, 8191, and 131071, and Fermat primes are 3, 5, 17, 257, and 65537. Mersenne and Fermat's primes satisfy the following property.

► **Proposition 17.** *Let λ be the security parameter. Fix an arbitrary $i \in \{0, 1, 2, \dots, \lambda - 1\}$. For all $x \in F_p$,*

$$\text{PHYS}_i(x) = \begin{cases} \text{PHYS}_0(2^{-i} \cdot x) & \text{if } p \equiv -1 \pmod{2^{i+1}} \\ \text{PHYS}_0(2^{-i} \cdot x + (2^{-i} - 1)) & \text{if } p \equiv 1 \pmod{2^{i+1}} \end{cases}$$

5.1 Leakage attack when $2^k \alpha_1 = \alpha_2$

Although $\alpha_1 \neq \alpha_2$, it may be possible that $2^k \alpha_1 = \alpha_2$, for some $k \in \{0, 1, \dots, \lambda - 1\}$. We prove that the secret-sharing scheme is insecure, taking care of this case in the algorithm of Figure 2. Suppose we are leaking the i -th bit of the first secret share and the j -th bit of the second secret share, such that $j - i = k$.

Suppose the secret is $s \in F_p$. Then, the secret share at evaluation place α is $s + u\alpha$, for uniformly random $u \in F$. The joint distribution of leakage is

$$(\text{PHYS}_i(s + u\alpha_1), \text{PHYS}_j(s + u\alpha_2)).$$

Let $v := u2^{-j}$ and $t := s2^{-j}$. When the order of the field is a Mersenne or Fermat's prime, Proposition 17 implies that the joint distribution of leakage is equivalent as (for uniformly random $v \in F$)

$$(\text{PHYS}_0(t2^k + v\alpha_1 2^k), \text{PHYS}_0(t + v\alpha_2)) \equiv (\text{PHYS}_0(t2^k + v\alpha_2), \text{PHYS}_0(t + v\alpha_2)),$$

because $2^k \alpha_1 = \alpha_2$. When $t = 0$, both the leakage bits are identical. On the other hand, for $t = t^* := (2^k - 1)^{-1}$, the joint distribution of leakage is

$$(\text{PHYS}_0(1 + t^* + v\alpha_1), \text{PHYS}_0(t^* + v\alpha_2))$$

These two leakage bits are different with $(1 - 1/p)$ probability. Therefore, one can distinguish the secret 0 and secret t^*2^j with $(1 - 1/p) \sim 1$ advantage by leaking $\text{PHYS}_{i,j}$; whence the following lemma.

► **Lemma 18.** *Let F be the prime field of order $p = 2^\lambda \pm 1$. Consider distinct evaluation places $\alpha_1, \alpha_2 \in F^*$ such that $2^k \cdot \alpha_1 = \alpha_2$ for some $k \in \{0, 1, \dots, \lambda - 1\}$. Then,*

$$\text{SD} \left(\text{PHYS}_{i,j}(\text{Share}(0)), \text{PHYS}_{i,j}(\text{Share}(s)) \right) \geq 1 - \frac{1}{p},$$

where $i, j \in \{0, 1, \dots, \lambda - 1\}$, $j - i = k \pmod{\lambda}$. If $p = 2^\lambda - 1$, $s = (2^k - 1)^{-1} \cdot 2^j$ and if $p = 2^\lambda + 1$, $s = (2^k - 1)^{-1} \cdot 2^j - 1$.

5.2 Upper Bound on insecurity

► **Corollary 19.** *Let F be the prime field of order $p = 2^\lambda \pm 1$. Consider distinct evaluation places $\vec{\alpha} = (\alpha_1, \alpha_2)$ and the corresponding ShamirSS(2, 2, $\vec{\alpha}$) secret-sharing scheme over the prime field F . Suppose the algorithm in Figure 2 determines $\vec{\alpha}$ to be secure. Then,*

$$\varepsilon_{\text{PHYS}}(\vec{\alpha}) \leq \frac{1 + 8^{5/4}}{\sqrt{p}} + \frac{13/2}{p}.$$

Among all possible distinct evaluation places $\alpha_1, \alpha_2 \in F^*$, the algorithm of Figure 1 determines at least

$$\geq 1 - \frac{\ln p}{\ln 2} \cdot \frac{\frac{1}{4} \cdot \sqrt{p} \cdot \ln p + \frac{3}{2} \cdot \sqrt{p} + \frac{1}{2}}{p - 2} \geq^{(*)} 1 - \frac{\ln p}{\ln 2} \cdot \left(\frac{\ln p}{4\sqrt{p}} + \frac{5/2}{\sqrt{p}} \right).$$

fraction of them to be secure. The () inequality holds for all $p \geq 11$.*

5.3 Derandomization

We conclude this section by presenting a “derandomization” result that is a direct consequence of Theorem 10.

► **Corollary 20.** *Let F be the prime field of Mersenne prime order $p = 2^\lambda - 1$ where $\lambda > 3$. Define $t := \lfloor \lambda/2 \rfloor$. Consider $\vec{\alpha} = (\alpha_1, \alpha_2) \in [1 : 2^t - 1]$ respectively. Then*

$$\varepsilon_{\text{PHYS}}(\vec{\alpha}) \leq \frac{4 \cdot (2^{\lfloor \lambda/2 \rfloor} + 2^{\lceil \lambda/2 \rceil}) - 6}{p}.$$

A similar result holds for Fermat primes as well. Note that if $p = 2^\lambda + 1$ is a prime, then $\lambda/2$ is an integer because λ must be a power of 2.

► **Corollary 21.** *Let F be the prime field of Fermat prime order $p = 2^\lambda + 1$. Define $t := \lambda/2$. Consider $\vec{\alpha} = (\alpha_1, \alpha_2) \in [1 : 2^t - 1]$ respectively. Then*

$$\varepsilon_{\text{PHYS}}(\vec{\alpha}) \leq \frac{8 \cdot 2^{\lambda/2} - 3/2}{p}.$$

6 Extension to arbitrary Number of Parties

We extend our derandomization results to Shamir’s secret-sharing scheme with the reconstruction threshold k equal to the number of parties $n \in \{2, 3, \dots\}$. We begin by stating the following general lifting theorem.

► **Theorem 22.** *Consider ShamirSS($n, n, \vec{\alpha}$) over a prime field F . For every $i \in \{1, 2, \dots, n\}$, define*

$$\beta_i := \left(\alpha_i \prod_{j \neq i} (\alpha_i - \alpha_j) \right)^{-1}.$$

Suppose there are two indices $1 \leq i^* < j^* \leq n$ such that ShamirSS(2, 2, $(\beta_{i^*}, \beta_{j^*})$) has ε -insecurity against physical bit leakages. Then, ShamirSS($n, n, (\alpha_1, \alpha_2, \dots, \alpha_n)$) has at most 2ε -insecurity against physical bit leakages.

The proof of this theorem is Fourier-analytic and uses properties of the Generalized Reed-Solomon (GRS) codes. Corollary 23 is a consequence of this theorem.

► **Corollary 23.** *Let F_p be the prime field of order $p = 2^\lambda \pm 1$ and $n \in \{2, 3, \dots\}$. Consider distinct evaluation places $\vec{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_n)$ and the corresponding ShamirSS($n, n, \vec{\alpha}$) secret-sharing scheme over the prime field F_p . Suppose the algorithm in Figure 3 determines $\vec{\alpha}$ to be secure. Then,*

$$\varepsilon_{\text{PHYS}}(\vec{\alpha}) \leq \frac{2 \cdot 8^{5/4}}{\sqrt{p}} + \frac{13}{p}.$$

Among all possible distinct evaluation places $\vec{\alpha} \in (F_p^*)^n$, the algorithm of Figure 3 determines at least

$$1 - \left(\frac{1}{4 \ln 2} \cdot \frac{(\ln p)^2 \sqrt{p}}{p - n} + \frac{5}{2 \ln 2} \cdot \frac{(\ln p) \sqrt{p}}{p - n} \right)$$

fraction of them to be secure.

References

- 1 Donald Q. Adams, Hemanta K. Maji, Hai H. Nguyen, Minh L. Nguyen, Anat Paskin-Cherniavsky, Tom Suad, and Mingyuan Wang. Lower bounds for leakage-resilient secret sharing schemes against probing attacks. In *IEEE International Symposium on Information Theory ISIT 2021*, 2021.
- 2 Divesh Aggarwal, Ivan Damgård, Jesper Buus Nielsen, Maciej Obremski, Erick Purwanto, João Ribeiro, and Mark Simkin. Stronger leakage-resilient and non-malleable secret sharing schemes for general access structures. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 510–539. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26951-7_18.
- 3 Saikrishna Badrinarayanan and Akshayaram Srinivasan. Revisiting non-malleable secret sharing. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 593–622. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17653-2_20.
- 4 Fabrice Benhamouda, Akshay Degwekar, Yuval Ishai, and Tal Rabin. On the local leakage resilience of linear secret sharing schemes. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 531–561. Springer, Heidelberg, August 2018. doi:10.1007/978-3-319-96884-1_18.
- 5 Allison Bishop, Valerio Pastro, Rajmohan Rajaraman, and Daniel Wichs. Essentially optimal robust secret sharing with maximal corruptions. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 58–86. Springer, Heidelberg, May 2016. doi:10.1007/978-3-662-49890-3_3.
- 6 Andrej Bogdanov, Yuval Ishai, and Akshayaram Srinivasan. Unconditionally secure computation against low-complexity leakage. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 387–416. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26951-7_14.
- 7 Viktor Bouniakowsky. *Sur les diviseurs numériques invariables des fonctions rationnelles entières*. De l’Imprimerie de l’Académie impériale des sciences, 1854.
- 8 Luís T. A. N. Brandão and René Peralta. NIST first call for multi-party threshold schemes. <https://csrc.nist.gov/publications/detail/nistir/8214c/draft>, January 25, 2023.
- 9 Nishanth Chandran, Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Short leakage resilient and non-malleable secret sharing schemes. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 178–207. Springer, Heidelberg, August 2022. doi:10.1007/978-3-031-15802-5_7.

- 10 Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 398–412. Springer, Heidelberg, August 1999. doi:10.1007/3-540-48405-1_26.
- 11 Eshan Chattopadhyay, Jesse Goodman, Vipul Goyal, Ashutosh Kumar, Xin Li, Raghu Meka, and David Zuckerman. Extractors and secret sharing against bounded collusion protocols. In *61st FOCS*, pages 1226–1242. IEEE Computer Society Press, November 2020. doi:10.1109/FOCS46700.2020.00117.
- 12 Roni Con, Noah Shetty, Itzhak Tamo, and Mary Wootters. Repairing reed-solomon codes over prime fields via exponential sums. In *2023 IEEE International Symposium on Information Theory (ISIT)*, pages 1330–1335. IEEE, 2023. doi:10.1109/ISIT54713.2023.10206937.
- 13 Roni Con and Itzhak Tamo. Nonlinear repair of reed-solomon codes. *IEEE Trans. Inf. Theory*, 68(8):5165–5177, 2022. doi:10.1109/TIT.2022.3167615.
- 14 Sebastian Faust, Loïc Masure, Elena Micheli, Maximilian Ortl, and François-Xavier Standaert. Connecting leakage-resilient secret sharing to practice: Scaling trends and physical dependencies of prime field masking. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024*, pages 316–344, Cham, 2024. Springer Nature Switzerland. doi:10.1007/978-3-031-58737-5_12.
- 15 Serge Fehr and Chen Yuan. Towards optimal robust secret sharing with security against a rushing adversary. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 472–499. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17659-4_16.
- 16 Serge Fehr and Chen Yuan. Robust secret sharing with almost optimal share size and security against rushing adversaries. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 470–498. Springer, Heidelberg, November 2020. doi:10.1007/978-3-030-64381-2_17.
- 17 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987. doi:10.1145/28395.28420.
- 18 Louis Goubin and Jacques Patarin. DES and differential power analysis (the “duplication” method). In Çetin Kaya Koç and Christof Paar, editors, *CHES'99*, volume 1717 of *LNCS*, pages 158–172. Springer, Heidelberg, August 1999. doi:10.1007/3-540-48059-5_15.
- 19 Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th ACM STOC*, pages 685–698. ACM Press, June 2018. doi:10.1145/3188745.3188872.
- 20 Venkatesan Guruswami and Mary Wootters. Repairing reed-solomon codes. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 216–226. ACM Press, June 2016. doi:10.1145/2897518.2897525.
- 21 Venkatesan Guruswami and Mary Wootters. Repairing reed-solomon codes. *IEEE Trans. Inf. Theory*, 63(9):5684–5698, 2017. doi:10.1109/TIT.2017.2702660.
- 22 Alfréd Haar. Aur theorie der orthogonalen funktionensysteme. *Math. Annalen*, 69:331–371, 1910.
- 23 Jonathan I. Hall. Notes on coding theory. <https://users.math.msu.edu/users/halljo/classes/codenotes/GRS.pdf>, 2015.
- 24 JL Hammond Jr and RS Johnson. A review of orthogonal square-wave functions and their application to linear networks. *Journal of the Franklin Institute*, 273(3):211–225, 1962.
- 25 Walter J Harrington and John W Cell. A set of square-wave functions orthogonal and complete in $l_2(0, 2)$. *Duke Math. J.*, 28(1):393–407, 1961.
- 26 Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, and Mor Weiss. The price of active security in cryptographic protocols. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 184–215. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45724-2_7.

- 27 Jihun Hwang, Hemanta K. Maji, Hai H. Nguyen, and Xiuyu Ye. Security of shamir's secret-sharing against physical bit leakage: Secure evaluation places. <https://www.cs.purdue.edu/homes/hmaji/papers/HMNY24.pdf>, 2023.
- 28 Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 104–113. Springer, Heidelberg, August 1996. doi:10.1007/3-540-68697-5_9.
- 29 Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer, Heidelberg, August 1999. doi:10.1007/3-540-48405-1_25.
- 30 Ashutosh Kumar, Raghu Meka, and Amit Sahai. Leakage-resilient secret sharing against colluding parties. In David Zuckerman, editor, *60th FOCS*, pages 636–660. IEEE Computer Society Press, November 2019. doi:10.1109/FOCS.2019.00045.
- 31 Jeffrey C Lagarias. The computational complexity of simultaneous diophantine approximation problems. *SIAM Journal on Computing*, 14(1):196–209, 1985. doi:10.1137/0214016.
- 32 Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische annalen*, 261(ARTICLE):515–534, 1982.
- 33 Hemanta K. Maji, Hai H. Nguyen, Anat Paskin-Cherniavsky, Tom Suad, and Mingyuan Wang. Leakage-resilience of the shamir secret-sharing scheme against physical-bit leakages. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 344–374. Springer, Heidelberg, October 2021. doi:10.1007/978-3-030-77886-6_12.
- 34 Hemanta K. Maji, Hai H. Nguyen, Anat Paskin-Cherniavsky, Tom Suad, Mingyuan Wang, Xiuyu Ye, and Albert Yu. Tight estimate of the local leakage resilience of the additive secret-sharing scheme & its consequences. In Dana Dachman-Soled, editor, *3rd Conference on Information-Theoretic Cryptography, ITC 2022, July 5-7, 2022, Cambridge, MA, USA*, volume 230 of *LIPIcs*, pages 16:1–16:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITC.2022.16.
- 35 Hemanta K. Maji, Hai H. Nguyen, Anat Paskin-Cherniavsky, and Xiuyu Ye. Constructing leakage-resilient shamir's secret sharing: Over composite order fields. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024*, pages 286–315. Springer Nature Switzerland, 2024. doi:10.1007/978-3-031-58737-5_11.
- 36 Pasin Manurangsi, Akshayaram Srinivasan, and Prashant Nalini Vasudevan. Nearly optimal robust secret sharing against rushing adversaries. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 156–185. Springer, Heidelberg, August 2020. doi:10.1007/978-3-030-56877-1_6.
- 37 Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and kdc's. In *International conference on the theory and applications of cryptographic techniques*, pages 327–346. Springer, 1999. doi:10.1007/3-540-48910-X_23.
- 38 Hai Nguyen. Physical bit leakage resilience of linear code-based secret sharing. In *Eurocrypt 2025*. Springer, 2025.
- 39 Hans Rademacher. Einige sätze über reihen von allgemeinen orthogonalfunktionen. *Mathematische Annalen*, 87(1-2):112–138, 1922.
- 40 Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979. doi:10.1145/359168.359176.
- 41 Akshayaram Srinivasan and Prashant Nalini Vasudevan. Leakage resilient secret sharing and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 480–509. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26951-7_17.
- 42 R. Tittsworth. Coherent detection by quasi-orthogonal square-wave pulse functions (corresp.). *IRE Transactions on Information Theory*, 6(3):410–411, 1960. doi:10.1109/TIT.1960.1057575.

- 43 Anastassios Voudouris, Ilias Politis, and Christos Xenakis. Secret sharing a key in a distributed way, lagrange vs newton. In *Proceedings of the 17th International Conference on Availability, Reliability and Security, ARES '22*, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3538969.3544424.
- 44 Joseph L Walsh. A closed set of normal orthogonal functions. *American Journal of Mathematics*, 45(1):5–24, 1923.

A Solving Simultaneous Diophantine Equations

Figure 4 presents our algorithm. In this section, the “LLL algorithm” refers to the algorithm with the following guarantees.

► **Theorem 24** (LLL [32, Proposition 1.39]). *There exists a polynomial-time algorithm that, given a positive integer d and rational numbers $r_1, r_2, \dots, r_d, \varepsilon$ satisfying $0 < \varepsilon < 1$, finds integers $s_1, s_2, \dots, s_d$, and t for which*

$$|s_i - t \cdot r_i| \leq \varepsilon,$$

for $1 \leq i \leq d$ and $1 \leq t \leq 2^{d(d+1)/4} \cdot \varepsilon^{-d}$.

Input. $\alpha_1, \alpha_2 \in F^*$, where F is the prime field of order p

Output. Elements $u, v \in F^*$ such that $(u, v) \in [\alpha_1 : \alpha_2]$ and

$$u, v \in \{-B, -(B-1), \dots, 0, 1, \dots, (B-1), B\} \pmod{p},$$

where $B := \lceil 2^{3/4} \cdot \sqrt{p} \rceil$.

Algorithm.

1. Interpret $\alpha_1, \alpha_2 \in \{0, 1, \dots, p-1\}$ as positive integers
2. Define $d = 2$
3. Define $r_1 = \alpha_1/p \in \mathbb{Q}$ and $r_2 = \alpha_2/p \in \mathbb{Q}$
4. Define $\varepsilon = B/p \in \mathbb{Q}$
5. Use the LLL algorithm to find integers s_1, s_2 , and t
6. Interpret t as an element of F . Define $u = \alpha_1 \cdot t \in F$ and $v = \alpha_2 \cdot t \in F$

■ **Figure 4** Our Algorithm to obtain (u, v) from (α_1, α_2) using the LLL-algorithm.

Let us proceed to analyze our algorithm of Figure 4. The parameter setting needs to ensure that $t \leq 2^{d(d+1)/4} \varepsilon^{-d} < p$. Recall that $\varepsilon = B/p$. Substituting this value and rearranging, one needs to ensure that $2^{d(d+1)/4} \cdot p^{d-1} < B^d$. Therefore we have chosen $B = \lceil 2^{(d+1)/4} p^{1-1/d} \rceil$. Consequently, one can interpret t as an F^* element.

By definition, $(u, v) \in [\alpha_1 : \alpha_2]$ because $u = t \cdot \alpha_1$ and $v = t \cdot \alpha_2$. Next, note that

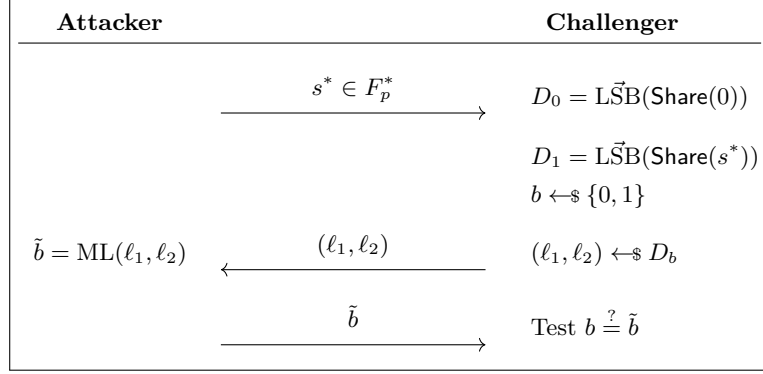
$$|\alpha_1 \cdot t - s_1 \cdot p| \leq \varepsilon \cdot p = B, \text{ and } |\alpha_2 \cdot t - s_2 \cdot p| \leq \varepsilon \cdot p = B.$$

This argument completes the analysis that for every (α_1, α_2) how we obtain $(u, v) \in [\alpha_1 : \alpha_2]$ such that u and v are “small (positive/negative) numbers.”

B Efficient Distinguisher Construction

Consider the following security game (illustrated in the figure below). The attacker picks a secret $s \in F_p^*$ and sends it to the challenger. The challenger picks a random bit $b \in \{0, 1\}$. If $b = 0$, the challenger samples (ℓ_1, ℓ_2) from distribution $D_0 := \vec{\text{LSB}}(\text{Share}(0))$ and sends it to

the attacker. Otherwise, the challenger samples (ℓ_1, ℓ_2) from distribution $D_1 := \vec{\text{LSB}}(\text{Share}(s))$ and sends it to the attacker. The attacker aims to guess which distribution (ℓ_1, ℓ_2) is sampled from. It uses the maximum likelihood decoder and then returns its guess $\tilde{b}$ to the challenger. The attacker wins the security game if $b = \tilde{b}$.



The maximum likelihood distinguisher outputs

$$\tilde{b} = \begin{cases} 0 & \text{if } \Pr[(\ell_1, \ell_2)|s = 0] \geq \Pr[(\ell_1, \ell_2)|s = s^*] \\ 1 & \text{if } \Pr[(\ell_1, \ell_2)|s = 0] < \Pr[(\ell_1, \ell_2)|s = s^*] \end{cases}$$

In other words, the output depends on $\text{sign}(\Pr[(\ell_1, \ell_2)|s = 0] - \Pr[(\ell_1, \ell_2)|s = s^*])$. The maximum likelihood distinguisher can compute $(-1)^{\ell_1 + \ell_2} \cdot \text{sign}_p(u) \cdot \text{sign}_p(v)$. If $(-1)^{\ell_1 + \ell_2} \cdot \text{sign}_p(u) \cdot \text{sign}_p(v) > 0$, then it outputs $\tilde{b} = 0$. Otherwise, it outputs $\tilde{b} = 1$. We provide a complete proof of the distinguishing advantage and security guarantee of this adversary in the full version of our paper [27].

C Attack on ShamirSS(3, 3, $\vec{\alpha}$)

Consider ShamirSS(3, 3, $\vec{\alpha}$) and the underlying prime field F of order $p = 4w^2 + 6w + 9$ where $w \geq 4$ and $w \not\equiv 0 \pmod{3}$. The evaluation places are $\vec{\alpha} = (1, \sigma, \sigma^2)$ for $\sigma = 2w \cdot 3^{-1} \in F_p$.

▷ **Claim 25.** $(2w \cdot 3^{-1})^3 = 1 \pmod{p}$ when $p = w^2 + 6w + 9$ and $w \geq 4$.

Proof. $(2w \cdot 3^{-1})^3 = 1 \pmod{p} \iff (2w)^3 - 3^3 = 0 \pmod{p} \iff (2w - 3) \cdot (4w^2 + 6w + 9) = 0 \pmod{p}$ holds since $p = 4w^2 + 6w + 9$. $\triangleleft$

Observe that $2w < \sqrt{p}$ and $3 < \sqrt{p}$. Then, by our classifier in Figure 1, $[1 : \sigma]$ is a good evaluation place since $[1 : \sigma] = [3 : 2w]$, $\gcd(3, 2w) = 1$, and $3 \cdot 2w$ is an even integer.

Note that $1 + \sigma + \sigma^2 = 1$; therefore, this secret sharing inherits the vulnerability of the additive secret sharing against LSB leakage [33]. Therefore, ShamirSS(3, 3, $\vec{\alpha}$) is insecure against LSB leakage, where its insecurity is $\geq (2/\pi)^3 \geq 0.25$ [34, 14].

D Derandomization for $n = k = 3$

Consider $\beta_1 = \frac{1}{\alpha_1(\alpha_1 - \alpha_2)(\alpha_1 - \alpha_3)}$ and $\beta_2 = \frac{1}{\alpha_2(\alpha_2 - \alpha_1)(\alpha_2 - \alpha_3)}$, we want to ensure $\frac{\beta_1}{\beta_2} = \Lambda$ where $\Lambda \in F^*$ is a good evaluation place in the ShamirSS(2, 2, $[1 : \Lambda]$) case. After expanding the expressing and solving the corresponding linear constraints, we obtain the following assignment.

$$\alpha_1 = \frac{2}{1 + \Lambda} \alpha_3 \quad \alpha_2 = \frac{1 - \Lambda}{1 + \Lambda} \alpha_3.$$

Specifically, $\alpha_1 = 2$, $\alpha_2 = 1 - \Lambda$, and $\alpha_3 = 1 + \Lambda$ suffices. We can ensure that $[\beta_1 : \beta_2]$ is secure with these evaluation places.

Time-Space Tradeoffs of Truncation with Preprocessing

Krzysztof Pietrzak 

IST, Klosterneuburg, Austria

Pengxiang Wang 

EPFL, Lausanne, Switzerland

Abstract

Truncation of cryptographic outputs is a technique that was recently introduced in Baldimtsi et al. [2]. The general idea is to try out many inputs to some cryptographic algorithm until the output (e.g. a public-key or some hash value) falls into some sparse set and thus can be compressed: by trying out an expected 2^k different inputs one will find an output that starts with k zeros.

Using such truncation one can for example save substantial gas fees on Blockchains where storing values is very expensive. While [2] show that truncation preserves the security of the underlying primitive, they only consider a setting without preprocessing. In this work we show that lower bounds on the time-space tradeoff for inverting random functions and permutations also hold with truncation, except for parameters ranges where the bound fails to hold for “trivial” reasons.

Concretely, it’s known that any algorithm that inverts a random function or permutation with range N making T queries and using S bits of auxiliary input must satisfy $S \cdot T \geq N \log N$. This lower bound no longer holds in the truncated setting where one must only invert a challenge from a range of size $N/2^k$, as now one can simply save the replies to all $N/2^k$ challenges, which requires $S = \log N \cdot N/2^k$ bits and allows to invert with $T = 1$ query.

We show that with truncation, whenever S is somewhat smaller than the $\log N \cdot N/2^k$ bits required to store the entire truncated function table, the known $S \cdot T \geq N \log N$ lower bound applies.

2012 ACM Subject Classification Theory of computation $\rightarrow$ Interactive proof systems; Theory of computation $\rightarrow$ Cryptographic protocols; Security and privacy $\rightarrow$ Information-theoretic techniques

Keywords and phrases Time-Space Lower Bounds, Blockchains

Digital Object Identifier 10.4230/LIPIcs.ITC.2025.4

1 Introduction

1.1 Truncation

In [2] Baldimtsi, Chalkias, Chatzigiannis and Kelkar suggested *truncation* as a technique to shorten outputs of cryptographic primitives like hash values, signatures or public keys. The technique applies whenever there’s a part of the input that can be chosen arbitrarily, like the randomness in signing and key generation or parts of the input when hashing. The general idea is to simply try out many different inputs until the output lands in some sparse domain, say it starts with Δ zeros, and such an output can be encoded using Δ less bits than a general output.

While the applicability of this technique is limited by the fact that finding an output that starts with Δ zeros will require around 2^Δ invocations of the primitive (so e.g., compressing by 20 bits already requires around a million invocations), in [2] interesting applications are identified where this can make a big difference, including in the context of the Ethereum blockchain, where saving a few bits to be stored on-chain can lead to significant savings. We discuss another application to proofs of space as used in the Chia network in the open problems Section 6.



© Krzysztof Pietrzak and Pengxiang Wang;
licensed under Creative Commons License CC-BY 4.0
6th Conference on Information-Theoretic Cryptography (ITC 2025).
Editor: Niv Gilboa; Article No. 4; pp. 4:1–4:10



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

While truncation puts extra burden on the evaluator (say, a signer), there's no extra cost for the parties that use the output (say, verify the signature). Moreover [2] show that truncation preserves security in the bit security framework of [9]. Informally, a primitive has bit security κ , if every adversary who breaks the security with advantage ϵ must run in time $2^\kappa \cdot \epsilon$.

1.2 Time-Space Tradeoffs

While the preservation of bit security under truncation gives some confidence in this technique, it only considers a setting without precomputation. Unfortunately, truncation can decrease the security of primitives when the adversary is given some auxiliary input. Let us illustrate this considering the one-wayness of a permutation over n bits, which we'll denote with $f : [N] \rightarrow [N]$ where $N = 2^n$. A random permutation f does have n bits of bit security, i.e., given a random $y \in [N]$, finding $x, f(x) = y$ will require $N/2$ invocations to f in expectation. But given S bits of advice (that depend on f but not the challenge y), it's possible to invert f using just T invocations whenever

$$S \cdot T \geq N \log(N) \quad (1)$$

The general idea is to compute and store values $x_0, x_T, x_{2T}, \dots$ where $x_i = f(x_{i-1})$. On challenge y one now applies f until one of the stored $x_{i \cdot T}$ values is hit, and then continues to apply f to $x_{(i-1)T}$ until one hits y . For functions the best space-time tradeoff are somewhat worse. For *random* functions an attack with

$$S^2 \cdot T \in \Theta(N^2 \log(N))$$

is achieved by Hellman tables [7] or rainbow tables [8]. So, any permutation can be inverted with time and space, e.g. $T = S \approx N^{1/2}$ while *random* functions can be inverted with $T = S \approx N^{2/3}$ (for functions that are not random the existing bounds are somewhat worse [5]).

De, Trevisan and Tulsiani [4] (building on work by Yao [11], Gennaro-Trevisan [6] and Wee [10]) prove that the attack as in eq. (1) is basically optimal as any adversary who inverts a *random* permutation or function with a range of size N must satisfy

$$S \cdot T \in \Omega(N) \quad (2)$$

Note that the above is seemingly wrong if $T = 0$ or $S = 0$ as one can invert with no space but N queries or $N \log N$ space and no queries. The proof of the lower bound assumes adversary must always query f on its output, so T is at least 1, and S is assumed to be at least $\log N$ which is required to store the challenge.

The same lower bound applies for random *functions*, though note that in this case it's not matching the upper bound. The actual lower bound is slightly more general showing that $S \cdot T \in \Omega(\epsilon \cdot N)$ must hold for any adversary who inverts with some probability ϵ , but for this introduction we assume ϵ is some constant.

1.3 Time-Space Tradeoffs with Truncation

Now consider the truncated setting, where we have a random function or permutation $f : [N] \rightarrow [N]$, but must only invert it on outputs sampled from some truncated range, say where the first Δ bits are set to zero. We'll denote this range with $[\delta N]$ where $\delta = 2^{-\Delta}$.

Now one can store the δN preimages of f on $[\delta N]$ using $S = \delta N \log N$ bits, and using this advice, it's possible to find the preimage x , $f(x) = y$ for any $y \in [\delta N]$ without invoking f at all. As outlined above, this means $T = 1$, but it still contradicts the lower-bound from eq. (2) as $\delta N \log N \notin \Omega(N)$ for $\delta \in o(1/\log N)$.

The main result in this work is Theorem 4, which basically states that the issue just described is the only reason for the lower bound to fail: as long as S is too small to basically store the entire function table of the *truncated* function, the $S \cdot T \in \Omega(N)$ lower bound applies. While we state and prove the bound for functions, it can easily be adapted to permutations. We discuss other truncated primitive in the open problems Section 6.

The technical result implying Theorem 4 is stated in Lemma 5. It uses a so-called compression argument: it shows how an adversary that inverts a random function f can be turned into an encoding for the function table of f , where the length of the encoding depends on the space and time efficiency of the adversary. As the function table of a random function is incompressible, as stated in Fact 1, we can derive a lower bound on the space and time complexity of the adversary.

1.4 The Compression Argument

The starting point for proving the Lemma is a proof of the $S \cdot T \geq N \log N$ lower bound for inverting random functions from [1]. This proof is less elegant than the proof of De et al. from [4], which uses a high level argument about sets, while [1] provide pseudocode of the encoding and argue about the length of its output.

Their encoding is basically Algorithm 1 in this paper for $\delta = 1$. It takes as input an adversary $\mathcal{A}$ who is guaranteed to invert some function $f : [N] \rightarrow [N]$ on an ϵ fraction of the range and making no more than T queries.

$\mathcal{A}$ is then invoked on some challenges, and every time $\mathcal{A}$ inverts a challenge we get one entry in the function table “for free”. With every invocation, $\mathcal{A}$ can make up to T queries to f , and those queries can later no longer be used as challenges, i.e., we “spoil” up to T of the $\epsilon \cdot N$ challenges with every successful inversion. Overall we can get $\mathcal{A}$ to invert something in the order of $\epsilon \cdot N/T$ challenges before running out of unspoiled challenges, and thus get an encoding that is around $\epsilon \cdot N/T$ bits shorter than the function table. As a random function is incompressible, this then implies that the advice used by $\mathcal{A}$ must be around $S \gtrsim \epsilon \cdot N/T$. That's how the $S \cdot T \in \Omega(\epsilon \cdot N)$ lower bound in [1] is proven.

In the truncated setting when $\delta \ll 1$, $\mathcal{A}$ is only guaranteed to invert on an ϵ fraction of the sparse domain $[\delta N]$, and the above argument would only gives us a $S \gtrsim \delta \cdot \epsilon \cdot N/T$ bound.

A key observation is the fact that we get a $S \gtrsim \epsilon \cdot N/T$ lower bound even in the truncated case if we assume that not much more than a δ fraction of the queries made by $\mathcal{A}$ map to the sparse $[\delta \cdot N]$ domain, as then each compressed entry only spoils around $\delta \cdot T$ of the $[\epsilon N]$ available challenges.

To prove our lemma we now make a case distinction. If for every compressed value we typically do not spoil more than $T_g \leq 12\delta T$ potential challenges, we use the observation above, so in this case we use basically the same argument as in [1].

In the other case the queries made by $\mathcal{A}$ fall into the sparse $[\delta N]$ set at least 12 times more often than a random query would. Knowing that the output $f(x)$ on a query x is in $[\delta N]$ means we can encode it using just $\log N - \log 1/\delta$ bits. We will encode the positions of the queries made by $\mathcal{A}$ that fall into $[\delta N]$ (there are around $\epsilon \delta N$ such queries) and then encode each output using just $\log N - \log 1/\delta$ bits. As the queries that fall into $[\delta N]$ are sufficiently dense (i.e., 12 times denser than a random query would), encoding those positions

4:4 Time-Space Tradeoffs of Truncation with Preprocessing

uses a bit less per entry than the $\log 1/\delta$ bits we save knowing the entry is in $[\delta N]$. So overall we save $\epsilon \delta N$ bits, and thus for this case can conclude (again using the incompressibility of a random function table) that the space used by $\mathcal{A}$ must be at least $S \gtrsim \epsilon \delta N$.

2 Notation and Basic Facts

We use brackets like $(x_1, x_2, \dots)$ or $\{x_1, x_2, \dots\}$ to denote ordered sets (aka. lists) and unordered sets, respectively. $[N]$ denotes some domain of size N , and for notational convenience we assume $N = 2^n$ is a power of two and identify $[N]$ with $\{0, 1\}^n$. For some $\Delta \leq n$ and $\delta = 2^{-\Delta}$, we'll denote with $[\delta N]$ some subset of $[N]$ of size δN (the truncated range), say $0^\Delta \| \{0, 1\}^{n-\Delta}$, the set of n bits strings that start with Δ zeros, but in principle any subset whose elements can be compressed to (not much more than) $n - \Delta$ bits will do.

For a function $f : [N] \rightarrow [M]$ and a set $S \subseteq [N]$, we denote with $f(S)$ the set $\{f(S[1]), \dots, f(S[|S|])\}$, similarly for a list $L \subseteq [N]$, $f(L)$ is the list $(f(L[1]), \dots, f(L[|L|]))$.

Randomized adversaries are treated as if they were deterministic. However, this is w.l.o.g. as they are only invoked within encoding/decoding procedures that have access to a shared randomness that can be used to derandomize them.

The following well known fact captures the fact that one cannot compress a random string.

► **Fact 1** (from [4]). *For any randomized encoding procedure $\text{Enc} : \{0, 1\}^r \times \{0, 1\}^n \rightarrow \{0, 1\}^m$ and decoding procedure $\text{Dec} : \{0, 1\}^r \times \{0, 1\}^m \rightarrow \{0, 1\}^n$ where*

$$\Pr_{x \leftarrow \{0, 1\}^n, r \leftarrow U_{|r|}} [\text{Dec}(r, \text{Enc}(r, x)) = x] \geq \delta$$

we have $m \geq n - \log(1/\delta)$

► **Fact 2.** *If a set X is at least ϵ dense in Y , i.e., $X \subset Y$, $|X| \geq \epsilon|Y|$, and Y is known, then X can be encoded using $|X| \cdot \log(e/\epsilon)$ additional bits.*

This fact follows from the inequality $\binom{n}{\epsilon n} \leq (en/\epsilon)^{\epsilon n}$, which implies $\log \binom{n}{\epsilon n} \leq \epsilon n \log(e/\epsilon)$. Encoding X can be done by identifying which $\epsilon|Y|$ elements to choose from Y .

3 Main Theorem and Lemma

De, Trevisan and Tulsiani [4] show that the simple time-space tradeoff eq. (1) for inverting permutations is basically tight.

► **Theorem 3** ([4], as stated in [1]). *Fix some $\epsilon \geq 0$ and an oracle algorithm $\mathcal{A}_{\text{aux}}$ that takes an advice string aux of length $|\text{aux}| = S$ and makes at most T oracle queries. If with non-negligible probability for a random function (or permutation) $f : [N] \rightarrow [N]$ there exists a string aux such that*

$$\Pr_{y \leftarrow [N]} [f(\mathcal{A}_{\text{aux}}^f(y)) = y] \geq \epsilon$$

then

$$T \cdot S \in \Omega(\epsilon N) . \tag{3}$$

In this work we show that this result extends to the case where the challenge comes from a sparse set $[\delta N]$

► **Theorem 4** (main). *Fix some $\epsilon \geq 0$ and an oracle algorithm $\mathcal{A}_{\text{aux}}$ that takes an advice string aux of length $|\text{aux}| = S$ and makes at most T oracle queries. If with non-negligible probability for a random function (or permutation) $f : [N] \rightarrow [N]$ there exists a string aux such that*

$$\Pr_{y \leftarrow [\delta N]} [f(\mathcal{A}_{\text{aux}}^f(y)) = y] \geq \epsilon$$

then either

$$S \in \Omega(\delta \epsilon N) \quad \text{or} \quad T \cdot S \in \Omega(\epsilon N) . \quad (4)$$

The theorem is proven using a compression argument as stated in the following lemma.

► **Lemma 5** (generalized version of a Lemma from [1]). *Let $\mathcal{A}_{\text{aux}}, T, S, \epsilon$ and f be as Theorem 4, and assume $T \leq \delta \epsilon N / 40$. There are randomized encoding and decoding procedures Enc, Dec such that if $f : [N] \rightarrow [N]$ is a function and for some $\text{aux}, |\text{aux}| = S$*

$$\Pr_{y \leftarrow [\delta N]} [f(\mathcal{A}_{\text{aux}}^f(y)) = y] \geq \epsilon$$

then

$$\Pr_{r \leftarrow U_{|r|}} [\text{Dec}(r, \text{Enc}(r, \text{aux}, f)) = f] \geq 0.9 \quad (5)$$

and the length of $\text{Enc}(r, \text{aux}, f)$ is at most

$$\underbrace{N \log N}_{=|f|} - \frac{\epsilon \delta N}{2T_g} + S + \log(N) \quad (6)$$

Moreover for some $T_g, 1 \leq T_g \leq T$ which is defined by the encoding algorithm: if $T_g \geq 12\delta T$, we can improve the length of the encoding to

$$\underbrace{N \log N}_{=|f|} - \frac{\epsilon \delta N}{2T_g} + S + \log(N) - \delta \epsilon N \quad (7)$$

The T_g above is the average number of f queries that land in the sparse set (i.e., x s.t. $f(x) \in [\delta N]$) that $\mathcal{A}_{\text{aux}}^f$ (when invoked by Enc as defined by Algorithm 1 below) makes for every value it inverts. As $\mathcal{A}$ makes at most T queries per challenge we have $T_g \leq T$. For a random query $x \in [N]$ we have $f(x) \in [\delta N]$ with probability δ , so if T_g is significantly larger than δT , this means that the queries made by $\mathcal{A}_{\text{aux}}$ are special in the sense that they map to $[\delta N]$ much more often than random queries would. We use this crucial observation for a case distinction, deriving the left or the right hand side of eq. (4), depending on whether T_g is below or above $12\delta T$.

4 How Theorem 4 follows from Lemma 5

4.1 Proof of Thm. 4 if $T_g \leq 12\delta T$

If $T_g \leq 12\delta T$, the theorem follows from Lemma 5 using Fact 1 as follows: assume the function table of f in the lemma is chosen uniformly at random (i.e., x in Fact 1 is a uniform $N \log N$ bit string), then the term in eq. (6) can be lower bounded as

$$\underbrace{N \log N}_{=|f|} - \frac{\epsilon \delta N}{2T_g} + S + \log(N) \geq N \log N - \log(1/0.9).$$

4:6 Time-Space Tradeoffs of Truncation with Preprocessing

Reordering we get

$$S \geq \frac{\epsilon\delta N}{2T_g} - \log(N) - \log(1/0.9)$$

using our assumption that $T_g \leq 12\delta T$

$$S \geq \frac{\epsilon N}{24T} - \log(N) - \log(1/0.9)$$

$$T \cdot S \geq \frac{\epsilon N}{24} - T \cdot \log(N) - T \cdot \log(1/0.9).$$

So $T \cdot S \in \Omega(\epsilon N)$ as claimed (on the rhs of eq. (4) in the Theorem). Note that the extra assumption that $T \leq \epsilon N/40$ in the lemma doesn't matter, as if it's not satisfied the theorem is trivially true.

4.2 Proof of Thm. 4 if $T_g > 12\delta T$

If $T_g > 12\delta T$, the theorem again follows from Lemma 5 using Fact 1, i.e., we again assume the function table of f in the lemma is chosen uniformly at random (i.e., x in Fact 1 is a uniform $N \log N$ bit string), and now the term in eq. (7) can be lower bounded as

$$\underbrace{N \log N}_{=|f|} - \frac{\epsilon\delta N}{2T_g} + S + \log(N) - \delta\epsilon N \geq N \log N - \log(1/0.9).$$

Note that, as in this case we use eq. (7) rather than eq. (6), we have an extra $-\delta\epsilon N$ term on the lhs. Reordering we get

$$S \geq \frac{\epsilon\delta N}{2T_g} - \log(N) - \log(1/0.9) + \delta\epsilon N$$

and thus $S \in \Omega(\delta\epsilon N)$ as claimed.

5 Proof of Lemma 5

We always assume that if $A_{\text{aux}}^f(y)$ outputs some value x , it makes the query $f(x)$ at some point. This is basically w.l.o.g. as we can turn any adversary into one satisfying this by making at most one extra query. If at some point $A_{\text{aux}}^f(y)$ makes an oracle query x where $f(x) = y$, then we also w.l.o.g. assume that right after this query A outputs x and stops.

5.1 The Size of the Encoding

We will now upper bound the size of the encoding of $G, f(Q'), (|q_1|, \dots, |q_{|G|}|), f([N] - \{G^{-1} \cup Q'\})$ as output in line (15) of the **Enc** algorithm.

Let $T_g := |B|/|G|$ be the average number of elements we added to the bad set B for every element added to the good set G , then

$$|G| \geq N\epsilon\delta/2T_g. \tag{8}$$

To see this we note that when we leave the while loop (see line (8) of the algorithm **Enc**) it holds that

$$|B| \geq |J|/2 = \epsilon\delta N/2 \text{ so } |G| = |B|/T_g \geq |J|/2T_g = N\epsilon\delta/2T_g \tag{9}$$

■ **Algorithm 1** Enc.

-
- 1: **Input:** $\mathcal{A}$, aux, randomness r and a function $f : [N] \rightarrow [N]$ to compress (using that $\mathcal{A}_{\text{aux}}^f(\cdot)$ inverts f on some ϵ fraction of $[\delta N]$)
 - 2: Initialize: $B, G := \emptyset, c := -1$
 - 3: Throughout we identify $[N]$ with $\{0, \dots, N-1\}$ and $[\delta N]$ with $\{0, \dots, \delta N-1\}$
 - 4: Pick a random permutation $\pi : [\delta N] \rightarrow [\delta N]$ (using random coins r)
 - 5: Let $J := \{y \in [\delta N] : f(\mathcal{A}_{\text{aux}}^f(y)) = y\}$, $|J| = \epsilon \delta N$ $\triangleright$ The set $J \subset [\delta N]$ where $\mathcal{A}$ inverts
 - 6: For $i = 0, \dots, \delta N - 1$ define $y_i := \pi(i)$. $\triangleright$ Randomize the order
 - 7: For $y \in J$ let $q(y)$ denote all queries made by $\mathcal{A}^f(y)$ except the last query (which is x s.t. $f(x) = y$).
 - 8: **while** $|B| < |J|/2 (= \epsilon \delta N/2)$ **do** $\triangleright$ While the bad set contains less than half of J
 - 9: $c := \min\{c' > c : y_{c'} \in \{J \setminus B\}\}$ $\triangleright$ Increase c to the next y_c in $J \setminus B$
 - 10: $G := G \cup y_c$ $\triangleright$ Add this y_c to good set
 - 11: $B := B \cup (f(q(y_c)) \cap J)$ $\triangleright$ Add spoiled queries to bad set
 - 12: **end while**
 - 13: Let $G = \{g_1, \dots, g_{|G|}\}$, $Q = q(g_1), \dots, q(g_{|G|})$, and define $Q' = (q'_1, \dots, q'_{|G|})$, $q'_i \subseteq q(g_i)$ to contain only the “fresh” queries in Q by deleting all but the first occurrence of every element. E.g. if $(q(g_1), q(g_2)) = ((1, 2, 3, 1), (2, 4, 5, 4))$ then $(q'_1, q'_2) = ((1, 2, 3), (4, 5))$.
 - 14: Let $G^{-1} = \{\mathcal{A}_{\text{aux}}^f(y) : y \in G\}$
 - 15: Output an encoding of (the set) G , (the lists) $f(Q'), (|q'_1|, \dots, |q'_{|G|}|)$, (the remaining outputs) $f([N] - \{G^{-1} \cup Q'\})$ and (the advice string) aux. $\triangleright$ We discuss how exactly this encoding looks in the proof. The encoding of $f(Q')$ will depend on the average number of spoiled queries per challenge $T_g = |B|/|G| (\approx \epsilon \delta N/2|G|)$
-

G : Instead of G we will actually encode the set $\pi^{-1}(G) = \{c_1, \dots, c_{|G|}\}$, from this encoding Dec (who gets r , and thus knows π) can then reconstruct $G = \pi(\pi^{-1}(G))$. We claim that the elements in $c_1 < c_2 < \dots < c_{|G|}$ are whp. at least $\epsilon \delta/2$ dense in $[c_{|G|}]$ (equivalently, $c_{|G|} \leq 2|G|/\epsilon \delta$). By Fact 2 we can thus encode $\pi^{-1}(G)$ using $|G| \log(2e/\epsilon \delta) + \log N$ bits (the extra $\log N$ bits are used to encode the size of G which is required so decoding later knows how to parse the encoding). To see that the c_i 's are $\epsilon \delta/2$ dense whp. consider line (9) in Enc which states $c := \min\{c' > c : y_{c'} \in \{J \setminus B\}\}$. If we replace $J \setminus B$ with J , then the c_i 's would be whp. close to $\epsilon \delta$ dense in $[N]$ as $J \subset N$ has size $\epsilon \delta N$ and the y_i are uniformly random. As $|B| < |J|/2$, using $J \setminus B$ instead of J will decrease the density by at most a factor 2. If we don't have this density, i.e., $c_{|G|} > 2|G|/\epsilon \delta$, we consider encoding to have failed.

$(|q'_1|, \dots, |q'_{|G|}|)$: Require $|G| \log T$ bits as each $q'_i \leq q_i \leq T$. A more careful argument (using Fact 2 and that the q'_i are on average at most T_g) requires $|G| \log(eT_g)$ bits.

$f([N] - \{G^{-1} \cup Q'\})$: Requires $(N - |G| - |Q'|) \log N$ bits (using that $G^{-1} \cap Q' = \emptyset$ and $|G^{-1}| = |G|$).

aux: Is S bits long.

$f(Q')$: This is a list of $|Q'|$ elements in $[N]$ and can be encoded using $|Q'| \log N$ bits, but we'll encode it with less if T_g is large.

■ **Algorithm 2** Dec.

-
- 1: **Input:** $\mathcal{A}, r$ and the encoding $(G, f(Q'), (|q'_1|, \dots, |q'_{|G|}|), f([N] - \{G^{-1} \cup Q'\}), \text{aux})$.
 - 2: Let π be as in **Enc**.
 - 3: Let $(g_1, \dots, g_{|G|})$ be the elements of G ordered as they were added by **Enc** (i.e., $\pi^{-1}(g_i) < \pi^{-1}(g_{i+1})$ for all i).
 - 4: Invoke $\mathcal{A}_{\text{aux}}^{(\cdot)}(\cdot)$ sequentially on inputs $g_1, \dots, g_{|G|}$ using $f(Q')$ to answer $\mathcal{A}_{\text{aux}}$'s oracle queries.
 - 5: Combine the mapping $G^{-1} \cup Q' \rightarrow f(G^{-1} \cup Q')$ (which we learned in the previous step) with $[N] - \{G^{-1} \cup Q'\} \rightarrow f([N] - \{G^{-1} \cup Q'\})$ to learn the entire $[N] \rightarrow f([N])$
 - 6: Output $f([N])$
-

Summing up we get

$$\begin{aligned}
& |\text{Enc}(r, \text{aux}, f)| \\
&= \underbrace{|G| \log(2e/\epsilon\delta) + \log N}_{\text{encoding of } G} + \underbrace{|Q'| \log N}_{f(Q')} + \underbrace{|G| \log(eT_g)}_{(|q'_1|, \dots, |q'_{|G|}|)} + \underbrace{(N - |G| - |Q'|) \log N}_{f([N] - \{G^{-1} \cup Q'\})} + \underbrace{S}_{\text{aux}} \\
&= |G| \log(2e^2 T_g / \epsilon\delta) + (N - |G| - |Q'|) \log N + |Q'| \log N + S + \log N.
\end{aligned}$$

Using the assumption $T_g \leq T \leq \epsilon\delta N/40$ in the statement of the lemma, which in turn implies $\log(2e^2 T_g / \epsilon\delta) \leq \log(N) - 1$, we get

$$\begin{aligned}
&\leq |G|(\log N - 1) + (N - |G| - |Q'|) \log N + |Q'| \log N + S + \log N \\
&= (N - |Q'|) \log N + |Q'| \log(N) - |G| + S + \log N \\
&= N \log N - |G| + S + \log N
\end{aligned} \tag{10}$$

Plugging in the bound from eq. (8) for $|G|$ we get

$$|\text{Enc}(r, \text{aux}, f)| \leq N \log N - \frac{N\delta\epsilon}{2T_g} + S + \log N$$

proving eq. (6) in the Lemma.

5.2 Improved bound if $T_g > 12\delta T$

We'll now prove a bound on the length of the encoding as stated in eq. (7) in the lemma. Here we assume that T_g is large, i.e., $T_g > 12\delta T$. The bound on the length of the encoding improves the expression we got without this assumption, i.e., eq. (6), by $\delta\epsilon N$ bits. We will achieve this by improving the length of the encoding of $f(Q')$ from the trivial $|Q'| \log(N)$ to $|Q'| \log(N) - \delta\epsilon N$ by exploiting the fact that now a large fraction of the elements in $f(Q')$ falls into the sparse set $[\delta N]$, i.e.,

▷ **Claim 6.** If $T_g > 12\delta T$, $f(Q')$ can be encoded using $|Q'| \log N - \delta\epsilon N$ bits.

With this claim we can improve eq. (10) to $(N - |Q'|) \log N + |Q'| \log(N) - |G| + S + \log N$, which now gives eq. (7) from the Lemma.

It remains to prove the claim. By eq. (8) and eq. (9) $|G| \geq \delta\epsilon N/2T_g$, $|Q'| \leq T \cdot |G|$ and $|B| \geq \delta\epsilon N/2$, which implies

$$|B|/|Q'| \geq \delta\epsilon N/2T \cdot |G| \geq T_g/T \geq 12\delta$$

I.e., a 12δ fraction of the queries made during decoding falls into $B \subset [\delta N]$. Using this with Fact 2 we can encode which of the $|B|$ queries from Q' map into B using $|B| \log(e/12\delta)$ bits. For any $x \in B$, we can encode $f(x)$ using $\log(N) - \log(1/\delta)$ bits as $f(x) \in [\delta N]$.

We can now encode $f(Q')$ by first encoding the positions of B in Q' , and then $f(B)$ and $f(Q' \setminus B)$ separately, which requires

$$\begin{aligned} & (|Q'| - |B|) \log N + |B| (\log(e/10\delta) + \log(N) - \log(1/\delta)) \\ = & |Q'| \log N + |B| \log(e/12) \\ \leq & |Q'| \log N - |B| \cdot 2 \\ \leq & |Q'| \log N - \delta \epsilon N \end{aligned}$$

bits as claimed.

6 Conclusion and Open Problems

In this work we showed that the known time-space tradeoff $S \cdot T \geq N$ for inverting random functions or permutations $f : [N] \rightarrow [N]$ also holds for truncated outputs almost up to the point where S is big enough to store the entire truncated function table. This shows that the general idea of truncating cryptographic primitives as suggested in [2] is secure even when preprocessing is considered (as long as the truncated function table is big enough so it can't be stored in practice).

While in this paper we only considered one-wayness of random functions, we believe that our result can be adapted to time-space lower bounds for other primitives, showing the bounds apply also for their truncated analogues. An interesting example considered in [2] is the discrete logarithm problem, for which a $S \cdot T^2 \geq N$ time-space lower bound is known [3].

A likely more challenging but particularly interesting case is the adaption of the “beyond Hellman” proofs of space from [1] to the truncated setting. The key primitive in [1] are functions $[N] \rightarrow [N]$ for which a time-space lower bound of $S^k \cdot T \geq N^k$ (for any constant k) can be proven. This improves on the $S \cdot T \geq N$ lower bound for “normal” functions, and the reason this doesn't contradict known upper bounds (by rainbow tables) is the fact that those functions cannot be efficiently evaluated in forward direction (but their entire function table can be computed in time N). A proof that truncation is secure for these functions would allow for better security. Currently those functions are deployed in the proof of space underlying the **chia.net** blockchain, but there's a trade-off that farmers (who are supposed to dedicate space analogous to miners dedicating computation in Bitcoin) can make: by dropping a few bits of every entry, they save space at the cost of having to do some extra computation when computing the proofs¹. One could use a truncated version of those functions, which would make the initialization of the space for the farmers somewhat more costly, but it would make such “bit dropping” attacks much less attractive.

References

- 1 Hamza Abusalah, Joël Alwen, Bram Cohen, Danylo Khilko, Krzysztof Pietrzak, and Leonid Reyzin. Beyond hellman's time-memory trade-offs with applications to proofs of space. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 357–379. Springer, Heidelberg, December 2017. doi:10.1007/978-3-319-70697-9_13.

¹ <https://github.com/Chia-Network/drplotter>

- 2 Foteini Baldimtsi, Konstantinos Chalkias, Panagiotis Chatzigiannis, and Mahimna Kelkar. Truncator: Time-space tradeoff of cryptographic primitives. Cryptology ePrint Archive, Paper 2022/1581, 2022. URL: <https://eprint.iacr.org/2022/1581>.
- 3 Henry Corrigan-Gibbs and Dmitry Kogan. The discrete-logarithm problem with preprocessing. 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 415–447. Springer, 2018. doi:10.1007/978-3-319-78375-8_14.
- 4 Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and PRGs. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 649–665. Springer, Heidelberg, August 2010. doi:10.1007/978-3-642-14623-7_35.
- 5 Amos Fiat and Moni Naor. Rigorous time/space tradeoffs for inverting functions. In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 534–541. ACM, 1991. doi:10.1145/103418.103473.
- 6 Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *41st FOCS*, pages 305–313. IEEE Computer Society Press, November 2000. doi:10.1109/SFCS.2000.892119.
- 7 Martin E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Trans. Inf. Theory*, 26(4):401–406, 1980. doi:10.1109/TIT.1980.1056220.
- 8 Philippe Oechslin. Making a faster cryptanalytic time-memory trade-off. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 617–630. Springer, 2003. doi:10.1007/978-3-540-45146-4_36.
- 9 Shun Watanabe and Kenji Yasunaga. Bit security as computational cost for winning games with high probability. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021 – 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part III*, volume 13092 of *Lecture Notes in Computer Science*, pages 161–188. Springer, 2021. doi:10.1007/978-3-030-92078-4_6.
- 10 Hoeteck Wee. On obfuscating point functions. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 523–532. ACM Press, May 2005. doi:10.1145/1060590.1060669.
- 11 Andrew Chi-Chih Yao. Coherent functions and program checkers (extended abstract). In *22nd ACM STOC*, pages 84–94. ACM Press, May 1990. doi:10.1145/100216.100226.

Information-Theoretic Random-Index PIR

Sebastian Kolby  

Aarhus University, Denmark

Lawrence Roy  

Aarhus University, Denmark

Jure Sternad  

Aarhus University, Denmark

Sophia Yakoubov  

Aarhus University, Denmark

Abstract

A Private Information Retrieval (PIR) protocol allows a client to learn the i th row of a database held by one or more servers, without revealing i to the servers. A Random-Index PIR (RPIR) protocol, introduced by Gentry *et al.* (TCC 2021), is a PIR protocol where, instead of being chosen by the client, i is random. This has applications in e.g. anonymous committee selection. Both PIR and RPIR protocols are interesting only if the communication complexity is smaller than the database size; otherwise, the trivial solution where the servers send the entire database suffices.

Unlike PIR, where the client must send at least one message (to encode information about i), RPIR can be executed in a single round of server-to-client communication. In this paper, we study such one-round, information-theoretic RPIR protocols. The only known construction in this setting is SimpleMSRPIR (Gentry *et al.*), which requires the servers to communicate approximately $\frac{N}{2}$ bits, N being the database size. We show an $\Omega(\sqrt{N})$ lower bound on communication complexity for one-round two-server information-theoretic RPIR, and a sublinear upper bound. Finally, we show how to use a sublinear amount of database-independent correlated randomness among multiple servers to get near-optimal online communication complexity (the size of one row plus the size of one index description per server).

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques

Keywords and phrases Private information retrieval, Multi-server, Lower bounds

Digital Object Identifier 10.4230/LIPIcs.ITC.2025.5

Funding Danish Independent Research Council under Grant-ID DFF-2032-00122B and DFF-2064-00016B (YOSO)

Acknowledgements Supported by the Danish Independent Research Council under Grant-ID DFF-2032-00122B and DFF-2064-00016B (YOSO).

1 Introduction

PIR (private information retrieval [4]) enables a client to retrieve the i th row of a database from one or more servers in such a way that i remains unknown to the server(s). The trivial solution involves the server(s) sending the client the entire database; however, it is desirable to bring the communication complexity down so that it is strictly smaller than the size of the database.

RPIR (random-index PIR [7]) is a relaxation of PIR, where instead of learning a database row of her choice, the client learns a random row. Just like PIR, RPIR guarantees that the index i of that row is unknown to the server(s). Any PIR construction can be converted to a RPIR construction, simply by having the client choose the index i at random. However, while PIR constructions require at least one message from the client (to communicate something about i) followed by at least one message from the server(s) (to communicate something about the database), not all RPIR constructions require the client to communicate.



© Sebastian Kolby, Lawrence Roy, Jure Sternad, and Sophia Yakoubov;
licensed under Creative Commons License CC-BY 4.0

6th Conference on Information-Theoretic Cryptography (ITC 2025).

Editor: Niv Gilboa; Article No. 5; pp. 5:1–5:15



Leibniz International Proceedings in Informatics

LIPIcs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Random-index PIR constructions that only take one round of server-client communication are termed “non-interactive”. Gentry *et al.* [7] show how to build one-server non-interactive RPIR from homomorphic encryption. They also show how to build two-server non-interactive information-theoretic RPIR (which they dub SimpleMSRPIR), the communication complexity of which is $\frac{dw}{2} + \log(d)$ (where the database contains d rows each of size w).

Our Contributions

In this paper, we take steps towards understanding the communication complexity of non-interactive information-theoretic RPIR. We prove that non-interactive two-server information-theoretic RPIR must have communication complexity at least $O(\sqrt{d})w$.

On the positive side, we give an upper bound with communication complexity

$$\frac{dw}{d^{\frac{\log \log(d)}{\log(d)}}} + \frac{dw}{\frac{\log(d)}{\log \log d} + 1} + \Theta(d \log(d))$$

bits, showing that the communication complexity can in fact be sub-linear in the database size dw (when $w = \omega(\log(d))$).¹

Finally, we describe a construction that uses a sublinear amount of correlated randomness, but gets the near-optimal online communication complexity of $n \log(d)w$ (where n is the number of servers, d is the number of database rows, and w is the size of each row).

Related Work

Information-theoretic Private Information Retrieval (PIR) was first introduced by Chor *et al.* [4], who provided a general construction for the case of n servers, achieving a total communication complexity of $O(d^{\frac{1}{n}})$. Additionally, they proposed a specialized construction for the two-server case with a communication complexity of $O(d^{\frac{1}{3}})$. Shortly after, Ambainis [1] significantly improved the upper bound for the n -server setting, reducing the complexity to $O(d^{\frac{1}{2n-1}})$. Beimel and Ishai [2] made further progress, considering a scenario in which up to t servers may collude. They presented a construction achieving a communication complexity of $O(d^{\frac{1}{\lceil (2n-1)/t \rceil}})$.

Wehner and de Wolf [11] made a key contribution to lower bounds in information-theoretic PIR, where they additionally parametrized the communication complexity by probe complexity b – a measure of the number of actual bits the client reads from the messages sent by the servers – as an additional parameter in the communication complexity analysis. In particular, they established a lower bound of $\Omega(d^{\frac{1}{b+1}})$ for the two-server case. Complementing this, they presented a scheme with $b = 1$ and communication complexity $O(\sqrt{d})$. Their results also demonstrated that the original scheme by Chor *et al.* [4], which with some modifications can be adapted to have $b = 3$, already achieved near-optimal complexity of $O(d^{\frac{1}{4}})$. Moreover, in the general case where the probe complexity equals the length of the messages sent by the two servers, their result implies a lower bound of $5 \log d$, which remains the best-known lower bound for two servers.

For the two-server setting, the upper bound of Chor *et al.* was finally improved by Dvir and Gopi [5], who established an upper bound of $n^{O(\sqrt{\frac{\log \log d}{\log d}})}$, which remains the best known result for this case. In the multi-server setting, the currently best-known upper

¹ Note that there is a qualitative gap between our lower and upper bounds as they are currently presented: Our lower bound relies on perfect correctness, while our upper bound allows a failure to occur with some probability. The lower bound can be extended to account for a fixed probability of failure.

bound is due to works [6, 9, 3]. Efremenko [6] provided a three-server construction with communication complexity $\exp(O(\sqrt{\log d \log \log d}))$ and a general 2^r -server construction with a communication complexity of $\exp(O(\sqrt[r]{\log d \log^{r-1} d}))$. Recently, Ghasemi, Kopparty and Sudan [8] improved Efremenko's three-server construction, reducing the exponent from square root to the cube root. Other improvements – such as a reduction of the number of required servers of Efremenko's 2^r -server construction – were made by Itoh and Suzuki [9], and Chee *et al.* [3].

1.1 Technical Overview

In Section 2 we describe our definitions of non-interactive multi-server RPIR, adapted from Gentry *et al.* [7]. In Section 3 we describe our lower bound for the communication complexity of such protocols. We use the fact that in non-interactive two-server RPIR without correlated randomness, the client can combine any message msg_1 from the first server with any message msg_2 from the second server to retrieve a database row. So, if we take $\sqrt{d}$ messages msg_1 and $\sqrt{d}$ messages msg_2 , the client can retrieve $\sqrt{d} \times \sqrt{d} = d$ database rows. Some of these may be the same, but we prove that, in expectation, the client ends up with cd different database rows for some constant c , which amounts to cdw bits of information. Communicating cdw bits of information naturally requires that many bits; so, if this much information can be extracted from $2\sqrt{d}$ messages, on average each of these messages must have size at least $\frac{cdw}{2\sqrt{d}} = O(\sqrt{d})w$.

In Section 4 we describe our sublinear upper bound (without correlated randomness). We build on the SimpleMSRPIR construction of Gentry *et al.*. In SimpleMSRPIR, one server sends a random database row, and the other server randomly pairs the rows and sends an XOR of each pair (as well as a concise description of the pairing). The client uses row i sent by the first server and the XOR of row i and row j sent by the second server to recover row j as well. With probability $\frac{1}{d}$ it returns row i ; the rest of the time, it returns row j . In SimpleMSRPIR, the servers send a total of $\frac{dw}{2} + 2\log(d) + w$ bits, which is linear in the size dw of the database. In our construction (which we dub “Bucket-PIR”), instead of sending a single row, the first server sends each row with probability p . Instead of pairing the database rows, the second server partitions them into buckets of size b . By setting $p = \frac{1}{d^{\frac{\log \log(d)}{\log(d)}}}$ and $b = \frac{\log(d)}{\log \log(d)} + 1$, we ensure that the client retrieves a database row with probability at least half, while pushing the total communication complexity to be sublinear in the size dw of the database as long as $w = O(\log(d))$. (The probability that the client retrieves a row can be boosted arbitrarily close to 1 via parallel repetition; notice that the number of repetitions depends only on the desired probability of success, and not on the database size, so it does not impact the sublinearity of our communication complexity.)

In Section 5, we describe our final construction, which uses correlated randomness. We consider a setting with $n \geq 2$ servers, each of which holds Shamir shares of u random one-hot vectors (with zeros in all but one place) of length s . By locally taking the tensor product of all of its vectors of shares, each server ends up with shares of a size- d random one-hot vector, with a one at i . Each server takes the dot product of this vector of shares with the database, and sends the resulting value to the client. The client can treat the messages it receives as shares which reconstruct exactly to the i th database row.

2 Definitions

We use a definition of multi-server random-index PIR based on that of Gentry *et al.* [7, Definition 3]. Consider n servers $\mathcal{S}_1, \dots, \mathcal{S}_n$, and a client $\mathcal{C}$. Since we limit ourselves to *non-interactive (one-round)* protocols, we can describe the entire protocol in terms of

algorithms Server_p (for $p \in [n]$) and Client . Each server $\mathcal{S}_p$ runs the algorithm Server_p on the database D and randomness ρ_p (from distribution $\mathcal{R}_S$) to obtain message msg_p . The client $\mathcal{C}$ runs the algorithm Client on messages $\text{msg}_1, \dots, \text{msg}_n$ and randomness ρ_C (from distribution $\mathcal{R}_C$) to obtain (i, D_i) .

Sometimes, the servers use database-independent correlated randomness. We use cr_p to denote server $\mathcal{S}_p$'s correlated randomness, and CR to denote the distribution from which $(\text{cr}_1, \dots, \text{cr}_n)$ is drawn. We use grey to denote elements of an MS-RPIR scheme that are only present when correlated randomness is used.

► **Definition 1** (MS-RPIR: Multi-Server Random-Index PIR for Databases With d Rows of Size w). *A semi-honest, non-interactive (one-round) c -correct n -server threshold- t random-index PIR scheme $(\{\text{Server}_p\}_{p \in [n]}, \text{Client}, \text{CR})$ must satisfy the following properties:*

c -Correctness ($0 < c \leq 1$): *For every database $D \in \{0, 1\}^{w \times d}$ and index $i \in [d]$, the client's probability of outputting D_i is $\frac{c}{d}$.*

More formally, for every database $D \in \{0, 1\}^{w \times d}$,

$$\Pr \left[\begin{array}{l} (i, D[i]) = \text{Client}(\text{msg}_1, \dots, \text{msg}_n; \rho_C) \\ \text{where} \\ \rho_p \leftarrow \mathcal{R}_S \text{ for } p \in [n], \rho_C \leftarrow \mathcal{R}_C; \\ (\text{cr}_1, \dots, \text{cr}_n) \leftarrow \text{CR}; \\ \text{msg}_p \leftarrow \text{Server}(D, \text{cr}_p; \rho_p) \text{ for } p \in [n] \end{array} \right] = \frac{c}{d},$$

for every index $i \in [d]$. (Any other output of Client is interpreted to be $\perp$.)

Client Privacy *For every database $D \in \{0, 1\}^{w \times d}$, for every set of corrupt servers $I \subset [n]$ s.t. $|I| \leq t$, the index of the row output by the client is independent of the view of the corrupt servers.*

More formally, the following two distributions should be statistically indistinguishable:

$$\left\{ \begin{array}{l} (i, \{\rho_p, \text{cr}_p\}_{p \in I}) \\ \text{where} \\ \rho_p \leftarrow \mathcal{R}_S \text{ for } p \in [n], \rho_C \leftarrow \mathcal{R}_C; \\ (\text{cr}_1, \dots, \text{cr}_n) \leftarrow \text{CR}; \\ \text{msg}_p \leftarrow \text{Server}(D, \text{cr}_p; \rho_p) \text{ for } p \in [n]; \\ (i, D[i]) = \text{Client}(\text{msg}_1, \dots, \text{msg}_n; \rho_C) \end{array} \right\} \sim \left\{ \begin{array}{l} (i, \{\rho_p, \text{cr}_p\}_{p \in I}) \\ \text{where} \\ \rho_p \leftarrow \mathcal{R}_S \text{ for } p \in [n]; \\ (\text{cr}_1, \dots, \text{cr}_n) \leftarrow \text{CR}; \\ i \leftarrow [d] \text{ with pr. } c, i = \perp \text{ otherwise} \end{array} \right\}$$

► **Remark 2.** Notice that we allow the client to output $\perp$ with constant probability. Parallel executions can bring c down arbitrarily close to 0.

Finally, the trivial solution of the server(s) sending the client the entire database, and the client selecting a random index i and outputting $(i, D[i])$, satisfies the above definitions. In order to be interesting, a MS-RPIR scheme must also be *non-trivial*:

► **Definition 3** (Non-Triviality). *An MS-RPIR scheme is non-trivial if the communication complexity of the protocol is asymptotically smaller than the size of the database.*

More formally, let $\text{CC}(n, w, d)$ be the expected communication complexity of the protocol (that is, the expected sum of $|\text{msg}_1|, \dots, |\text{msg}_n|$). The scheme is non-trivial if there exists a d_0 s.t. for all $d > d_0$, there exists a w_0 s.t. for all $w > w_0$,

$$\text{CC}(n, w, d) < wd.$$

3 A $\Omega(\sqrt{dw})$ Lower Bound on Communication Complexity

In this section, we focus on the scenario where we have two servers, one of which is corrupt. In this setting, the servers must send $\Omega(\sqrt{dw})$ bits. We will prove this by showing that choosing a random size $\sqrt{d}$ subset from the space of each server's possible messages in expectation allows recovering a constant fraction of the database.

► **Theorem 4** (1-out-of-2 IT MS-RPIR Lower Bound). *Any MS-RPIR protocol with perfect security for two servers, where one may be corrupt, must have at least $\Omega(\sqrt{dw})$ expected communication.*

Proof. Let $D \in (\{0, 1\}^w)^d$ be the database held by the two servers. For $i \in \{1, 2\}$ let s_i be the number of possible random tapes for $\mathcal{S}_i$ and s_0 be the number of possible random tapes for $\mathcal{C}$. For a fixed random tape a server will always send the same message; some distinct tapes may result in the same message.

For servers $\mathcal{S}_1$ and $\mathcal{S}_2$ we define a matrix $M \in [d]^{s_1 \times s_2}$, indexed by the choices of $\mathcal{S}_1$ and $\mathcal{S}_2$ randomness. Entry $M_{r,c}$ encodes the index $i \in [d]$ which may be recovered by the client for server randomnesses r and c . More formally,

$$(i, D[i]) \leftarrow \text{Client}(\text{Server}_1(D; r), \text{Server}_2(D; c); T_{r,c}),$$

where client randomness $T_{r,c}$ is chosen uniformly and independently for each of the $s_1 \cdot s_2$ entries of M .

First, we address the case where $\mathcal{S}_1$ has fewer than $\sqrt{d}$ possible choices of randomness. If we take all messages that might be sent by $\mathcal{S}_1$ and any single message sent by $\mathcal{S}_2$ it must be possible to recover the entire database. This follows by correctness and privacy, as the client must always be able to recover some row of the database given a pair of messages, and all rows must still be equally likely even when fixing the message of one server. Iterating over all $< \sqrt{d}$ of $\mathcal{S}_1$ messages and client randomness will allow the recovery of the database, and therefore the messages of $\mathcal{S}_1$ (together with any one message of $\mathcal{S}_2$) on average contains $\geq \sqrt{dw}$ bits of information. A symmetric argument can be made for $\mathcal{S}_2$ with fewer than $\sqrt{d}$ possible random tapes.

From now on we will only consider the case where $s_1, s_2 \geq \sqrt{d}$. We will analyse how much of the database is recovered given server messages for k_1 choices of $\mathcal{S}_1$ randomness and k_2 choices of $\mathcal{S}_2$ randomness. Sample k_i distinct choices of randomness for server $i \in \{1, 2\}$ as

$$R = (R_1, \dots, R_{k_1}), \quad C = (C_1, \dots, C_{k_2}).$$

For all rows $i \in [d]$ in the database, and $a \in [k_1], b \in [k_2]$, define the random variable:

$$M_{R_a, C_b}^i = \begin{cases} 1 & \text{if } (i, D[i]) = \text{Client}(\text{Server}_1(D; R_a), \text{Server}_2(D; C_b); T_{a,b}) \\ 0 & \text{otherwise} \end{cases}$$

We may count the occurrences of each row $i \in [d]$ across the sampled rows and columns,

$$A_{R,C}^i = \sum_{a \in [k_1], b \in [k_2]} M_{R_a, C_b}^i.$$

And define an indicator variable for whether row $i \in [d]$ is recovered for any pair,

$$B_{R,C}^i = \begin{cases} 0 & \text{if } A_{R,C}^i = 0 \\ 1 & \text{otherwise} \end{cases}$$

To prove Theorem 4, it suffices to show that

$$E \left[\sum_{i \in [d]} B_{R,C}^i \right] \approx \frac{d}{2},$$

since this expectation describes how many database rows can be retrieved in expectation from our $\Omega(\sqrt{d})$ messages.²

By linearity of expectation, $E \left[\sum_{i \in [d]} B_{R,C}^i \right] = \sum_{i \in [d]} E[B_{R,C}^i]$; so, we can focus on $E[B_{R,C}^i] = \Pr[A_{R,C}^i > 0]$ for a single i .

Going forward we suppress the superscript i , as we will only handle one row $i \in [d]$ at a time. By the second moment method,

$$\Pr[A_{R,C} > 0] \geq \frac{E[A_{R,C}]^2}{E[A_{R,C}^2]}.$$

We can tackle $E[A_{R,C}]$ and $E[A_{R,C}^2]$ separately.

Bounding the expectation of $A_{R,C}$

First, we handle the simple case for $E[A_{R,C}]$. By linearity of expectation,

$$E[A_{R,C}] = \sum_{a \in [k_1], b \in [k_2]} E[M_{R_a, C_b}] = \sum_{a \in [k_1], b \in [k_2]} \Pr[M_{R_a, C_b}].$$

For all $c \in [s_2]$, privacy implies $\Pr[M_{R_a, c}] = 1/d$. Therefore, $\Pr[M_{R_a, C_b}] = 1/d$, and it follows,

$$E[A_{R,C}] = \sum_{a \in [k_1], b \in [k_2]} \frac{1}{d} = \frac{k_1 \cdot k_2}{d}.$$

Bounding the expectation of $A_{R,C}^2$

Expanding the definition and by linearity of expectation,

$$E[(A_{R,C})^2] = E \left[\left(\sum_{a \in [k_1], b \in [k_2]} M_{R_a, C_b} \right)^2 \right] = \sum_{\substack{a \in [k_1], b \in [k_2] \\ c \in [k_1], d \in [k_2]}} E[M_{R_a, C_b} \cdot M_{R_c, C_d}].$$

We wish to derive an upper bound for this expectation. The terms in the sum may be split into four disjoint cases, (1) when $a = c$ and $b = d$, (2) when $a = c$ but $b \neq d$, (3) when $a \neq c$ but $b = d$, and (4) when $a \neq c$ and $b \neq d$.

We start with case (1) where $a = c$ and $b = d$,

$$\begin{aligned} \sum_{\substack{a=c \in [k_1] \\ b=d \in [k_2]}} E[M_{R_a, C_b} \cdot M_{R_c, C_d}] &= \sum_{\substack{a=c \in [k_1] \\ b=d \in [k_2]}} \Pr[M_{R_a, C_b}, M_{R_c, C_d}] = \sum_{\substack{a \in [k_1] \\ b \in [k_2]}} \Pr[M_{R_a, C_b}] \\ &= \sum_{a \in [k_1], b \in [k_2]} \frac{1}{d} = \frac{k_1 \cdot k_2}{d}. \end{aligned}$$

² It would suffice to show that $E \left[\sum_{i \in [d]} B_{R,C}^i \right] \approx cd$ for any constant c ; however, the math works out for $c = \frac{1}{2}$.

Before we proceed to the next cases, it will be helpful to define some notation. Let s_0 be the number of choices of client randomness. For $r \in [s_1]$ and $c \in [s_2]$ let $p_{r,c}$ be the number of client randomness choices for which the client outputs $(i, D[i])$ for server messages $\text{Server}_1(D; r), \text{Server}_2(D; c)$.

For any fixed $c \in [s_2]$ privacy implies the message i must occur for a fraction $1/d$ of the client choices and server 1 randomness,

$$\sum_{r=1}^{s_1} p_{r,c} = \frac{s_1 \cdot s_0}{d}.$$

Similarly, for any fixed $r \in [s_1]$,

$$\sum_{c=1}^{s_2} p_{r,c} = \frac{s_2 \cdot s_0}{d}.$$

Let $t_1 = s_1 s_0 / d$ and $t_2 = s_2 s_0 / d$. It will be convenient to consider a random variable $p_{X,Y}$ which is equal to $p_{r,c}$ when $X = r$ and $Y = c$.

Now we may proceed to case (2) when $a = c$ but $b \neq d$,

$$E[M_{R_a, C_b} \cdot M_{R_a, C_d}] = \Pr[M_{R_a, C_b}, M_{R_a, C_d}].$$

For any outcome $r \in [s_1]$ such that $R_a = r$,

$$\Pr[M_{r, C_b}, M_{r, C_d}] = \sum_{u \neq v \in [s_2]} \frac{p_{r,u}}{s_0} \cdot \frac{p_{r,v}}{s_0} \cdot \Pr[C_b = u, C_d = v] = \frac{1}{s_0^2 \cdot s_2(s_2 - 1)} \sum_{u \neq v \in [s_2]} p_{r,u} \cdot p_{r,v}$$

The sum may be simplified as,

$$\sum_{u \neq v \in [s_2]} p_{r,u} \cdot p_{r,v} = \left(\sum_{u \in [s_2]} p_{r,u} \right)^2 - \sum_{u \in [s_2]} p_{r,u}^2 = t_2^2 - \sum_{u \in [s_2]} p_{r,u}^2.$$

As $f(x) = x^2$ is convex we may apply Jensen's inequality to see $t_2^2/s_2 \leq \sum_{u \in [s_2]} p_{r,u}^2$,

$$\left(\frac{t_2}{s_2} \right)^2 = \left(\frac{\sum_{u=1}^{s_2} p_{r,u}}{s_2} \right)^2 \leq \frac{\sum_{u=1}^{s_2} (p_{r,u})^2}{s_2}.$$

Putting things together we see,

$$\begin{aligned} \Pr[M_{r, C_b}, M_{r, C_d}] &\leq \frac{1}{s_0^2 \cdot s_2(s_2 - 1)} \left(t_2^2 - \frac{t_2^2}{s_2} \right) = \frac{t_2^2}{s_0^2 \cdot s_2(s_2 - 1)} \left(1 - \frac{1}{s_2} \right) \\ &= \frac{s_0^2 \cdot s_2^2}{d^2 s_0^2 \cdot s_2(s_2 - 1)} \left(\frac{s_2 - 1}{s_2} \right) = \frac{1}{d^2}. \end{aligned}$$

For this case we may conclude, $E[M_{R_a, C_b} \cdot M_{R_a, C_d}] \leq \frac{1}{d^2}$.

Moving on to case (3) where $a \neq c$ but $b = d$, here an analogous argument to that of case (2) gives

$$E[M_{R_a, C_b} \cdot M_{R_c, C_b}] \leq \frac{1}{d^2}.$$

We proceed to the final case: (4) $a \neq c$ but $b \neq d$.

$$\begin{aligned}
 & E[M_{R_a, C_b} \cdot M_{R_c, C_d}] \\
 &= \sum_{\substack{u \in [s_1] \\ u \neq v}} \sum_{\substack{v \in [s_1] \\ v \neq u}} \sum_{\substack{n \in [s_2] \\ n \neq m}} \sum_{\substack{m \in [s_2] \\ m \neq n}} \frac{p_{u,n} p_{v,m}}{s_0^2} \cdot \Pr[R_a = u, R_c = v, C_b = n, C_d = m] \\
 &= \frac{1}{s_1(s_1 - 1)s_2(s_2 - 1)} \sum_{\substack{u \in [s_1] \\ u \neq v}} \sum_{\substack{v \in [s_1] \\ v \neq u}} \sum_{\substack{n \in [s_2] \\ n \neq m}} \sum_{\substack{m \in [s_2] \\ m \neq n}} \frac{p_{u,n} p_{v,m}}{s_0^2}
 \end{aligned}$$

Using Jensen's inequality as we did previously, for any $u \neq v \in [s_1]$,

$$\sum_{\substack{n \in [s_2] \\ n \neq m}} \sum_{\substack{m \in [s_2] \\ m \neq n}} p_{u,n} p_{v,m} = \left(\sum_{n \in [s_2]} p_{u,n} \right)^2 - \sum_{n \in [s_2]} p_{u,n}^2 \leq t_2^2 \left(\frac{s_2 - 1}{s_2} \right).$$

Therefore,

$$\begin{aligned}
 E[M_{R_a, C_b} \cdot M_{R_c, C_d}] &\leq \frac{1}{s_1(s_1 - 1)s_2(s_2 - 1)} \sum_{\substack{u \in [s_1] \\ u \neq v}} \sum_{\substack{v \in [s_1] \\ v \neq u}} t_2^2 \left(\frac{s_2 - 1}{s_2 s_0^2} \right) \\
 &= \frac{s_1(s_1 - 1)(s_2 - 1)}{s_1(s_1 - 1)s_2^2(s_2 - 1)s_0^2} \cdot t_2^2 = \frac{1}{s_2^2 s_0^2} \cdot \left(\frac{s_2 s_0}{d} \right)^2 = \frac{1}{d^2}.
 \end{aligned}$$

Having bounded all terms for cases (2), (3) and (4) by $1/d^2$, we see

$$\sum_{\substack{a, c \in [k_1] \\ b, d \in [k_2] \\ a \neq c \vee b \neq d}} E[M_{R_a, C_b} \cdot M_{R_c, C_d}] \leq \frac{k_2 k_1 (k_1 - 1) + k_1 k_2 (k_2 - 1) + k_1 (k_1 - 1) k_2 (k_2 - 1)}{d^2},$$

letting $k_1 = k_2 = \sqrt{d}$ this simplifies to

$$\sum_{\substack{a, c \in [k_1] \\ b, d \in [k_2] \\ a \neq c \vee b \neq d}} E[M_{R_a, C_b} \cdot M_{R_c, C_d}] \leq \frac{2(\sqrt{d} - 1) + (\sqrt{d} - 1)^2}{d} = \frac{d - 1}{d}.$$

Combining this with our bound from case (1) we arrive at

$$\sum_{\substack{a, c \in [\sqrt{d}] \\ b, d \in [\sqrt{d}]}} E[M_{R_a, C_b} \cdot M_{R_c, C_d}] \leq \frac{\sqrt{d}^2}{d} + \frac{d - 1}{d} = 2 - \frac{1}{d},$$

which approaches 2 for increasing d .

So, we get

$$E[B_{R,C}^i] = \Pr[A_{R,C}^i > 0] \geq \frac{1}{2 - \frac{1}{d}},$$

$$E \left[\sum_{i \in [d]} B_{R,C}^i \right] \geq \frac{d}{2 - \frac{1}{d}} \approx \frac{d}{2},$$

as desired. ◀

SimpleMSRPIR [7]

Server₁(D) : Pick $i \leftarrow [d]$ and return $\text{msg}_1 = (i, D[i])$.

Server₂(D) : Choose a random mask $\delta \leftarrow [d]$. If $\delta = 0$, set $D' = \perp$. otherwise, let $p_1, \dots, p_{d/2}$ be the list of pairs of indices $p_k = (j_{k,1}, j_{k,2})$ such that $j_{k,1} \oplus j_{k,2} = \delta$ (ordered by increasing smallest value in the pair). Define D' such that $D'[k] = D[j_{k,1}] \oplus D[j_{k,2}]$. (D' has size $\frac{dw}{2}$.) Return $\text{msg}_2 = (\delta, D')$.

Client($\text{msg}_1 = (i, D[i]), \text{msg}_2 = (\delta, D')$) : If $\delta = 0$, return $(i, D[i])$. Else, find p_k such that $i \in p_k$. Return $(i \oplus \delta, D'[k] \oplus D[i])$.

■ **Figure 1** Description of SimpleMSPIR [7].

4 Bucket-RPIR

In this section, we present a construction based on splitting the database into random subsets, which we refer to as *buckets*. This construction is inspired by SimpleMSPIR of Gentry *et al.* [7], in which $\mathcal{S}_1$ sends a single random row, and $\mathcal{S}_2$ pairs the rows and sends the XOR of each pair, for total communication $\frac{dw}{2} + O(1)$. The client returns the row which $\mathcal{S}_1$'s row was paired with. We reproduce SimpleMSPIR in Figure 1.

SimpleMSPIR has communication complexity linear in the size of the database. In this section, we describe a generalization of SimpleMSPIR where, instead of pairing rows, $\mathcal{S}_2$ distributes the rows into larger buckets. In order to enable the client to retrieve a row from XORs of larger sets, $\mathcal{S}_1$ sends many random rows instead of just one. Balancing the size of $\mathcal{S}_2$'s buckets and the number of rows $\mathcal{S}_1$ sends allows us to get communication complexity sublinear in the database size. We describe our protocol in Figure 2, where p is the probability that $\mathcal{S}_1$ sends any given row, and b is the size of $\mathcal{S}_2$'s buckets.

Bucket-RPIR

Server₁(D) : Let $I = \emptyset$. For each row index $i \in [d]$, add i to I with probability p . Return $\text{msg}_1 = \{(i, D[i])\}_{i \in I}$.

Server₂(D) : Pick a random mapping $\pi : [d] \rightarrow [\frac{d}{b}]$ subject to the constraint that exactly b indices map to each value. (π can be represented in $d \log(\frac{d}{b})$ bits.) Define D' such that $D'[k] = D[i_1] \oplus \dots \oplus D[i_b]$ where $\pi[i_1] = \dots = \pi[i_b] = k$. (D' has size $\frac{dw}{b}$.) Let $\text{msg}_2 = (\pi, D')$.

Client($\text{msg}_1 = \{(i, D[i])\}_{i \in I}, \text{msg}_2 = (\pi, D')$) : Let π_k denote the set of indices $i_1, \dots, i_b$ such that $\pi[i_1] = \dots = \pi[i_b] = k$.

If there does not exist a bucket k where exactly all but one of π_k are included in msg_1 , abort.

Otherwise, with probability $\frac{|\text{msg}_1|}{d}$, output a random value from msg_1 .

With probability $1 - \frac{|\text{msg}_1|}{d}$, pick a random bucket k for which all but one of π_k are included in msg_1 . Let i be such that $\pi(i) = k$ and i is missing from msg_1 . Return $(i, D'[k] \oplus (\oplus_{j \in \pi_k \setminus \{i\}} D[j]))$.

■ **Figure 2** Description of Bucket-RPIR, Parametrized by p and b .

Next, we need to determine what p and b should be. The probability that the client is able to retrieve a row is

$$\begin{aligned} & (\text{number of buckets}) \times \Pr[\text{all but one row in some bucket is sent by } \mathcal{S}_1] \\ &= \frac{d}{b} \times b(1-p)p^{b-1} = d(1-p)p^{b-1}. \end{aligned}$$

We need to pick b and p such that this probability is greater than half. Set

$$b = \frac{\log(d)}{\log \log(d)} + 1, p = \frac{1}{d^{\frac{\log \log(d)}{\log(d)}}}.$$

Let's check that with these parameters, the probability of success is indeed greater than half:

$$\begin{aligned} \Pr[\text{successful row retrieval}] &= d(1-p)p^{b-1} \\ &= d(1-p) \left(\frac{1}{d^{\frac{\log \log(d)}{\log(d)}}} \right)^{\frac{\log(d)}{\log \log(d)} + 1 - 1} = 1 - p. \end{aligned}$$

For large enough d , p becomes smaller than half, and thus $1-p$ becomes greater than half.

Circling back to what each server sends:

$\mathcal{S}_1$ sends $pd = \frac{d}{d^{\frac{\log \log(d)}{\log(d)}}} = o(d)$ separate rows, which amounts to $o(d)w$ bits.

$\mathcal{S}_2$ sends $\frac{d}{b} = \frac{d}{\frac{\log(d)}{\log \log(d)} + 1}$ XORed buckets, as well as a description of the function π . This amounts to $o(d)w + d \log(\frac{d}{b})$ bits.

The total amount of communicated information is thus sublinear in dw .

► **Theorem 5.** *Bucket-RPIR (Figure 2) is a secure MS-RPIR protocol (Definition 1) when p and b have the values described above.*

Proof. We start with correctness. The probability that the client outputs $\perp$ is equal to the probability that there is no bucket k such that msg_1 contains all but one row from π_k . We have set this probability to be $1-p$. Next, subject to the constraint that there is such a bucket, because of the randomness of π every row i not in msg_1 is equally likely to be the row missing from bucket k . The client chooses a row in msg_1 and not in msg_1 with proportional probabilities. So, Bucket-RPIR is $(1-p)$ correct.

We argue privacy against $\mathcal{S}_1$ and $\mathcal{S}_2$ separately. $\mathcal{S}_1$ learns nothing about i because, since the client chooses a row in msg_1 and not in msg_1 with proportional probabilities, whether a row is included or excluded from msg_1 does not correlate with its probability of being chosen. If a row in msg_1 is chosen, it is chosen randomly from msg_1 ; if a row outside of msg_1 is chosen, that choice is dictated by the choice of π , where any row outside of msg_1 is equally likely to be grouped with $b-1$ rows in msg_1 .

$\mathcal{S}_2$ similarly learns nothing about i . This is because each row is included in msg_1 with the same probability, and due to the fact that π partitions the rows into buckets of equal size, each bucket is equally likely to be almost full (and each row is equally likely to be excluded from the almost-full bucket). ◀

► **Theorem 6.** *Bucket-RPIR (Figure 2) is non-trivial (Definition 3) when p and b have the values described above.*

Proof. $\mathcal{S}_1$ communicates $pd(\log(d) + w)$ bits; when $p = \frac{1}{d^{\frac{\log \log(d)}{\log(d)}}}$ and $w = \Omega(\log d)$, this is sublinear in dw . $\mathcal{S}_2$ communicates $d \log(\frac{d}{b}) + \frac{dw}{b}$ bits; when $b = \frac{\log(d)}{\log \log(d)} + 1$, this is also sublinear in dw . ◀

5 OneHot-RPIR

In this section, we present a construction based on one-hot vectors. We start by recalling the Shamir secret sharing scheme upon which we provide a high-level overview of the protocol and then offer a formal description in Figure 3.

5.1 Building Block: Secret Sharing

► **Definition 7** (*t-out-of-n secret sharing scheme*). A *t-out-of-n secret sharing scheme* is a pair of algorithms (**share**, **reconstruct**), such that:

- $(s_1, \dots, s_n) \xleftarrow{\$} \text{share}(s, n, t; \rho)$ is a randomized algorithm that, on input a secret s , outputs a set of n shares $(s_1, \dots, s_n)$ using randomness ρ and fixes the threshold t .
- $s \leftarrow \text{reconstruct}(s_{i_1}, \dots, s_{i_t})$ is a deterministic algorithm that on input t shares $(s_{i_1}, \dots, s_{i_t})$, where $(i_1, \dots, i_t) \subset [n]$, outputs the secret s .

A secret sharing scheme must satisfy the following properties:

Correctness: $\forall s, \forall T = (i_1, \dots, i_t) \subset [n], \forall \rho$ we have:

$$\Pr_{\text{share}(s, n, t; \rho) \rightarrow (s_1, \dots, s_n)} [\text{reconstruct}(s_{i_1}, \dots, s_{i_t}) = s] = 1.$$

Perfect Security: $\forall s, s', \forall I \subseteq \{1, \dots, n\}, \forall \rho$ where $|I| < n$, the following distributions are indistinguishable:

$$\{(s_i : i \in I) \mid (s_1, \dots, s_n) \leftarrow \text{share}(s, n, t; \rho)\} \\ \{(s'_i : i \in I) \mid (s'_1, \dots, s'_n) \leftarrow \text{share}(s', n, t; \rho)\}$$

We describe the Shamir secret sharing scheme [10], which we use to instantiate the construction from expendable correlated randomness.

Let $\mathbb{F}_q$ be a finite field, where q is chosen as the smallest prime greater than n ; and let $s \in \mathbb{F}_q$ be the secret we want to share. We start by picking a uniformly random polynomial $p \in \mathbb{F}_q[X]$ of degree $t - 1$, such that $p(0) = s$, and for $i \in [n]$ define the shares $s_i = p(i)$. For $i \in [n]$ we distribute $p(i)$ to the i -th party. Now, observe that given that p is a $t - 1$ degree polynomial, any set of at least t shares uniquely determines p . Upon collecting this set of at least t shares, we can reconstruct via polynomial interpolation.

For the sake of completeness, we formally describe the Shamir secret sharing scheme using the **share** and **reconstruct** algorithms:

► **Definition 8** (Shamir secret sharing scheme [10]).

- **share**($s, n, t; \rho$): Using randomness ρ , sample $a_1, \dots, a_{t-1} \xleftarrow{\rho} \mathbb{F}_q$, build the polynomial $p(x)$ as $p(x) = s + \sum_{i=1}^{t-1} a_i x^i$ and construct the shares $(s_1, \dots, s_n)$ as $s_i \leftarrow p(i)$.
- **reconstruct**($s_{i_1}, \dots, s_{i_t}$): Interpolate the polynomial $p(x)$ for each $(i_1, \dots, i_t) \subset [n]$ points corresponding to the secret shares and output $s = p(0)$.

Properties

- **Additive homomorphism:** Let s^1 and s^2 be two secrets that we secret share using polynomials $p_1, p_2 \in \mathbb{F}_q$ both of degree $t - 1$. Now, observe that in order to perform the addition of s^1 and s^2 , each party can just locally add the two of its shares - e.g. the party P_i can compute $s_i^1 + s_i^2$ which corresponds to the i -th share of the polynomial $p_1 + p_2$ (of the same degree $t - 1$) and therefore the secret $s^1 + s^2$.

- **Multiplicative homomorphism:** Let s^1 and s^2 be two secrets that we secret share using polynomials $p_1, p_2 \in \mathbb{F}_q$ of different degrees, $t_1 - 1$ and $t_2 - 1$. Now observe that if each party locally performs the multiplication of its shares, it obtains still a valid share of the polynomial $p_1 \cdot p_2$ but of degree $t_1 + t_2 - 2$. In order for the reconstruction to still naively work,³ n must be such that $t_1 + t_2 - 1 \leq n$.

5.2 Overview of the construction

Let $\mathcal{S}_1, \dots, \mathcal{S}_n$ denote the set of servers, where each server holds a copy of the database $D \in (\{0, 1\}^w)^d$.

Let $\mathbf{v}_1, \dots, \mathbf{v}_u$ represent a set of randomly generated one-hot vectors, each of length s , and let $\mathbf{v}_{i,j}$ denote the j -th component of the i -th one-hot vector. As the correlated randomness setup, we assume that each server holds a secret share of every component of every one-hot vector. More formally, server $\mathcal{S}_p$ holds a set of secret shares $\{\mathbf{v}_{i,j}^p\}_{i \in [u], j \in [s]}$, where $\mathbf{v}_{i,j}^p$ is the p -th share of the component $\mathbf{v}_{i,j}$, generated using the secret sharing algorithm $\text{share}(\mathbf{v}_{i,j}, n, t+1, \rho)$. We argue that although we heavily rely on this correlated randomness, this can easily be established in a pre-processing phase, for example, using a multi-party computation protocol.

Note that the tensor product of one-hot vectors results in another one-hot vector, but of a multiplied length (that is, the tensor product of our one-hot vectors will have length s^u). We refer to the resulting vector as the *tensored one-hot vector* $\mathbf{v}$. In the online phase, each server $\mathcal{S}_p$ computes the tensor product over its shares of the one-hot vectors. Specifically, each server computes

$$\mathbf{v}^p = \mathbf{v}_1^p \otimes \dots \otimes \mathbf{v}_u^p,$$

where for $i \in [u]$, the vector $\mathbf{v}_i^p$ is formed by stacking the shares $\{\mathbf{v}_{i,j}^p\}_{j \in [s]}$ into a single column vector.

Because Shamir sharing has a limited form of multiplicative homomorphism, by multiplying their shares locally as described above, the servers end up with shares of each component of the tensored one-hot vector. However, the degree of the sharing of the tensored one-hot vector will be the sum of the degrees of the sharings of the individual one-hot vectors. We must tune the threshold t (the degree of the polynomials used to share the individual components of the one-hot vectors) such that, after multiplication, there are still enough shares to reconstruct the vector. This can be achieved by setting the parameters t and u such that the number of servers $n > t \cdot u$.

For the final step of the protocol, each server computes the inner product of its shares of the tensored one-hot vector with the database. This corresponds to obtaining a share of the database row at the position where $\mathbf{v}^p$ is non-zero. Specifically, server $\mathcal{S}_p$ computes:

$$\text{msg}_p = \langle D, \mathbf{v}^p \rangle$$

and sends msg_p to the client. To ensure correctness, we set the size of the original one-hot vectors s such that $s^u = d$, where d is the number of rows in the database.

Upon receiving the messages $\text{msg}_1, \dots, \text{msg}_n$, the client obtains a random row of the database by computing:

$$D_* \leftarrow \text{reconstruct}(\text{msg}_1, \dots, \text{msg}_n).$$

³ To work without using techniques such as degree reduction that require additional communication.

MS-RPIR Protocol based on One-Hot Vectors

Servers $\mathcal{S}_1, \dots, \mathcal{S}_n$ each hold the database $D \in (\{0, 1\}^w)^d$.

Setup: Each server $\mathcal{S}_p$ for $p \in [n]$ holds shares of one-hot vectors $\mathbf{v}_1, \dots, \mathbf{v}_u$ of length s . These shares are computed over the individual components of the vectors as $\text{share}(\mathbf{v}_{i,j}, n, t+1, \rho)$, for $i \in [u], j \in [s]$, where $\mathbf{v}_{i,j}$ is the j -th component of the vector $\mathbf{v}_i$. Namely, each server $\mathcal{S}_p$ holds $\{\mathbf{v}_{i,j}^p\}_{i \in [u], j \in [s]}$.

Online Phase:

Server_p($D, \{\mathbf{v}_{i,j}^p\}_{i \in [u], j \in [s]}$) : Compute $\mathbf{v}^p = \mathbf{v}_1^p \otimes \dots \otimes \mathbf{v}_u^p$, where for $i \in [u]$, the vector $\mathbf{v}_i^p$ is formed by stacking the shares $\{\mathbf{v}_{i,j}^p\}_{j \in [s]}$ into a single column vector. Set $\text{msg}_p = \langle D, \mathbf{v}^p \rangle$ and send msg_p to the client.

Client($\text{msg}_1, \dots, \text{msg}_n$) : Compute $D_* \leftarrow \text{reconstruct}(\text{msg}_1, \dots, \text{msg}_n)$ and output $(*, D_*)$.^a

^a We assume that the database has the index of each row encoded inside that row.

■ **Figure 3** Description of the One-Hot Vectors Protocol.

► **Theorem 9.** Let q be a $\lceil \max \{\log(d) + w, \log(n)\} \rceil$ -bit prime and $\mathbb{F}_q$ be a finite field. Then the one-hot vectors protocol from Figure 3 is a non-trivial n -server threshold- t MS-RPIR protocol, satisfying Definition 1, with communication complexity $\text{CC}(n, w, d) = n(\log(d)w + 1)$.

The protocol is *non-trivial* (Definition 3), since we replace communication complexity with pre-processing size. During the online phase, each server sends one message of size $\lceil \log q \rceil$; since the client obtains n messages (one from each server), the communication complexity is

$$\text{CC}(n, w, d) = n \lceil \log q \rceil = n \lceil \max \{\log(d) + w, \log(n)\} \rceil.$$

The protocol also requires pre-processing of size non-trivially less than the number of rows d as long as $u > 1$, which can only happen as long as $n > 2t$.

Proof.

Correctness. Let $f_{i,j}(x)$ be the random polynomial used for the secret sharing of the j -th component of the i -th one-hot vector. After establishing the setup, the server $\mathcal{S}_p$ has shares $\{f_{i,j}(p)\}_{i \in [u], j \in [s]} = \{\mathbf{v}_{i,j}^p\}_{i \in [u], j \in [s]}$ which can be rearranged as a set of one-hot vectors $\{\mathbf{v}_1^p, \mathbf{v}_2^p, \dots, \mathbf{v}_u^p\}$, where

$$\mathbf{v}_i^p = \begin{bmatrix} \mathbf{v}_{i,1}^p \\ \mathbf{v}_{i,2}^p \\ \vdots \\ \mathbf{v}_{i,s}^p \end{bmatrix}. \quad (1)$$

Note, however, that only one of the rows corresponds to a share of 1, whereas the other rows correspond to some share of 0.

Next, each server $\mathcal{S}_p$ computes the tensor product $\mathbf{v}^p = \mathbf{v}_1^p \otimes \mathbf{v}_2^p \otimes \dots \otimes \mathbf{v}_u^p$, which more closely looks like

$$\mathbf{v}^p = \begin{bmatrix} \mathbf{v}_{1,1}^p \cdot \mathbf{v}_{2,1}^p \cdots \mathbf{v}_{u-1,1}^p \cdot \mathbf{v}_{u,1}^p \\ \mathbf{v}_{1,1}^p \cdot \mathbf{v}_{2,1}^p \cdots \mathbf{v}_{u-1,1}^p \cdot \mathbf{v}_{u,2}^p \\ \vdots \\ \mathbf{v}_{1,1}^p \cdot \mathbf{v}_{2,1}^p \cdots \mathbf{v}_{u-1,1}^p \cdot \mathbf{v}_{u,s}^p \\ \mathbf{v}_{1,1}^p \cdot \mathbf{v}_{2,1}^p \cdots \mathbf{v}_{u-1,2}^p \cdot \mathbf{v}_{u,1}^p \\ \mathbf{v}_{1,1}^p \cdot \mathbf{v}_{2,1}^p \cdots \mathbf{v}_{u-1,2}^p \cdot \mathbf{v}_{u,2}^p \\ \vdots \\ \mathbf{v}_{1,1}^p \cdot \mathbf{v}_{2,1}^p \cdots \mathbf{v}_{u-1,2}^p \cdot \mathbf{v}_{u,s}^p \\ \vdots \\ \vdots \\ \vdots \\ \mathbf{v}_{1,s}^p \cdot \mathbf{v}_{2,s}^p \cdots \mathbf{v}_{u-1,s}^p \cdot \mathbf{v}_{u,1}^p \\ \mathbf{v}_{1,s}^p \cdot \mathbf{v}_{2,s}^p \cdots \mathbf{v}_{u-1,s}^p \cdot \mathbf{v}_{u,2}^p \\ \vdots \\ \mathbf{v}_{1,s}^p \cdot \mathbf{v}_{2,s}^p \cdots \mathbf{v}_{u-1,s}^p \cdot \mathbf{v}_{u,s}^p \end{bmatrix} \quad (2)$$

Lastly, the server computes the message $\mathbf{msg}_p$ as the inner product $\langle \mathbf{v}^p, D \rangle$. Let the indices of the database range from 0 to $d-1$. Then, we can rewrite

$$\mathbf{msg}_p = \langle D, \mathbf{v}^p \rangle = \sum_{s_1, \dots, s_u \in [s]} D[\sum_{i \in [u]} (s_i - 1) \cdot s^{u-i}] \prod_{j \in [u]} \mathbf{v}_{j, s_j}^p.$$

Notice that this expression is a summation of s^u values where each of the values is a database row (some constant in $\mathbb{F}_t$), multiplied by a product of exactly u shares. Since every share corresponds to a polynomial of degree t , the product of u shares corresponds to a secret share of a polynomial of degree $t \cdot u$. As the client obtains $n > t \cdot u$ shares, it can successfully interpolate the polynomial in all n points.

Observe again that since $\mathbf{v}^p$ is a tensor product of u one-hot vectors, only one row is a share of 1 while the others are shares of 0. With that follows the correctness of the client's output.

The argument that any row in the database is obtained with the same probability $\frac{1}{d}$ follows directly from the randomness of one-hot vectors $\mathbf{v}_1, \dots, \mathbf{v}_u$, as the position of the non-zero row in each of the vectors is picked uniformly at random results in the position of the non-zero row in the vector $\mathbf{v}$, obtained by the tensor product $\mathbf{v} = \mathbf{v}_1 \otimes \dots \otimes \mathbf{v}_u$ also being uniformly random.

Client Privacy. To argue for client privacy, we need to show that the distribution of the index output by the client, along with the view of up to threshold- t colluding servers, is indistinguishable from the distribution where the index is sampled uniformly at random. This follows from the randomness of the one-hot vectors: the index of the row received by the client corresponds to the position of the non-zero entry in the one-hot vector $\mathbf{v}$, which is computed as the tensor product $\mathbf{v} = \mathbf{v}_1 \otimes \dots \otimes \mathbf{v}_u$. As in the correctness argument, each vector $\mathbf{v}_1, \dots, \mathbf{v}_u$ has its non-zero position chosen uniformly at random. Consequently, the non-zero position in $\mathbf{v}$ is also uniformly random, ensuring that the client's obtained index remains uniformly distributed.

The view of up to threshold- t colluding servers is exactly t shares of each vector $\mathbf{v}_1, \dots, \mathbf{v}_u$. As we use the Shamir sharing scheme, which has perfect security, this set of shares reveals no information about the non-zero position in vectors and the index obtained by the client. Therefore, the two distributions are indistinguishable. $\blacktriangleleft$

References

- 1 Andris Ambainis. Upper bound on communication complexity of private information retrieval. In Pierpaolo Degano, Roberto Gorrieri, and Alberto Marchetti-Spaccamela, editors, *ICALP 97*, volume 1256 of *LNCS*, pages 401–407. Springer, Heidelberg, July 1997. doi:10.1007/3-540-63165-8_196.
- 2 Amos Beimel and Yuval Ishai. Information-theoretic private information retrieval: A unified construction. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *ICALP 2001*, volume 2076 of *LNCS*, pages 912–926. Springer, Heidelberg, July 2001. doi:10.1007/3-540-48224-5_74.
- 3 Yeow Meng Chee, Tao Feng, San Ling, Huaxiong Wang, and Liang Feng Zhang. Query-efficient locally decodable codes of subexponential length, 2010. arXiv:1008.1617.
- 4 Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th FOCS*, pages 41–50. IEEE Computer Society Press, October 1995. doi:10.1109/SFCS.1995.492461.
- 5 Zeev Dvir and Sivakanth Gopi. 2-server PIR with sub-polynomial communication. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 577–584. ACM Press, June 2015. doi:10.1145/2746539.2746546.
- 6 Klim Efremenko. 3-query locally decodable codes of subexponential length. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 39–44. ACM Press, May / June 2009. doi:10.1145/1536414.1536422.
- 7 Craig Gentry, Shai Halevi, Bernardo Magri, Jesper Buus Nielsen, and Sophia Yakoubov. Random-index PIR and applications. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part III*, volume 13044 of *LNCS*, pages 32–61. Springer, Heidelberg, November 2021. doi:10.1007/978-3-030-90456-2_2.
- 8 Fatemeh Ghasemi, Swastik Kopparty, and Madhu Sudan. Improved pir schemes using matching vectors and derivatives, 2024. doi:10.48550/arXiv.2411.11611.
- 9 Toshiya ITOH and Yasuhiro SUZUKI. Improved constructions for query-efficient locally decodable codes of subexponential length. *IEICE Transactions on Information and Systems*, E93-D(2):263–270, 2010. doi:10.1587/transinf.e93.d.263.
- 10 Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979. doi:10.1145/359168.359176.
- 11 Stephanie Wehner and Ronald de Wolf. Improved lower bounds for locally decodable codes and private information retrieval. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP 2005*, volume 3580 of *LNCS*, pages 1424–1436. Springer, Heidelberg, July 2005. doi:10.1007/11523468_115.

MetaDORAM: Info-Theoretic Distributed ORAM with Less Communication

Brett Hemenway Falk   

University of Pennsylvania, Philadelphia, PA, USA

Daniel Noble¹   

Silence Laboratories, Singapore

Rafail Ostrovsky   

UCLA, CA, USA

Abstract

A Distributed Oblivious RAM is a multi-party protocol that securely implements a RAM functionality on secret-shared inputs and outputs. This paper presents two information-theoretically secure DORAMs whose communication costs are asymptotic improvements over the state of the art. Let n be the number of memory locations and let d be the bit-length of each location.

The first, MetaDORAM1, is *statistically* secure, with $n^{-\omega(1)}$ leakage. It has amortized $O(\log_b(n)d + b\omega(1)\log(n) + \log^3(n)/\log(\log(n)))$ bits of communication per memory access. Here, $b \geq 2$ is a free parameter and $\omega(1)$ is any super-constant function (in n). The most communication-efficient prior statistically secure DORAM was that of Abraham et al (PKC 2017), which has cost $O(\log_b(n)d + b\omega(1)\log_b(n)\log^2(n))$. MetaDORAM1 is a $\Theta(\omega(1)\log(\log(n)))$ -factor improvement over the work of Abraham et al whenever $d = O(\log^2(n))$.

The second protocol, MetaDORAM2, achieves *perfect* security. It has amortized communication cost $O(\log_b(n)d + b\log(n) + \log^3(n)/\log(\log(n)))$ where, again, $b \geq 2$ is a free parameter. The best prior perfectly secure DORAM is that of Chan et al (ASIACRYPT 2018) which has communication cost $O(\log(n)d + \log^3(n))$. MetaDORAM2 is therefore a $\Omega(\log(\log(n)))$ -factor improvement over the DORAM of Chan et al under any parameter range (by setting $b = \log(n)$) and is a $\Theta(\log(n))$ -factor improvement for $d = \Omega(n^\epsilon)$ for any constant $\epsilon > 0$ (by setting $b = d/\log(n)$). Our work is the first perfectly secure DORAM with sub-logarithmic communication overhead. MetaDORAM2 comes at the cost of a once-off (for any given n) setup phase which requires exponential (in n) computation.

Both DORAMs are in the 3-party setting with security against 1 semi-honest, static corruption. By a trivial transformation, these can be transformed, respectively, into statistically and perfectly secure active 3-server ORAM protocols secure against 1 corrupt server, with the same communication costs. These multi-server ORAM protocols are likewise asymptotic improvements over the state of the art.

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques; Theory of computation → Communication complexity; Theory of computation → Cryptographic primitives; Security and privacy → Management and querying of encrypted data

Keywords and phrases ORAM, MPC, DORAM, multi-server ORAM, active ORAM

Digital Object Identifier 10.4230/LIPIcs.ITC.2025.6

Related Version *Full Version:* <https://eprint.iacr.org/2024/011.pdf> [32]

Funding This research was supported in part by DARPA under Cooperative Agreement HR0011-20-2-0025, the Algorand Centers of Excellence programme managed by Algorand Foundation, NSF grants CNS-2246355, CCF-2220450 and CNS-2001096, US-Israel BSF grant 2022370, Amazon Faculty Award, Cisco Research Award, Sunday Group, ONR grant (N00014-15-1-2750) “SynCrypt: Automated Synthesis of Cryptographic Constructions” and a gift from Ripple Labs, Inc. Any views, opinions, findings, conclusions or recommendations contained herein are those of the author(s) and

¹ Work done while at University of Pennsylvania.



© Brett Hemenway Falk, Daniel Noble, and Rafail Ostrovsky;
licensed under Creative Commons License CC-BY 4.0

6th Conference on Information-Theoretic Cryptography (ITC 2025).

Editor: Niv Gilboa; Article No. 6; pp. 6:1–6:23



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, the Department of Defense, the Algorand Foundation, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes, notwithstanding any copyright annotation therein.

Acknowledgements Daniel Noble would also like to acknowledge God for supporting him in this research.

1 Introduction

Generic secure Multi-Party Computation (MPC) has historically been designed in the *circuit model* [40, 20, 6], in which the computation is represented as a circuit on bits or group elements. However, many computations are more naturally and efficiently implemented in the *RAM model*, in which it is possible to access locations in memory according to index variables. For instance, RAM is assumed in many classic algorithms and data structures, such as implementations of dictionaries, pointers, graphs and priority queues. An efficient implementation of a RAM functionality for MPC would therefore enable the adoption of generic efficient MPC.

Distributed Oblivious RAM (DORAM) is a functionality that implements RAM for MPC. It stores n d -bit blocks of data in a secret-shared memory, and allows accesses to that memory (reads or writes) at secret-shared locations. A DORAM is secure if the views of the parties can be efficiently simulated without knowledge of any private values. See section 5 for a complete definition of the DORAM functionality.

In this work we construct information-theoretically secure DORAMs in the 3-party honest-majority setting. Our results show new asymptotic results for DORAM as a category of problem, but are also significant because the 3-party honest-majority setting is prevalent in both research and practice. An honest majority is necessary to achieve information-theoretically secure generic MPC [6], so the 3-party honest-majority setting is the minimal setting for achieving generic information-theoretic DORAM. Furthermore, there are certain techniques which are extremely efficient in the 3-party honest majority setting (e.g. [13, 29]). Due to these factors, the 3-party honest-majority setting has received particular attention from both academia [16, 2, 18, 24, 3] and industry [7, 23].

DORAM is closely related to the problem of Oblivious RAM [21], which solves the problem of a client outsourcing memory to an untrusted server (or servers). In particular a w -party DORAM can be converted into an ORAM with w active (i.e. computation-performing) servers and vice-versa, usually without increase in the communication cost. See section 2 for more details. Therefore, efficient DORAM is intrinsically tied to efficient multi-server active ORAM. A primary metric of this efficiency is the total amount of communication per memory access. This is often measured as the *overhead*, that is the number of blocks (of size d) of communication required per memory access.

Our Contribution. In this work, we present two novel DORAM protocols for the 3-party semi-honest honest-majority setting. The first, MetaDORAM1, achieves *statistical* security. That is, the statistical distance between the adversary's view and a simulated view is negligible in n . Unlike most statistically secure DORAMs, our protocol's security does not deteriorate with the number of accesses to the DORAM; the leakage remains negligible in n even as the number of accesses tends to infinity. The statistically secure protocol achieves amortized communication cost $\Theta(\log_b(n)d + b\omega(1)\log(n) + \log^3(n)/\log(\log(n)))$. Here $b \geq 2$ is a free parameter. The best prior work is that of Abraham et al (PKC 2017) [1] which has communication cost $\Theta(\log_b(n)d + b\omega(1)\log_b(n)\log^2(n))$. MetaDORAM1 and

Abraham et al [1] are asymptotically equivalent when $d = \omega(\log^{2+\epsilon}(n))$. However, when $d = O(\log^2(n))$, the communication cost of MetaDORAM1 is $\Theta(\log^3(n)/\log(\log(n)))$ (e.g. by setting $b = \log(n)/\log(\log(n))$) whereas the work of Abraham et al has communication cost $\Theta(\omega(1)\log^3(n))$.

Our second DORAM protocol, MetaDORAM2 achieves *perfect* security. That is, the adversary's view is chosen from an identical distribution as the simulated view. MetaDORAM2 requires communication cost $\Theta(\log_b(n)d + b\log(n) + \log^3(n)/\log(\log(n)))$. The best prior work by Chan et al. (ASIACRYPT 2018) [10] has communication cost $\Theta(\log(n)d + \log^3(n))$. MetaDORAM2 is an asymptotic improvement over all parameter ranges. When $b = \log(n)$, MetaDORAM2 has communication cost $\Theta(\log(n)d/\log(\log(n)) + \log^3(n)/\log(\log(n)))$, that is a $\log(\log(n))$ -factor improvement over Chan et al for all parameter ranges. Additionally, the free parameter b allows MetaDORAM2 to have improved performance for large d . Whenever $d = \Omega(n^\epsilon)$ for some constant $\epsilon > 0$, setting $b = d/\log(n)$ yields a protocol with communication cost $\Theta(d)$, that is the overhead is constant, which is a $\Theta(\log(n))$ -factor improvement over the DORAM of Chan et al. MetaDORAM2 is the first perfectly secure DORAM protocol with sub-logarithmic overhead over any parameter range.

MetaDORAM2's perfect security comes at a cost in setup computation. MetaDORAM2 requires hash tables with well chosen hash functions. These hash functions must be chosen such that whichever m inputs are chosen (from a universe of size $2n$), it must be possible to build the hash table. Using $\Theta(\log(n))$ hash functions that map to disjoint spaces, causes this to occur with high probability [41]. The problem is that it is difficult to *verify* that a given choice of hash functions satisfies this property. As such, MetaDORAM2 achieves perfect security by first verifying that all subsets of size m from $\{1, \dots, 2n\}$ can be successfully built into a hash tables of size $\Theta(m)$ using the chosen hash functions. Each verification takes $\text{poly}(n)$ time, but there are $\binom{2n}{m} < 2^{n \log(n)}$ subsets to verify. While this computation need only occur once for any given value of n , this limits the feasibility of this construction in practice to tiny n . Note that the setup phase, while requiring exponential *computation*, does not require any *communication*, so does not affect the amortized communication cost of the protocol. Therefore, MetaDORAM2 shows for the first time that the problem of perfectly-secure DORAM has sub-logarithmic communication overhead.

Note that in the non-uniform model of computation, the expensive setup phase can be avoided. The hash functions can simply be provided in the advice string, as they do not depend on the access pattern, but only on the size of the DORAM, n .

Due to the conversion between DORAMs and ORAMs, MetaDORAM1 (resp. MetaDORAM2) can be converted to a 3-server active ORAMs that is statistically (resp. perfectly) secure and has communication cost $\Theta(\log_b(n)d + b\omega(1)\log(n) + \log^3(n)/\log(\log(n)))$ (resp. $\Theta(\log_b(n)d + b\log(n) + \log^3(n)/\log(\log(n)))$).

It has been proven that an ORAM must access $\Omega(\log(n)d)$ bits of physical memory for each d -bit virtual memory access [21, 27]. (This lower bound is normally phrased as $\Omega(\log(n))$ *overhead*, that is $\Omega(\log(n))$ blocks each of size d bits.) This bound applies even if there are multiple servers [28, 25]. If the servers are passive (non-active) this implies a $\Omega(\log(n)d)$ bound on communication. In an active ORAM, the $\Omega(\log(n)d)$ bound applies to the number of bits accessed, but need not apply to the amount of communication. This work, like [1], achieves sub-logarithmic communication overhead in the information-theoretic setting. This result is slightly surprising: it means that server-side computation is useful at reducing the asymptotic communication complexity even without the introduction of computational assumptions. Our result, taken with [1], shows that the asymptotic bounds on the DORAM problem are, as of yet, not well understood, and opens up many interesting questions regarding what lower bounds exist for DORAMs, as well as for active information-theoretic ORAMs in general (see section 7).

MetaDORAM1 and MetaDORAM2 also have lower communication overhead than most DORAM protocols that use computational assumptions. See Table 2 in section 2 for more details. MetaDORAM1 and MetaDORAM2 also have reasonable performance in other metrics. The computation and local memory access overheads are poly-logarithmic. The persistent memory usage is $\log_b(n)d \leq \log(n)d$ bits. The round complexity per query is $O(\log(n)\log(\log(n)))$. Throughout, the hidden constants in our asymptotic notation are small: most are single-digit and all are less than 20.

Organization. Our paper is organized as follows. Section 2 provides a short history of prior ORAM and DORAM protocols. Section 3 provides a technical overview of our results and techniques. Section 4 explains the notation used, in particular the various types of secret-sharing used and how they are represented. The formal DORAM functionality is presented in section 5, as well as the functionalities for Secret-Shared Private Information Retrieval (SSPIR) and secure routing, which are used by our DORAM protocol. Section 6 presents our full DORAM protocol, and analyzes its security and communication costs. Section 7 concludes by discussing some interesting open questions. The functionalities for SSPIR and secure routing can be implemented using standard techniques. For completeness, these are presented explicitly in appendices A and B.

2 Prior Works

Many DORAMs are constructed from instances of a related, well-studied, primitive called Oblivious RAM (ORAM). ORAMs were first identified by Goldreich and Ostrovsky in the 1980s [19, 33]. Consider a program which is running in a secure environment with very limited memory. The program wishes to make use of general memory on a device, but an adversary may be able to observe access patterns on this device. For instance, the program may be running in a secure enclave, such as Intel SGX [12], and may need to protect sensitive information even if the operating system is corrupted. The program can encrypt the data; this will hide the data’s contents, but will not hide the memory locations accessed by the program, which might leak sensitive information. An ORAM is an intermediary between the program and the main memory. It provides a RAM functionality to the program using the device’s memory in such a way that the access pattern on the device (the *physical* access pattern) reveals no information about the memory accesses by the program (the *virtual* access pattern), except for the number of accesses.

In a normal RAM, the cost of retrieving a block of data from memory is equal to the size of the block, denoted d . Adding obliviousness comes at a price; the *overhead* is the multiplicative increase in the number of bits that need to be accessed relative to a normal RAM. Goldreich initially presented an ORAM that had $O(\sqrt{n}\log(n))$ overhead, where n is the number of blocks of memory [19], Ostrovsky constructed the first solution with poly-logarithmic overhead of $O(\log^3(n))$ [33, 34] (the so-called “Hierarchical” solution). A series of improvements reduced the overhead to $O(\log^2(n))$ [38, 22], $O(\log^2(n)/\log(\log(n)))$ [26, 9], $O(\log(n)\log(\log(n)))$ [36] and finally $O(\log(n))$ [4, 5] (i.e. accessing $O(\log(n)d)$ bits of memory). This final result matches the proven asymptotic lower bound of $\Omega(\log(n))$ [21, 27, 28, 25, 37]. These ORAMs consider the setting where the untrusted memory is passive (i.e. performs no computation on behalf of the ORAM) and the ORAM only has enough memory to store $\Theta(1)$ blocks (and $\Theta(1)$ κ -bit PRF keys).

ORAM was quickly recognized as a solution to another challenge: outsourcing memory in a network such as the internet. Here, a client with limited memory capacity wishes to store data on an untrusted server such that the access pattern on the server’s memory leaks

no information about the actual memory access pattern by the client. The formalism of the original ORAM use-case is immediately applicable, with the client taking the place of the secure environment and the server replacing the device memory. However, this new application led several works to consider extensions of the model. First, the server is likely to also have significant computational resources, so may be able to perform computation to reduce communication overhead, a variant referred to as *active ORAM* [15, 11]. Second, some works observed that the client could easily interact with multiple servers, which under appropriate conditions could be assumed to not collude [31, 1, 10]. This was referred to as *multi-server ORAM*.

ORAM can be used to construct Distributed Oblivious RAM (DORAM). Any client-server ORAM can be transformed into a DORAM by evaluating the client's circuit inside of a secure computation and allowing one of the parties to act as a server, as was first observed in [35]. There is now no trusted client. Instead, there is a "virtual client" that is simulated by a secure computation. Furthermore, the extensions to the ORAM model that resulted from the memory outsourcing application are immediately relevant to DORAMs. Since DORAMs already have multiple non-colluding parties, they can trivially take advantage of multi-server ORAMs by having the parties act as different servers. Since the parties already perform computation as part of the MPC protocol, they can naturally perform the computation needed by servers in an active ORAM. In general, a s -server active ORAM tolerating t corrupt parties can be transformed into a (w, t) -secure DORAM protocol for any $w \geq s$ by simulating the client in a (w, t) -secure MPC and having each server's role taken by a distinct party. Note that it is also possible to convert a *DORAM* into a multi-server active ORAM by a trivial transformation. Any (w, t) -secure DORAM can also be converted to a (w, t) -secure multi-server active ORAM by each server acting as one party, and by the client initially secret-sharing their query to the servers, and the servers sending the client the shares to reconstruct the result.

While simulating an ORAM client in a MPC protocol is always possible, doing so may increase the communication costs of the protocol, or increase the number of parties required. If information-theoretic security is required, the MPC requires an honest majority so at least 3 parties are required even if the ORAM only uses 1 or 2 servers. Furthermore, any computation which was performed locally by the client, requiring no communication, must instead be evaluated within a MPC protocol. Depending on the computational complexity of the client, the communication cost of simulating the client may increase the total asymptotic communication costs. (In contrast, converting a DORAM to a multi-server ORAM will never increase the asymptotic communication, as it only requires 1 secret-sharing and 1 reconstruction.)

Table 1 presents the communication cost of the best prior information-theoretic secure DORAMs and of the DORAMs presented in this work. Note that while MetaDORAM2 requires a setup phase with exponential computation, this does not affect its communication complexity. Abraham et al. created an efficient 2-server ORAM using PIR [1]. Their ORAM is Tree-based, in which the number of children of each vertex is a configurable parameter, $b \geq 2$. This protocol achieved a communication cost of $\Theta(\log_b(n)d + b\omega(1)\log_b(n)\log^2(n))$. The parameter b should be set to reduce the amortized cost. For $d \geq 2\omega(1)\log^2(n)$ the cost is minimized by setting $b = \frac{d}{\omega(1)\log^2(n)}$, which results in a cost $\frac{\log(n)}{\log(d)}d$. For smaller d , the cost is minimal when $b = 2$. Note that while Abraham et al. constructed a 2-server ORAM, converting it to a DORAM while maintaining information-theoretic security requires a third party, to help simulate the virtual client. Chan et al. [10] created an efficient 3-server passive ORAM scheme with perfect security. Their DORAM is Hierarchical, but unlike most Hierarchical ORAMs which use PRFs, Chan et al. store items using truly random position

■ **Table 1** Comparison with prior state of the art. Note that while the ORAM of Abraham et al. requires only 2 servers, converting it to an information-theoretic DORAM requires an honest-majority MPC protocol, therefore necessitating at least 3 parties. For statistically secure protocols, $\omega(1)$ is a super-constant function in n and the statistical leakage is $2^{-\omega(\log(n))}$.

Protocol	Amortized Communication Cost (bits)	Security
Abraham et al. [1]	$\Theta(\log_b(n)d + b\omega(1)\log_b(n)\log^2(n))$	Statistical
Chan et al. [10]	$\Theta(\log(n)d + \log^3(n))$	Perfect
MetaDORAM1 (this work)	$\Theta\left(\log_b(n)d + b\omega(1)\log(n) + \frac{\log^3(n)}{\log(\log(n))}\right)$	Statistical
MetaDORAM2 (this work)	$\Theta\left(\log_b(n)d + b\log(n) + \frac{\log^3(n)}{\log(\log(n))}\right)$	Perfect

■ **Table 2** Complexity of select DORAM protocols. $\kappa = \omega(\log(n))$ is a cryptographic security parameter. $\omega(1)$ is any super-constant function in n and the statistical leakage is $2^{-\omega(\log(n))}$.

DORAM Protocol	Amortized Communication Cost (bits)	Security
Faber et al. [16]	$O(\omega(1)\log^2(n)d + \kappa\omega(1)\log^4(n))$	Computational
Jarecki and Wei [24]	$O(\log(n)d + \kappa\log^3(n))$	Computational
Bunn et al. [8]	$O((d + \kappa)\sqrt{n})$	Computational
Lu et al. Falk et al. [31, 17]	$O(\log(n)d + \kappa\log(n))$	Computational
DuORAM [39]	$O(\kappa \cdot d \cdot \log n)$	Computational
MetaDORAM1 (this work)	$\Theta\left(\log_b(n)d + b\omega(1)\log(n) + \frac{\log^3(n)}{\log(\log(n))}\right)$	Statistical
MetaDORAM2 (this work)	$\Theta\left(\log_b(n)d + b\log(n) + \frac{\log^3(n)}{\log(\log(n))}\right)$	Perfect

labels. These are stored in a position label ORAM, which is implemented recursively. The communication cost is $\Theta(\log(n)d + \log^3(n))$. Both prior works are multi-server ORAMs with simple clients, which can be converted to 3-party honest-majority DORAMs without increasing the amortized asymptotic communication cost.

Several works also investigated building Distributed Oblivious RAMs directly without simulating a client-server ORAM. While these works generally did not achieve the same asymptotic efficiency as [1] and [10], many had good concrete efficiency. They took advantage of the existence of multiple non-colluding servers by using Distributed Point Functions [8, 39], secret-shared PIR (SSPIR) [24] and secure shuffles/routing [17]. See Table 2 for details. These works all depended on computational assumptions. In comparison, the MetaDORAM protocols are information-theoretically secure. The MetaDORAM protocols also have strictly better asymptotic communication cost than these protocols over all parameter ranges, except for Lu et al. and Falk et al. [31, 17], which can have a lower asymptotic communication cost, and that only when $\kappa + d = o(\log^2(n)/\log(\log(n)))$.

3 Technical Overview

In this section we provide a broad overview of how the MetaDORAM1 and MetaDORAM2 protocols work. These protocols are, in fact, almost identical and only differ in how hash functions are chosen. Therefore, in this overview, we only discuss the common framework of the solution, which we refer to simply as MetaDORAM. Section 6 later presents MetaDORAM1 and MetaDORAM2 in full detail, including the different approaches used to select hash functions.

At a very high level, MetaDORAM uses Secret-Shared Private Information Retrieval to access items, and always writes the accessed item (whether modified or not) to a pre-determined new location. It uses an oblivious “metadata map” that maintains a mapping from indices to their locations. This is accessed in order to obtain shares of the location to input to the Secret-Shared PIR.

A simple way to instantiate this is to simply store the original secret-shared array in memory, and to always write new items to the next free location in memory. The metadata map would then consist only of the last access time for each index. If the index has been accessed, this uniquely determines the item’s current position; if the index has never been accessed, the position is determined by the index itself. This protocol is simple, but requires performing SSPIR over an array of size $\Omega(n)$.

In order to perform SSPIR over small arrays, we instead store items in hash tables, each with $h \in \Omega(\log(n))$ hash functions. As before, items are written to the next free location in memory, but when there are c such items, these too are built into a hash table. To limit the number of hash tables, when there are b tables of a given size, the contents of these tables are extracted and rebuilt into a single table. Let $T_{1,i}, \dots, T_{b,i}$ be the tables of the i^{th} smallest size. Furthermore, every n accesses, there is a *DORAM refresh* in which the contents of all tables are extracted and rebuilt into a single hash table. As a result, there are only ever at most $O(\log_b(n)b)$ tables, so the SSPIR need only be performed over arrays of size $O(c + b \log_b(n)h) = \text{poly}(\log(n))$.

Restricting the number of locations accessed improves efficiency, but introduces several security and logistical challenges. Let $G(x)$ be a function returning a 3-D array, such that $G(x)_{i,u,k}$ is the k^{th} location accessed in table $T_{i,u}$ when querying x .

1. The first challenge is that $G(x)$ must leak no information about x (even to a computationally unbounded adversary). This is essential, because the parties who access data during a read will inevitably learn $G(x)$ by the locations they access.
2. Additionally, each time x is accessed, $G(x)$ must be a new 3-D array, independent of the previous value(s) of $G(x)$, to avoid leaking that repeated queries occurred. This implies $G(\cdot)$ must be stateful.
3. The parties need an efficient protocol for building hash tables which does not leak any correlation between indexes and the locations in which they are stored.
4. The SSPIR-index depends not only on the index and the last access time (as in the $\Omega(n)$ -sized SSPIR solution presented above) but also on *which* of the h hash functions is being used to store the item, which was determined during the hash table build.

It is standard when a hash table satisfies properties (1) and (3) above, that it is called an *oblivious hash table*. Therefore, throughout all of the hash tables in this work are oblivious hash tables (OHTables).

We solve these by separating the required properties of stateful-updates, randomness and collision-resistance. We set $G(x)_{i,u,k} = H_{i,k}(\text{Rand}(\text{UniqueNonce}(x)))$, where:

- *UniqueNonce* is a stateful function. This function returns a different value for each x . Furthermore, each time x is queried by the DORAM, the value of $\text{UniqueNonce}(x)$ is updated. This is implemented simply by defining $\text{UniqueNonce}(x) = x$ for every x which has not been accessed (since the last DORAM refresh) and defining $\text{UniqueNonce}(x) = n + \text{LastAccess}(x)$ for each x which has been accessed. (Here $\text{LastAccess}(x) \in \{1, \dots, n\}$ is the last time x was accessed, measured from the last DORAM refresh.)
- *Rand* is a random permutation on $\{1, \dots, 2n\}$.
- $H_{i,k}$ are public, fixed hash functions. They are collision-resistant, but do not need to have any other security properties.

We define $Rand(UniqueNonce(x))$ as the rune (Random Unique NoncE) of x (also written $rune(x)$). Note that when x is updated, it will be assigned a new rune.

This solution simultaneously resolves challenges (1) and (2) presented above. $rune(x)$ will be information-theoretic independent of x , so $G(x)$ will be as well, solving (1). Since $UniqueNonce(x)$ will return a distinct value each time an index is queried (between each DORAM refresh), and $Rand$ is a permutation, $rune(x)$ will also be a distinct value chosen uniformly at random (without replacement) from $\{1, \dots, 2n\}$ for each queried x . Therefore, the locations distribution will be independent of the access pattern, and in particular independent of whether indexes are repeated in queries.

To solve challenge (3), we distinguish the roles of the parties. One party, P_0 , will be the *Builder*. The other two parties, P_1 and P_2 , will be *Holders*. The Holders will store the tables and will learn $rune(x)$ when x is queried, allowing them to compute $G(x)$. The Builder picks and stores $Rand$. Observe that when building a new table, the items which are being built have been accessed before, and so $UniqueNonce(x)$ depends only on the last access time of x . We write items to secret-shared memory each time they are accessed, according to a fixed schedule. Let $Mem(x)$ represent the location in secret-shared memory of item with index x at a given point in time. The mapping from $Mem(x)$ to $LastAccess(x)$ is therefore public, as is the mapping from $LastAccess(x)$ to $UniqueNonce(x)$. The Builder, knowing $Rand(\cdot)$ therefore also knows the mapping from $Mem(x)$ to $Rand(UniqueNonce(x)) = rune(x)$ *without knowing x* . The Builder can therefore locally compute $H_{i,k}(rune(x))$ and locally determines how the memory must be permuted in order to build the oblivious hash table. The Builder then provides this permutation to a secure routing protocol, resulting in a very efficient protocol for constructing oblivious hash tables.

Meanwhile the mapping from x to $rune(x)$ is stored in a smaller DORAM, which we refer to as the sub-DORAM, which is implemented recursively. During an access, this is queried and $rune(x)$ is provided to the Holders, who can use this to calculate $G(x)$, and determine which locations in the oblivious hash tables are to be used for the SSPIR. Note that the Builder knows $Rand$, but does not learn the runes obtained during queries, or the memory locations accessed by the Holders in the tables, so does not learn any information about the access pattern.

We turn to challenge (4). Note that the table, $T_{i,u}$, in which an item is stored is fully determined by the item's last access time. However, to perform SSPIR, we also need to obtain a secret-sharing of which of the h hash functions was used to store the item in this table. Observe that when the Builder is determining the permutation for building an oblivious hash table, this *does not depend* on the access pattern, but only depends on $Rand(\cdot)$. Furthermore, the Builder picks $Rand(\cdot)$ at the start of each DORAM refresh. Therefore, the Builder is able to pre-determine the permutations it will use, and therefore which hash function will be used for each rune in each oblivious hash table. This may be surprising, since the mapping from runes to indexes has not even been determined at this point. This is possible because, regardless of which index is accessed at time t , the item will be stored in the same initial location in memory, will be assigned the same unique nonce, the same rune, and will be placed in the same locations in each oblivious hash table. This means that the Builder can secret-share a *position schedule* which determines the SSPIR-index of the rune at each point in time (until the next DORAM refresh).

A final challenge is that SSPIR assumes that the 2 Holders have identical copies of the tables (rather than a secret-sharing of the tables). To provide privacy from the Holders, we therefore mask each item in each table with a rune-dependent information-theoretic mask (one-time pad). During each table rebuild, new masks are applied. This is achieved by the

Builder adding (as secret-shares) the masks for the new table, prior to the secure routing protocol. The Builder pre-determines all new masks at the start of each DORAM refresh, and secret-shares a mask schedule among the Holders, akin to the position schedule.

The metadata mapping therefore consists of mappings from indices to runes, from runes to a position schedule and from runes to a mask schedule. The position schedule and mask schedule are simply secret-shared arrays (with the runes public and the schedules secret-shared). The mapping from indices to runes is implemented using a sub-DORAM, which is implemented recursively.

Novel contributions. In addition to providing the communication-efficient DORAM with information-theoretic security, our paper introduces a number of new techniques. Specifically, we use time-stamping and novel data structures to obtain the precise location of data blocks, we make asymmetric use of the participating compute servers to allow efficient and oblivious construction and querying of these data structures, we use as a subroutine tiny-size PIR protocols where the “databases” are constructed on the fly during query execution, and we show a novel strategy for DORAM that bridges techniques from different ORAM strategies in conjunction with ideas explained above.

4 Preliminaries

We use lower-case Latin characters to represent parameters in the protocol. n is the size of the RAM, d is the bit-length of each item. h represents the number of hash functions used by the hash tables. For an explicit integer or integral parameter, a , $[1, a]$ denotes the set of integers $\{1, \dots, a\}$. We use upper-case Latin characters to represent arrays and matrices, which are indexed using standard subscript notation. $\lg$ represents the base-2 log. For asymptotic annotation, any constant base is equivalent, in which case $\log$ represents some arbitrary constant-base log.

We denote the 3 parties as P_0 , P_1 and P_2 . P_0 is the Builder. P_1 and P_2 are the Holders. The Adversary $\mathcal{A}$, is able to corrupt at most one of the parties. The corruption is semi-honest (passive), that is the corrupted party will still follow the protocol, but $\mathcal{A}$ is able to view all data visible to the corrupted party. The corruption is static, that is $\mathcal{A}$ cannot change which party is corrupted during the protocol. Our protocols are information-theoretically secure, that is $\mathcal{A}$ may perform an arbitrarily large amount of computation.

We utilize hash functions. Our hash functions are fixed and public. The hash functions implicitly map to ranges of different sizes (depending on the size of the OHTable). In this cases, the hash functions are calculated modulo the required range.

■ **Table 3** Types of Secret-Sharing with Notation.

Sharing type	Notation	Party Share			Construction
		P_0	P_1	P_2	
3RSS (Replicated)	$[x]$	(x_0, x_1)	(x_1, x_2)	(x_2, x_0)	$x_0 \oplus x_1 \oplus x_2 = x$
2XORS (2-Party XOR)	$[x]_{1,2}$	$\emptyset$	x_0	x_1	$x_0 \oplus x_1 = x$
1-2XORS (1-and-2 Party XOR)	$[x]_{0,(1,2)}$	x_0	x_1	x_1	$x_0 \oplus x_1 = x$
2-Priv (2-Party Private)	$[x]_{(1,2)}$	$\emptyset$	x	x	
1-Priv (1-Party Private)	$[x]_0$	x	$\emptyset$	$\emptyset$	
Public	x	x	x	x	

We use several kinds of secret-sharing, all of which are bit-wise (Boolean) secret-sharings. These are summarized in Table 3. Since all of these sharings are linear, they can easily be converted between each other. A sharing of an l -bit variable can be converted to a fresh sharing of any other l -bit variable by each party creating a fresh sharing of their share in the new sharing and XORing the resulting shares. (If 2 parties hold the same share, only one of them need send a sharing.) This requires only $\Theta(l)$ communication.

The most common sharing we use is a 3-party replicated secret sharing (3RSS) [2]. Here, $x \in \{0, 1\}^\ell$ is secret-shared by having $x_0, x_1, x_2 \in \{0, 1\}^\ell$ that are uniformly random subject to $x_0 \oplus x_1 \oplus x_2 = x$. P_i holds x_i and $x_{(i+1) \bmod 3}$. When variable x is held using this secret-sharing, it is represented as $[x]$.

We also use a 2-party XOR secret-sharing (2XORS), where 2 parties hold the secret-sharing and the third party is not involved. If P_1 and P_2 hold a 2-party XOR secret-sharing of variable x , this is denoted as $[x]_{1,2}$. Here P_1 holds x_0 and P_2 holds x_1 where x_0 and x_1 are randomly chosen subject to $x_0 \oplus x_1 = x$. We also use a variant of XOR secret-sharing in which 2 parties hold one of the shares, and the third party holds the other. For instance, when P_0 holds one share, and P_1 and P_2 hold the other share, this is denoted $[x]_{0,(1,2)}$, that is P_0 holds x_0 and P_1 and P_2 both hold x_1 where $x_0, x_1 \leftarrow \{0, 1\}^\ell$ subject to $x_0 \oplus x_1 = x$.

Sometimes a variable is held privately. If x is held privately by one party (1-Priv), for instance, by P_0 , we denote this as $[x]_0$. Sometimes a variable is known to 2 parties but not the third (2-Priv). If x is known to P_1 and P_2 , but not P_0 , this is denoted $[x]_{(1,2)}$.

For conciseness, conversions between types of secret-sharing are typically implicit in our pseudocode, indicated by the sharing-type of the result. For instance, $[C_j]_{(1,2)} = [v_{new}] \oplus [e]$ means that $[v_{new}]$ and $[e]$, both stored using 3RSS, are first XORed to create a result that is shared using 3RSS. This result is then revealed to P_1 and P_2 (but not P_0), who store the result and label it C_j .

We use the Arithmetic Black Box (ABB) model [14] to formalize the guarantees provided by secret-sharing and operations on secret-shares. This treats 3RSS, 2XORS and 1-2XORS secret-shared values as stored in a reactive functionality $\mathcal{F}_{ABB}$. $\mathcal{F}_{ABB}$ can also perform operations on secret-shares (e.g. AND, XOR) with the result again being stored by $\mathcal{F}_{ABB}$. Only when a $\mathcal{F}_{ABB}$ -stored value is converted to a private (2-Priv or 1-Priv) or public value (corresponding to share reconstruction) is that value released by $\mathcal{F}_{ABB}$ to the appropriate parties. The protocol of Araki et al. [2] securely implements $\mathcal{F}_{ABB}$ for any Boolean operation (AND, OR, NOT, XOR) over 3RSS-shared values. Locally XORing shares securely implements $\mathcal{F}_{ABB}$ for the XOR operation over 2XORS and 1-2XORS sharings.

5 Functionality

We wish to implement the following DORAM functionality:

Functionality $\mathcal{F}_{DORAM}$

$I \leftarrow \text{Init}(n, d, [A])$: Store array A containing n items of size d .
 $[v] \leftarrow I.\text{ReadWrite}([x], [y], f)$: Given an index $x \in [1, n]$, set v to A_x . Set $A_x = f([v], [y])$.
 $[A] \leftarrow I.\text{Extract}()$: Return the current state of the memory, A , as an array of secret-shares.

Our definition of a DORAM combines the Read and Write functionalities, allowing for reads, writes, or more complex functionalities. This is done by setting the public function f appropriately. For a read, define $f(v, y) = v$. For a write, define $f(v, y) = y$. Allowing the

written value to be a function of the input provides additional flexibility, such as writing to only particular bits of the data-value or applying a bit-mask to the memory value. Implicitly, f must be representable using a Boolean circuit containing $\Theta(d)$ AND gates.

Security is defined using the simulation paradigm, which is standard for proving the security of MPC protocols [30]. A simulator, given only a party's inputs and outputs from a protocol must generate a view consistent with the real view of a corrupted party during an execution. The DORAM is *perfectly* secure if the simulated view is from the same distribution as the real view. It is *statistically* secure if the distance between the distributions of the views is negligible in n . A protocol has *information-theoretic* security if it has either perfect or statistical security. It is *computationally* secure if a computationally-bounded adversary has a negligible advantage in distinguishing views in the simulated and real executions. All of our protocols have information-theoretic security. We present two DORAMs, one that is statistically secure and another that is perfectly secure. The only difference between the two protocols is the choice of hash functions: the perfectly secure protocol selects hash functions which allow for the construction of oblivious hash tables on all possible inputs, whereas the statistically secure protocol picks random hash functions which allow for the construction of oblivious hash tables on all possible inputs except with negligible probability. This is the only type of leakage in the statistically secure protocol. Apart from this all components of both protocols are perfectly secure.

Our DORAM implementation makes use of the following functionalities. These can be implemented with perfect security using standard techniques. We present explicit instantiations of the SSPIR and routing functionalities in appendices A and B respectively. The communication cost of **SSPIR** is $\Theta(\sqrt{md} + d)$ while that of **Route** is $\Theta((d + \log(q))q)$.

Functionality $\mathcal{F}_{SSPIR}$

$[v] \leftarrow \mathbf{SSPIR}(m, d, [A]_{(1,2)}, [x])$: Given an array A held (duplicated) by P_1 and P_2 , containing m elements of size d , and a share of $x \in [1, m]$, return a fresh secret-sharing of A_x .

Functionality $\mathcal{F}_{Route}$

$[B] \leftarrow \mathbf{Route}([A], [Q]_0)$: Given a secret-sharing of array A , of length m , and an injective mapping Q , held by P_0 , of length $q \geq m$, create a fresh secret-sharing B such that $B_{Q(i)} = A_i$ for all $i \in [1, m]$ and B_j is distributed uniformly at random for all $j \notin \{Q(i)\}_{i \in [1, m]}$.

6 DORAM Protocol

6.1 Overview

This section presents MetaDORAM1 and MetaDORAM2 in full and analyzes their security and communication costs. Since MetaDORAM1 and MetaDORAM2 have only minor differences, we first present the generic protocol, which we refer to as MetaDORAM, which does not specify how hash functions are chosen. We then show how the hash functions can be chosen to either provide statistical security or perfect security. For reference, the reader can also refer to the high-level technical overview of the MetaDORAM protocol from section 3.

The MetaDORAM protocol is presented in detail in sections 6.2 and 6.3. Section 6.4 then shows how a statistically secure DORAM (MetaDORAM1) and a perfectly secure DORAM (MetaDORAM2) can be instantiated depending on how the hash functions are chosen, and proves that these protocols achieves the desired security properties. Finally section 6.5

analyzes the complexity of these protocols. We assume the existence of functionalities for secret-shared PIR and secure routing, implementations of which are presented in appendices A and B respectively.

6.2 Writes and Rebuilds

We first show how the data-structure storing the blocks is written to and rebuilt. Initially, all blocks are stored in a single, large, OHTable. When an index is queried, it is assigned a new rune, which is picked by the Builder, and the sub-DORAM is updated with this information. A new block is then created which holds the new value for that index. This block is placed in an area called the *cache*. The cache is filled sequentially. The cache is of size c . When the cache becomes full, its contents are extracted and built into an OHTable.

We implement the OHTables using cuckoo hashing with many ($h \in \Omega(\log(n))$) hash functions. The block may be stored in locations corresponding to the output of the hash functions *on the block's rune*. Since the Builder knows the runes of every block, the Builder is able to *locally* compute an assignment from runes to locations. It can then collaborate with the Holders to securely route the blocks to their correct locations. It is important for the Holders not to be able to tell how the blocks were permuted. It is therefore necessary to mask the blocks using fresh masks during each build. All masks are achieved information-theoretically using one-time pads (OTPs), which are picked by the Builder.

We periodically combine multiple OHTables into a single OHTable. Once there are b OHTables of a given size, the contents of all of these OHTables are built into a single new OHTable. We refer to the OHTables as being arranged in levels. The first, or top, level, L_0 , contains the cache. The next level, L_1 , contains OHTables that were built using the contents of the cache. We label these tables $T_{1,1}, \dots, T_{1,b}$. Since the cache is of capacity c , each OHTable in L_1 will also be of capacity c . L_1 will contain at most b such OHTables; when there are b such tables, their contents will be combined into an OHTable of size bc which will be placed in L_2 , and so on. Note that, once the b th OHTable in a level is built, it is immediately combined with all other OHTables in that level to construct an OHTable in the next level. Therefore, during queries there are only at most $b - 1$ tables at any level. Since each level's capacity is b times larger than that of the level before it, a total of $\Theta(\log_b(n/c))$ levels will be needed to store the blocks created by n queries. After n queries, the refresh occurs, the contents of all OHTables and the cache are extracted, and the active blocks are rebuilt into a single, large OHTable of size n , as at the start of the protocol. To simplify the OHTable builds, we store an append-only secret-shared array, $[V]$, containing all values written (including for reads) since the last refresh. We use subsets of *this array* and *not the existing OHTables* to build new OHTables, which avoids the complexity of actually extracting the contents from existing OHTables and removing previous masks. The Rebuild protocol is presented in Figure 1, together with the overall DORAM ReadWrite function and the Write function.

6.3 Reads and Refreshes

The question remains as to how the function **Read**($[x]$) can be implemented efficiently. Firstly, we reveal the rune of x to the Holders, let it be called r . This greatly simplifies the problem. It is known that the block is stored either in the cache, or in location $H_k(r)$ of some table $T_{i,j}$, for some $i \in [1, \ell], j \in [1, b - 1], k \in [1, h]$. This reduces the number of possible locations to $c + \ell(b - 1)h$.

DORAM: ReadWrite Write Rebuild**Parameters:**

- Cache size: $c = b \cdot h$
- Tables per level: b (Configurable parameter)
- Number of levels: $\ell = \lceil \log_b(n/c) \rceil$
- Number of hash functions: h (Configurable parameter)
- Hash functions: $H_1, \dots, H_h$

DORAM.ReadWrite($[x], [y], f$):

1. $[v] = \text{Read}([x])$ (Defined in Figure 4)
2. $[v_{new}] = f([v], [y])$
3. **Write**($[x], [v_{new}]$)
4. **Rebuild**() (Performs rebuilds, if needed)
5. $t = t + 1$ (Counter indicating the number of queries)
6. Return $[v]$

Write($[x], [v_{new}]$):

1. P_0 sets $[r]_0 = [R_{n+t}]_0$. This will be a rune from $[1, 2n]$ which has not been used since the past refresh, and is chosen at random with replacement. (See **Init** in Figure 2.)
2. $j = t \bmod c$
3. P_0 sets $[e]_0 = [E_{1,[r]_0}]_0$. That is, $[e]_0$ is a OTP from $\{0, 1\}^d$, and P_0 selects it consistently with the mask schedule it generated during **Init** (Figure 2).
4. $[C_j]_{(1,2)} = [v_{new}] \oplus [e]$
5. For future rebuilds and refreshes, the secret-shared v_{new} and x are stored in append-only arrays:
 $[V_{n+t}] = [v_{new}]$
 $[X_{n+t}] = [x]$

Rebuild():

1. if $t = 0 \bmod c$:
 - a. Set i to the largest value such that $t = 0 \bmod cb^i$.
 - b. Set $u = (t/(b^i c)) \bmod b^{i+1} c$ (the number of tables in L_{i+1}).
 - c. The parties obtain the runes, values and (new) masks for the cb^i most-recently written items and build these into a new table.
 for $j \in [1, b^i c]$ (in parallel):
 - i. P_0 sets $[r]_0 = [R_{n+t-c \cdot b^i + j}]_0$
 - ii. P_0 sets $[e]_0$ to the new mask for rune r : $[e]_0 = [E_{i+1,[r]_0}]_0$.
 - iii. $[Z_j] = [V_{n+t-c \cdot b^i + j}] \oplus [e]$.
 - d. P_0 locally builds an OHTable using $R_{i,1 \dots b^i c}$, and hash functions $H_1, \dots, H_h$. Let $[Q]_0$ be the injective mapping from $[1, b^i c]$ to $[1, 2(1+\epsilon)b^i c]$ that maps $R_{i,1 \dots b^i c}$ to satisfying locations with these hash functions.
 - e. Use this mapping to build an OHTable containing the newly masked blocks:
 $[T_{i+1,u}]_{(1,2)} = \mathcal{F}_{Route}([Z], [Q]_0)$
 - f. Delete the old cache C and old tables $T_{1,1}, \dots, T_{i,b-1}$.
2. if $(t = n)$ **Refresh**()

■ **Figure 1** DORAM protocol overview, write function and rebuild function.

These locations constitute the array for the SSPIR; the protocol now needs to obtain secret-shares of the desired item's location in this array. P_0 knows, for each rune and each time, the location at which each item is stored. However, r cannot be revealed to P_0 during a read, since P_0 knows when the index with rune r was last accessed, which would allow P_0 to link access times of indices. In short, the Builder knows the location of each rune, but there seems no way to make use of this without leaking information about the current rune being queried.

Recall that in the description of the DORAM write protocol, P_0 gets to *pick* the rune. P_0 should pick the runes such that each rune is unique, but apart from this runes are chosen uniformly at random from $[1, 2n]$. Therefore, the choice of runes does not depend on any other activity in the protocol. Hence, P_0 is able to pick *all of the runes at the beginning of the protocol*. In other words, P_0 can predetermine the runes that it will assign at each point in time, and during the protocol can assign runes consistently with this original assignment.

Observe, further, that P_0 builds the OHTables based solely on the hash functions and the runes. Since these are both known at the start of the protocol, P_0 can also pre-calculate all assignments in all hash tables at the beginning of the protocol. This allows P_0 to locally create a *position schedule*, that is a data-structure storing exactly where each rune will be located at each point in time.

This allows us to sidestep the conundrum described above. The Builder can secret-share the position schedule containing all information about the locations of all of the runes, once, at the start of the protocol. The Holders can then access the relevant parts of the position schedule dynamically as they learn the rune of each queried block. Note that this location is the location among all of the possible locations that the block may have been located based on the rune (up to c cache locations, and up to $\ell(b-1)h$ table locations).

Given secret-shares of the location of the block, the protocol now engages in a secret-shared PIR (SSPIR) to obtain a secret-sharing of the block. SSPIR can be implemented using a simple modification of any 2-party PIR protocol. The SSPIR protocol we use is explained in more detail in appendix A.

This allows us to obtain secret-sharings of the masked value, but how can this be unmasked? P_0 knows which rune is masked using which OTP, but this information somehow needs to be accessed without revealing to P_0 which rune is being queried. This is the same problem we had with the location mapping, and it can be solved using the same solution. Since the Builder gets to *pick* the OTPs, he can pre-determine, at the initialization of the protocol, which OTPs it will use. He can then secret-share the OTPs that will be used *for all runes at all points in time*. Recall that each time a block is moved, it will be masked using a new OTP. Therefore, P_0 will secret-share a *mask schedule*, analogous to the position schedule, that contains the OTP used to mask each block at each point in time, and which can be accessed dynamically during reads to unmask blocks. This allows us to obtain a secret-sharing of the queried value, performing a read. The read protocol is presented formally in Figure 4.

Although we say above that the Builder will pre-determine all runes, locations and masks at the initialization of the protocol, in fact they will only pre-determine these for the first n queries so that the position schedule and mask schedule are not too large. The protocol will therefore have $2n$ runes (n initial index runes, and n which are assigned during the queries). The Builder only predetermines the assignment of runes and movement of blocks, for the next n queries, and therefore only creates and shares schedules for locations and masks over n points in time. After n queries, the DORAM is refreshed and the Builder generates new runes, position schedules and mask schedules for the next n queries. We stress that the Builder pre-determines the mapping from runes to access times and does *not* pre-determine the mapping from runes to indices. The mapping from access times to indices (and therefore runes to indices) is only determined during the execution of the DORAM as queries occur.

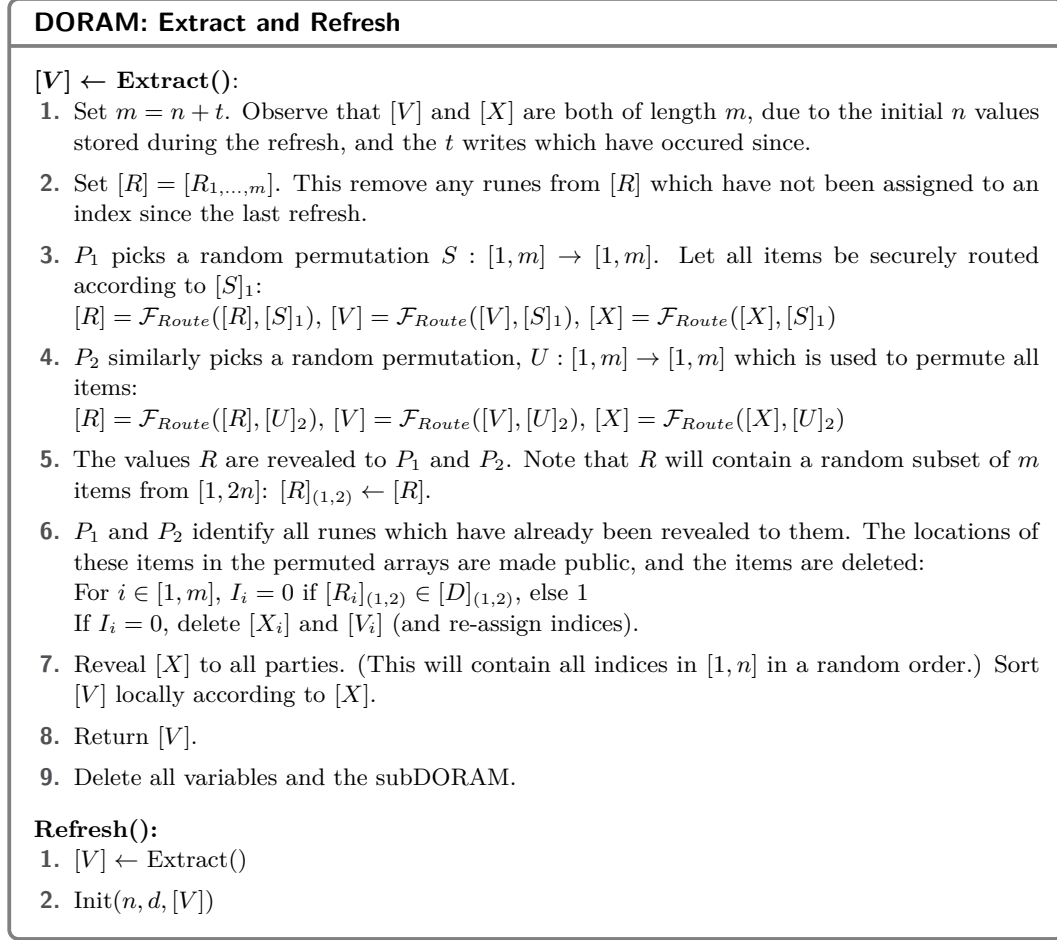
DORAM: Init**Init(n, d, [V]):**

1. P_0 creates a random permutation which determines the assignment of runes, $[R]_0 : [1, 2n] \rightarrow [1, 2n]$. The first n of these are the initial runes of the indices. $[R_{n+t}]_0$ is the rune which will be assigned to the index that is accessed at time t .
2. Initialize a new sub-DORAM containing these runes (with adjacent pairs appended together into a single entry).
 - a. for $i \in [1, n/2]$, $[B_i] = [R_{2i-1}]_0 || [R_{2i}]_0$
 - b. $\text{subDORAM} = \mathcal{F}_{\text{DORAM}}.\text{Init}(\frac{n}{2}, 2(\lg(n) + 1), [B])$
3. P_0 locally builds all the OHTables for the next n queries, based on its knowledge of the runes involved, and the hash functions.
 If there is *no satisfying assignment* for one of the OHTables, P_0 tells P_1 and P_2 to **abort** the protocol.
 Otherwise, P_0 can determine where each rune's block will be when, and it creates the *position schedule* which consists of these three matrices:
 - $[S_{i,r}]_0$ contains the time rune r 's block starts to be in its i^{th} position.
 - $[F_{i,r}]_0$ contains the time rune r 's block finishes to be its i^{th} position.
 - $[P_{i,r}]_0$ contains the i^{th} position of rune r 's block.
4. P_0 creates a mask-schedule. Note that the times will be the same as the position schedule. Therefore all that is needed is one addition matrix containing the OTPs:
 $[E_{i,r}]_0$ contains the OTP used to mask rune r 's block when it is in its i^{th} position.
5. P_0 XOR secret-shares the position schedule and mask schedule between P_1 and P_2 :
 $[S]_{1,2}, [F]_{1,2}, [P]_{1,2}, [E]_{1,2}$.
6. P_0 provides the masks to the blocks, based on his previous selection: $[E_i]_0 = [E_{0,r}]_0$ for $R_i = r$.
7. Based on the Builder's previous assignment of the initial locations of the initial runes, he sets $[Q]_0$ to be the injection from $[1, n]$ to $[1, 2(1 + \epsilon)n]$ that builds the initial table.
8. The parties create the OHTable containing the initial items, and P_1 and P_2 store the masked blocks:
 $[T_{\ell+1}]_{(1,2)} \leftarrow \mathcal{F}_{\text{Route}}([V] \oplus [E]_0, [Q]_0)$
9. The values of the initial items are stored for future reference in $[V_{1,\dots,n}]$.
10. The indexes are stored in an array $[X]$, that is for $i \in [1, n]$:
 $[X_i] = [i]$
11. Initialize the query counter: $t = 1$.

■ **Figure 2** DORAM: Init functionality.

We now describe the method for refreshing in more detail. The refresh can be divided into two parts. First, the contents of the up-to-date memory is extracted. This is achieved by randomly permuting all blocks and revealing their runes to the Holders. The Holders know which runes have been queried, so can identify these blocks as obsolete, leaving only the blocks which contain the most recently written value for each index. The extract protocol returns a secret-shared array of the current memory; that is using the same format as that provided for the Init function. The refresh protocol then simply call the Init function using this secret-shared array to create all of the data-structures necessary for a further n queries. The Extract functionality is useful in its own right, and may be called by the environment

at an arbitrary time (i.e. when there have been fewer than n queries since the last refresh). The Refresh and Extract protocols are presented formally in Figure 3, while the Init protocol is presented in Figure 2.



■ **Figure 3** Extract and Refresh functionalities.

6.4 Security Analysis

In this section, we show how to instantiate the MetaDORAM protocol so as to achieve statistical security (MetaDORAM1) and perfect security (MetaDORAM2). In short, MetaDORAM1 picks hash functions at random, and is secure in the event (which occurs except with negligible probability) that the hash functions are suitably chosen. MetaDORAM2 instead manually verifies that the hash functions are suitably chosen. We first prove the security of the protocol in the event that the hash functions are suitably chosen.

► **Theorem 1.** *Let $H_1, \dots, H_h$ satisfy the property that for all $i \in [0, l]$, for $m = cb^i$ and for all subsets M of size m of $[1, 2n]$, there exists an assignment $a_1, \dots, a_m \in [1, h]^m$ such that $H_{a_i}(M_i) \bmod 2(1 + \epsilon)m$ are distinct. In this case, the MetaDORAM protocol, as presented in Figures 1, 2, 3 and 4 is perfectly secure in the $\mathcal{F}_{ABB}$, $\mathcal{F}_{SSPIR}$, $\mathcal{F}_{Route}$ -hybrid model.*

DORAM: Read**Read**($[x]$):

1. Access the subDORAM to learn the rune of x . Note that indices are stored in the subDORAM in pairs, so the subDORAM will return a share of both x 's rune and a share of x 's neighbor's rune. The protocol reveals (only) x 's rune to P_1 and P_2 . Also, in order to access the subDORAM only once per query, the protocol takes the opportunity to use this access to also write the new rune that is being assigned to x .
 - a. Let $[x_{\ln(n)}]$ be the least significant bit of $[x]$ (i.e. if x is odd it is 1, otherwise 0).
 - b. Set $[x_{sig}]$ to be the $\lg(n) - 1$ most significant bits of $[x]$, (i.e. drop the last bit).
 - c. P_0 supplies the new rune $[r_{new}]_0$ which will be assigned to x when it is re-written.
 - d. We define f to overwrite x with its new rune, while leaving x 's neighbor as is. Formally $f(v, y)$, $v \in \{0, 1\}^{2(\lg(n)+1)}$, $y \in \{0, 1\}^{\lg(n)+2}$ is defined such that if $y_0 = 0$ (which will happen when x is even) $f(v, y) = v_{1, \dots, \lg(n)+1} || y_{1, \dots, \lg(n)+1}$ (the second half of the value is overwritten with the remaining bits of y) and if $y_0 = 1$ (x is odd), $f(v, y) = y_{1, \dots, \lg(n)+1} || v_{\lg(n)+2, \dots, 2\lg(n)+1}$ (the first half is overwritten).
 - e. $[v_{sub}] \leftarrow \text{subDORAM.ReadWrite}([x_{sig}], [x_{\ln(n)}] || [r_{new}], f)$.
 - f. If $[y_0] = 1$, securely set $[r_{old}]$ to be the first half of $[v_{sub}]$, otherwise securely set it to be the second half of $[v_{sub}]$.
 - g. Reveal x 's (old) rune to P_1 and P_2 : $[r]_{(1,2)} \leftarrow [r_{old}]$.
 - h. Append $[r]_{(1,2)}$ to $[D]_{(1,2)}$, the set of runes which P_1 and P_2 have already observed.
2. P_1 and P_2 create an array Y containing all of the (masked) blocks which may hold rune r 's block:
 - a. $[Y_{1, \dots, c}]_{(1,2)}$ contains the blocks from the cache. These are padded to length c with empty blocks if the cache is not full.
 - b. For $i \in [1, \ell + 1]$, $u \in [1, b - 1]$, $k \in [1, h]$, set $[Y_{c+(i-1)bh+(u-1)h+k}]_{(1,2)} \leftarrow [T_{i,u,H_k([r]_{(1,2)})}]_{(1,2)}$. This is, the $H_k([r])^{th}$ location in table $T_{i,u}$. If table $T_{i,u}$ does not exist, set location to an empty block.
3. Securely determine which time-slot is being used. That is, for $j \in [0, \ell + 1]$:
 - a. Set $[S_j] \leftarrow [S_{j,[r]_{(1,2)}}]_{1,2} \geq t$
 - b. Set $[F_j] \leftarrow [F_{j,[r]_{(1,2)}}]_{1,2} < t$
 - c. Set $[J_j]_{1,2} \leftarrow [S_j] \wedge [F_j]$
4. Securely select the correct location and OTP from the position and mask schedules:
 - a. For $j \in [0, \ell + 1]$, $[P_j] \leftarrow [P_{j,[r]_{(1,2)}}]_{1,2}$
 - b. For $j \in [0, \ell + 1]$, $[E_j] \leftarrow [E_{j,[r]_{(1,2)}}]_{1,2}$
 - c. For $j \in [0, \ell + 1]$, securely set $[p]$ to $[P_j]$ if $[J_j] = 1$
 - d. For $j \in [0, \ell + 1]$, securely set $[e]$ to $[E_j]$ if $[J_j] = 1$
5. $[v] \leftarrow \mathcal{F}_{\text{BalancedSSPIR}}(c + \ell(b-1)h, d, [Y]_{(1,2)}, [p]) \oplus [e]$
6. Return $[v]$

■ **Figure 4** DORAM read protocol.

Proof. All steps of the protocol are one of three cases. Either:

- A secure functionality is being accessed, that only outputs secret-shared results. This can either be a basic ABB functionality, like $\oplus$, or a more sophisticated functionality like **SSPIR**.
- The operations are on public, predetermined values (e.g. t , u).
- Some value is revealed to some party, or subset of the parties.

We need to examine all revealed values and examine whether they can be simulated without knowledge of the private inputs.

Init. No information is revealed to P_0 , rather all private variables it holds are the result of its own random choices (the runes and OTPs) and public parameters (the hash functions). In the case that $H_1, \dots, H_h$ have satisfying assignments for all subsets of size m of $[1, 2n]$, then P_0 will never abort. Therefore, the only information P_1 and P_2 learn during the Init function are the values $T_{\ell+1}$. All of these blocks have been masked by fresh OTPs, so this is simulatable by generating a uniformly random string.

Read. No information is revealed to P_0 .

P_1 and P_2 learn the rune queried. The runes are distributed uniformly at random from $[1, 2n]$, subject to the fact that they are each unique.

Write. No information is revealed to P_0 .

P_1 and P_2 learn C_j . This has been masked using a fresh OTP, so can be simulated by generating a random string.

Rebuild. No information is revealed to P_0 .

P_1 and P_2 learn $T_{i,u}$. This contains blocks which have been masked under fresh OTPs, so can be simulated by generating random strings.

Extract. P_0 learns X . This will contain the items $[1, n]$ in a randomly permuted order. This can be seen by induction. The protocol maintains the invariant that at each point in time, each index x has a single rune assigned to it which has not been observed by P_1 and P_2 . In other words, there is a single rune R_i , such that $X_i = x$ and $R_i \notin D$. Therefore, when the indices corresponding to viewed runes are deleted, a single instance of each index will remain. They will be in a random order because they have been shuffled according to a permutation known to no parties.

P_0 also learns I . This contains n 1s and $m - n$ 0s in a random order, for the reasons explained above.

P_1 and P_2 additionally learn R after it has been permuted. This contains a subset of m runes from $[1, 2n]$. It will necessarily include all $m - n$ runes from D , since these runes are definitely stored in the system. The other n runes are distributed uniformly at random from the set of the remaining $2n - (m - n)$ runes, so are efficiently simulatable. The ordering must be consistent with I , that is the $m - n$ previously observed runes must have $I_i = 0$.

Therefore, the views of all parties are perfectly simulatable, so the protocol is secure in the semi-honest setting against an adversary that corrupts any one of the parties. ◀

In the case that $H_1, \dots, H_h$ are such that there is no satisfying assignment for some subset of $[1, 2n]$ of some size $m = cb^i$, then there is some small leakage. In the case that this subset is chosen, this leads to an abort, so does not leak any information. However, if such a subset is not chosen, P_1 and P_2 learn that such a subset was not chosen. Since P_1 and P_2 learn the rune of items when they are accessed, they can therefore conclude that certain access patterns were impossible, as they would have led to tables that were unconstructable. Note that this type of leakage can occur even if the probability of P_0 actually choosing a rune assignment that would lead to an abort is negligible: P_1 and P_2 could learn of some access pattern which for certain did not occur.

We present two solutions to this problem. MetaDORAM1 selects $\omega(1)\log(n)$ hash functions at random, for any super-constant function $\omega(1)$. We show that, except with negligible probability in n , this results in a choice of hash functions which have a satisfying assignment for all subsets of size m of $[1, 2n]$, for all $m = cb^i$. This results in a protocol which has a negligible probability of any leakage, and is therefore statistically secure.

In MetaDORAM2, the protocol instead selects $\Theta(\log(n))$ hash functions at random and manually verifies that this choice of hash functions result in a satisfying assignment for all subsets of size m of $[1, 2n]$, for all $m = cb^i$. The verification stage requires an exponential time setup phase (which need only be done once for any value n). This allows for a perfectly secure protocol.

We prove both protocols secure by making use of Yeo's analysis of Robust Cuckoo Hashing [41]. Yeo was concerned with an adversary that could pick the indices of items in a hash table, and attempted to pick these such that would cause a build failure, given the predetermined hash functions. His analysis works in general for determining the probability that, given a large set of elements there exists some subset of these that would result in a build failure. Specifically, from his proof of Lemma 3 we can derive the following:

► **Lemma 2** (Derived from proof of [41] Lemma 3). *For some $m \leq 2n$, let C be a disjoint-table cuckoo hash table with αm locations ($\alpha \geq 1$), and h random hash functions $H_1, \dots, H_h$. Furthermore, each location in C is of capacity $l = 1$ and C does not have a stash ($s = 0$). Then all subsets of $[1, 2n]$ of size m can be successfully built by C , except with probability:*

$$\epsilon \leq \left(\frac{2n}{2^{h-3}} \right)^{h+1}$$

Note that this probability does not depend on m , except for requiring that $m \leq 2n$. For $h = \lg(n) + 5$, this simplifies to

$$\left(\frac{1}{2} \right)^{\lg(n)+6} = \frac{1}{64n}$$

For any $h = \omega(\log(n))$ this is negligible in n .

6.4.1 MetaDORAM1

For MetaDORAM1, we set $h = \lg^{1.5}(n) / \lg(\lg(n)) = \omega(\log(n))$. We select h independent random hash functions. By Lemma 2, this means that the failure probability is negligible in n . Note that this gives the failure probability for a given m , but as there are fewer than n such values of m to consider (even given the recursive implementation of the sub-ORAM) the probability that there is any m for which a subset of size m could not have a satisfying assignment is also negligible. Therefore MetaDORAM1 is perfectly secure, except in the case of an event (poorly chosen hash functions) which occurs with probability negligible in n . This leads to our desired result:

► **Corollary 3.** *MetaDORAM1 is a statistically secure implementation of functionality $\mathcal{F}_{DORAM}$ in the $\mathcal{F}_{ABB}$, $\mathcal{F}_{SSPIR}$, $\mathcal{F}_{Route}$ -hybrid model.*

Note that the subDORAMs, even though they have smaller sizes, use the same parameter h as the top level, so that the failure probability remains negligible in the size of the top DORAM, n .

6.4.2 MetaDORAM2

For MetaDORAM2, we set $h = \lg(n) + 5$. This means that the choice of hash functions satisfies all subsets of a given size m with probability $\frac{1}{64n}$. Therefore, it also satisfies all subsets for all $m \leq n$ except with probability at most $\frac{1}{64}$. The protocol selects a random $H_1, \dots, H_h$ and then attempts to build the hash tables using all subsets of $[1, 2n]$ of size m ,

for all $h \leq m \leq n$. If any subset does not have a satisfying assignment, new random hash functions are selected and the process is repeated. If all subsets have a satisfying assignment, these hash functions are used for the protocol.

Unfortunately, iterating over all subsets of $[1, 2n]$ of size m requires $\binom{2n}{m}$ iterations. Further iterating over all $m \in [h, n]$ results in nearly 2^{2n-1} iterations. This exponential-time setup phase makes the protocol infeasible for practical applications. Nevertheless, it does not affect the communication cost of the protocol, nor does it undermine its perfect security.

By thus choosing the hash functions, the condition of Theorem 1 is satisfied:

► **Corollary 4.** *MetaDORAM2 is a perfectly secure implementation of functionality $\mathcal{F}_{\text{DORAM}}$ in the $\mathcal{F}_{\text{ABB}}$, $\mathcal{F}_{\text{SSPIR}}$, $\mathcal{F}_{\text{Route-hybrid}}$ model.*

6.5 Complexity Analysis

The complexity analysis is presented in the full version of the paper [32].

7 Future Work

A discussion of future work is included in the full version [32].

References

- 1 Ittai Abraham, Christopher W Fletcher, Kartik Nayak, Benny Pinkas, and Ling Ren. Asymptotically tight bounds for composing ORAM with PIR. In *IACR International Workshop on Public Key Cryptography*, pages 91–120. Springer, 2017. doi:10.1007/978-3-662-54365-8_5.
- 2 Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. High-throughput semi-honest secure three-party computation with an honest majority. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 805–817, 2016. doi:10.1145/2976749.2978331.
- 3 Toshinori Araki, Jun Furukawa, Kazuma Ohara, Benny Pinkas, Hanan Rosemarin, and Hikaru Tsuchida. Secure graph analysis at scale. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 610–629. ACM, 2021. doi:10.1145/3460120.3484560.
- 4 Gilad Asharov, Ilan Komargodski, Wei-Kai Lin, Kartik Nayak, Enoch Peserico, and Elaine Shi. OptORAMa: optimal oblivious RAM. In *Advances in Cryptology-EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part II 30*, pages 403–432. Springer, 2020. doi:10.1007/978-3-030-45724-2_14.
- 5 Gilad Asharov, Ilan Komargodski, Wei-Kai Lin, and Elaine Shi. Oblivious ram with worst-case logarithmic overhead. *Journal of Cryptology*, 36(2):7, 2023. doi:10.1007/s00145-023-09447-5.
- 6 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10. ACM, 1988. doi:10.1145/62212.62213.
- 7 Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. In Sushil Jajodia and Javier López, editors, *Computer Security - ESORICS 2008, 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings*, volume 5283 of *Lecture Notes in Computer Science*, pages 192–206. Springer, Springer, 2008. doi:10.1007/978-3-540-88313-5_13.

- 8 Paul Bunn, Jonathan Katz, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient 3-party distributed ORAM. In *Security and Cryptography for Networks: 12th International Conference, SCN 2020, Amalfi, Italy, September 14–16, 2020, Proceedings 12*, pages 215–232. Springer, 2020. doi:10.1007/978-3-030-57990-6_11.
- 9 T-H Hubert Chan, Yue Guo, Wei-Kai Lin, and Elaine Shi. Oblivious hashing revisited, and applications to asymptotically efficient ORAM and OPRAM. In *Advances in Cryptology—ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*, pages 660–690. Springer, 2017. doi:10.1007/978-3-319-70694-8_23.
- 10 T-H Hubert Chan, Jonathan Katz, Kartik Nayak, Antigoni Polychroniadou, and Elaine Shi. More is less: Perfectly secure oblivious algorithms in the multi-server setting. In *Advances in Cryptology—ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part III 24*, pages 158–188. Springer, 2018. doi:10.1007/978-3-030-03332-3_7.
- 11 Hao Chen, Ilaria Chillotti, and Ling Ren. Onion ring oram: Efficient constant bandwidth oblivious ram from (leveled) tffe. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 345–360, 2019. doi:10.1145/3319535.3354226.
- 12 Victor Costan and Srinivas Devadas. Intel SGX explained. *IACR Cryptol. ePrint Arch.*, page 86, 2016. URL: <http://eprint.iacr.org/2016/086>.
- 13 Ronald Cramer, Ivan Damgård, and Yuval Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, volume 3378 of *Lecture Notes in Computer Science*, pages 342–362. Springer, Springer, 2005. doi:10.1007/978-3-540-30576-7_19.
- 14 Ivan Damgård and Jesper Buus Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In *Annual international cryptography conference*, pages 247–264. Springer, 2003. doi:10.1007/978-3-540-45146-4_15.
- 15 Srinivas Devadas, Marten van Dijk, Christopher W Fletcher, Ling Ren, Elaine Shi, and Daniel Wichs. Onion oram: A constant bandwidth blowup oblivious ram. In *Theory of Cryptography: 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II 13*, pages 145–174. Springer, 2016. doi:10.1007/978-3-662-49099-0_6.
- 16 Sky Faber, Stanislaw Jarecki, Sotirios Kentros, and Boyang Wei. Three-party ORAM for secure computation. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 360–385. Springer, 2015. doi:10.1007/978-3-662-48797-6_16.
- 17 Brett Hemenway Falk, Daniel Noble, and Rafail Ostrovsky. 3-party distributed ORAM from oblivious set membership. In *International Conference on Security and Cryptography for Networks*, pages 437–461. Springer, 2022. doi:10.1007/978-3-031-14791-3_19.
- 18 Jun Furukawa, Yehuda Lindell, Ariel Nof, and Or Weinstein. High-throughput secure three-party computation for malicious adversaries and an honest majority. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 225–255. Springer, 2017. doi:10.1007/978-3-319-56614-6_8.
- 19 Oded Goldreich. Towards a theory of software protection and simulation by oblivious RAMs. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 182–194, 1987. doi:10.1145/28395.28416.
- 20 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 307–328. ACM, 2019. doi:10.1145/3335741.3335755.
- 21 Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious RAMs. *Journal of the ACM (JACM)*, 43(3):431–473, 1996. doi:10.1145/233551.233553.

- 22 Michael T Goodrich and Michael Mitzenmacher. Privacy-preserving access of outsourced data via oblivious RAM simulation. In *International Colloquium on Automata, Languages, and Programming*, pages 576–587. Springer, 2011. doi:10.1007/978-3-642-22012-8_46.
- 23 Yuval Ishai, Eyal Kushilevitz, Steve Lu, and Rafail Ostrovsky. Private large-scale databases with distributed searchable symmetric encryption. In Kazue Sako, editor, *Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings*, volume 9610 of *Lecture Notes in Computer Science*, pages 90–107. Springer, Springer, 2016. doi:10.1007/978-3-319-29485-8_6.
- 24 Stanislaw Jarecki and Boyang Wei. 3PC ORAM with low latency, low bandwidth, and fast batch retrieval. In *Applied Cryptography and Network Security: 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings 16*, pages 360–378. Springer, 2018. doi:10.1007/978-3-319-93387-0_19.
- 25 Ilan Komargodski and Wei-Kai Lin. A logarithmic lower bound for oblivious ram (for all parameters). In *Advances in Cryptology-CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part IV 41*, pages 579–609. Springer, 2021. doi:10.1007/978-3-030-84259-8_20.
- 26 Eyal Kushilevitz, Steve Lu, and Rafail Ostrovsky. On the (in)security of hash-based oblivious RAM and a new balancing scheme. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 143–156. SIAM, 2012. doi:10.1137/1.9781611973099.13.
- 27 Kasper Green Larsen and Jesper Buus Nielsen. Yes, there is an oblivious RAM lower bound! In *Annual International Cryptology Conference*, pages 523–542. Springer, 2018. doi:10.1007/978-3-319-96881-0_18.
- 28 Kasper Green Larsen, Mark Simkin, and Kevin Yeo. Lower bounds for multi-server oblivious RAMs. In *Theory of Cryptography: 18th International Conference, TCC 2020, Durham, NC, USA, November 16–19, 2020, Proceedings, Part I 18*, pages 486–503. Springer, 2020. doi:10.1007/978-3-030-64375-1_17.
- 29 Sven Laur, Jan Willemson, and Bingsheng Zhang. Round-efficient oblivious database manipulation. In Xuejia Lai, Jianying Zhou, and Hui Li, editors, *Information Security, 14th International Conference, ISC 2011, Xi'an, China, October 26-29, 2011. Proceedings*, volume 7001 of *Lecture Notes in Computer Science*, pages 262–277. Springer, Springer, 2011. doi:10.1007/978-3-642-24861-0_18.
- 30 Yehuda Lindell. How to simulate it - A tutorial on the simulation proof technique. *Electron. Colloquium Comput. Complex.*, TR17-112:277–346, 2017. URL: <https://eccc.weizmann.ac.il/report/2017/112>.
- 31 Steve Lu and Rafail Ostrovsky. Distributed oblivious RAM for secure two-party computation. In *Theory of Cryptography Conference*, pages 377–396. Springer, 2013. doi:10.1007/978-3-642-36594-2_22.
- 32 Daniel Noble, Brett Hemenway Falk, and Rafail Ostrovsky. MetaDORAM: Info-theoretic distributed ORAM with less communication. *IACR Cryptol. ePrint Arch.*, page 11, 2024. URL: <https://eprint.iacr.org/2024/011>.
- 33 Rafail Ostrovsky. An efficient software protection scheme. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*. Springer, 1989. doi:10.1007/0-387-34805-0_55.
- 34 Rafail Ostrovsky. Efficient computation on oblivious RAMs. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 514–523, 1990. doi:10.1145/100216.100289.
- 35 Rafail Ostrovsky and Victor Shoup. Private information storage (extended abstract). In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 294–303. ACM, 1997. doi:10.1145/258533.258606.

- 36 Sarvar Patel, Giuseppe Persiano, Mariana Raykova, and Kevin Yeo. PanORAMa: Oblivious RAM with logarithmic overhead. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 871–882. IEEE, 2018. doi:10.1109/FOCS.2018.00087.
- 37 Giuseppe Persiano and Kevin Yeo. Limits of breach-resistant and snapshot-oblivious rams. In *Annual International Cryptology Conference*, pages 161–196. Springer, 2023. doi:10.1007/978-3-031-38551-3_6.
- 38 Benny Pinkas and Tzachy Reinman. Oblivious RAM revisited. In *Advances in Cryptology—CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15–19, 2010. Proceedings 30*, pages 502–519. Springer, 2010. doi:10.1007/978-3-642-14623-7_27.
- 39 Adithya Vadapalli, Ryan Henry, and Ian Goldberg. Duoram: A bandwidth-efficient distributed ORAM for 2- and 3-party computation. In Joseph A. Calandrino and Carmela Troncoso, editors, *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9–11, 2023*, pages 3907–3924. USENIX Association, 2023. URL: <https://www.usenix.org/conference/usenixsecurity23/presentation/vadapalli>.
- 40 Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3–5 November 1982*, pages 160–164. IEEE Computer Society, 1982. doi:10.1109/SFCS.1982.38.
- 41 Kevin Yeo. Cuckoo hashing in cryptography: Optimal parameters, robustness and applications. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20–24, 2023, Proceedings, Part IV*, volume 14084 of *Lecture Notes in Computer Science*, pages 197–230. Springer, 2023. doi:10.1007/978-3-031-38551-3_7.

A Secret-Shared Private Information Retrieval

The full version [32] presents an instantiation of the SSPIR protocol, using standard techniques.

B Secure Routing

The full version [32] presents an instantiation of the secure routing protocol using standard techniques.

Linear-Time Secure Merge in $O(\log \log n)$ Rounds

Mark Blunk 

Stealth Software Technologies, Inc., Los Angeles, CA, USA

Paul Bunn 

Stealth Software Technologies, Inc., Los Angeles, CA, USA

Samuel Dittmer 

Stealth Software Technologies, Inc., Los Angeles, CA, USA

Steve Lu 

Stealth Software Technologies, Inc., Los Angeles, CA, USA

Rafail Ostrovsky 

Departments of Computer Science and Mathematics, UCLA, Los Angeles, CA, USA

Abstract

The problem of Secure Merge consists of combining two sorted lists (which are either held separately by two parties, or secret-shared among two or more parties), and outputting a single merged (sorted) list, secret-shared among all parties. Just as *insecure* algorithms for comparison-based **sorting** are slower than **merging** (i.e., for lists of size n , $\Theta(n \log n)$ versus $\Theta(n)$), we explore whether an analogous separation exists for *secure* protocols; namely, if there exist techniques for performing *secure* merge that are more performant than simply invoking secure sort.

We answer this question affirmatively by constructing a secure merge protocol with optimal $\Theta(n)$ communication and computation, and $\Theta(\log \log n)$ rounds of communication. Our results are based solely on black-box use of basic secure primitives, such as secure comparison and secure shuffle. Since two-party secure primitives require computational assumptions, while three-party do not, our protocols achieve these bounds against semi-honest adversaries via a computationally secure two-party (resp. an information-theoretically secure three-party) secure merge protocol.

Secure **sort** is a fundamental building block used in many MPC protocols, e.g., various private set intersection protocols and oblivious RAM protocols. More efficient secure sort can lead to concrete improvements in the overall run-time. Since secure sort can often be replaced by secure merge – as inputs (from different participating players) can be presorted – an efficient secure merge protocol has wide applicability. There are also a range of applications in the field of secure databases, including secure database *joins*, as well as updatable database storage and search, whereby secure merge can be used to insert new entries into an existing (sorted) database.

In building our secure merge protocol, we develop several subprotocols that may be of independent interest. For example, we develop a protocol for secure *asymmetric* merge (when one list is much larger than the other).

2012 ACM Subject Classification Security and privacy → Cryptography; Security and privacy → Database and storage security

Keywords and phrases Secure Merge, Secure Sort, Secure Databases, Private Set Intersection

Digital Object Identifier 10.4230/LIPIcs.ITC.2025.7

Related Version *Full Version*: <https://eprint.iacr.org/2022/590>

Funding This work was performed under the following financial assistance award number 70NANB21H064 from U.S. Department of Commerce, National Institute of Standards and Technology. The statements, findings, conclusions, and recommendations are those of the author(s) and do not necessarily reflect the views of the National Institute of Standards and Technology or the U.S. Department of Commerce.



© Mark Blunk, Paul Bunn, Samuel Dittmer, Steve Lu, and Rafail Ostrovsky;
licensed under Creative Commons License CC-BY 4.0
6th Conference on Information-Theoretic Cryptography (ITC 2025).
Editor: Niv Gilboa; Article No. 7; pp. 7:1–7:23



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

Rafail Ostrovsky: This research was supported in part by DARPA under Cooperative Agreement HR0011-20-2-0025, the Algorand Centers of Excellence programme managed by Algorand Foundation, NSF grants CNS-2246355, CCF-2220450, CNS-2001096, US-Israel BSF grant 2022370, Amazon Faculty Award and Sunday Group. Any views, opinions, findings, conclusions, or recommendations contained herein are those of the author(s) and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, the Department of Defense, the Algorand Foundation, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes, notwithstanding any copyright annotation therein.

1 Introduction

As the practice of collecting and analyzing data has increased in recent decades, combined with the growing desire of examining data of different types held by different organizations, there has been a corresponding desire for protocols that are able to compute on aggregated data without actually requiring the raw data to be combined or exposed. Indeed, the field of secure multiparty computation (MPC) within cryptography seeks to address exactly such scenarios. Meanwhile, with the **sort** functionality being a prominent component of several desired analyses, there has been significant work in optimizing *secure sort* functionality (e.g. **sort** under MPC), as well other basic secure functionalities that require **sort** as a subroutine. For example, in research in the MPC subfields of Private-Set Intersection (PSI) [8, 17, 24] and Oblivious RAM (ORAM) [20, 23], it is often the case that the secure **sort** is the bottleneck in resulting protocols in these areas.

While MPC protocols for **sort** – and related functionalities that build upon it – have become extremely efficient, there is an unavoidable cost of e.g., a security parameter multiplier in measuring the complexity of any secure computation, on top of the $\Theta(n \log n)$ cost of performing (insecure) **sort** (here n is the size of the list in question). Given the common scenario in which a handful of organizations each have access to their own data, and require a **sort** on their aggregated lists, it is natural to ask if one can avoid the inherent overhead of securely implementing **sort** by first having parties locally sort their own data (which can be done *insecurely*, and hence without incurring this overhead), and then performing a secure **merge** on the individual sorted lists. Indeed, this approach has the promise of adding security at minimal cost, since the overhead of adding security is now only applied to the $\Theta(n)$ **merge** protocol, which effectively means there is an extra $\Theta(\log n)$ cushion to absorb the overhead cost of performing the computation securely, and still having an overall $\Theta(n \log n)$ *secure sort* protocol.

Unfortunately, attempting to minimize the overhead of secure computation by first performing a local (insecure) **sort** followed by secure **merge** has to-date been an ineffective strategy, due to the fact that much less is known about secure **merge** protocols than secure **sort** protocols. Indeed, prior to our work, there was no known secure **merge** protocol that simultaneously achieved near-optimal asymptotic performance in the three key metrics: computation, communication, and round complexity (see Section 2.2 and Table 1 for a summary of prior works exploring secure merge, and how they perform in these three metrics). In this work, we present a secure **merge** protocol that effectively eliminates all inefficiencies, thereby substantially reducing the cost of secure **sort** (where the above strategy of first locally sorting one’s own data can be employed) and any secure protocol that builds on top of it. Namely, we construct a protocol that instantiates the following:

► **Theorem 1 (Informal).** *There exists a secure merge protocol for two or three parties with $\Theta(n)$ run-time (computation and communication) and $\Theta(\log \log n)$ rounds that relies only on black-box access to the secure functionalities defined in §3.4: comparison, shuffle, and conditional-select (mux).*

Theorem 1 is asymptotically optimal in two of the three key metrics (computation and communication), and near-optimal in round-complexity. Our theorem gives a two-party secure merge under standard cryptographic assumptions, and a three-party secure merge with information-theoretic setting with semi-honest security and no collusion. Indeed, in the two-party case, all black-box functionalities referenced in the theorem above can be efficiently realized using two standard cryptographic assumptions: Oblivious transfer (OT) and the existence of an additively homomorphic public key cryptosystem. Meanwhile, in the three-party case, these black-box functionalities can be realized with information-theoretic security, see e.g., [11]. While there are ways to extend our results to more parties beyond three, see for example the discussion in the full version; we do not explore these extensions directly in this paper.

Besides being useful as a stand-in for secure `sort` in scenarios where two or more parties can each locally sort their own data (“in the clear”), the secure `merge` functionality has broader relevance in many other applications as well. In the area of database management, for example, in settings where records are encrypted, our secure merge protocol can be used for database operations such as joins, or for inserting new records into the database [2, 9, 18, 27].

1.1 Paper Structure

In §2 we summarize previous work on the secure sort/merge problem, and provide a comparison of our main result with relevant earlier works in Table 1. In §3 we set notation and discuss the high level techniques and primitives on which our merge protocols rely. We present our main secure merge protocols, together with statements and proof sketches of their properties, in §4; and then analyze their complexity in §5. In §6 we present the two underlying secure merge protocols required by our protocols in §4. Supplementary material including a survey of applications whose performance could be improved by the adoption of our secure merge protocol as well as an expanded discussion of our main results with additional protocols – including the description of a separate constant round secure merge protocol in the asymmetric case that is a generalization of the protocol $\Pi_{\text{SAM-}n^{1/3}}$ of §6 (where one list has size n and the other n^α for fixed $\alpha < 1$) – can be found in the full version.

2 Previous Work

As the merging/sorting of lists is a fundamental problem in computer science, there has been enormous research in this area. Consequently, we summarize here only the most relevant works; see cited works and references therein for a more complete overview and discussion. Not surprisingly, many of the protocols for *securely* merging/sorting lists draw inspiration from their *insecure* counterparts (not to mention the generic approach of converting insecure protocols to secure protocols, e.g., via garbled circuits or ORAM or Garbled RAM (GRAM) or fully-homomorphic encryption). We discuss below previous work for both insecure and secure variants, and compare previous results to our main result in Table 1.

2.1 Insecure Merge Algorithms

Sorting Networks. The relationship between secure merging and secure sorting can be traced back to [3], which built a sorting network of size $O(n \log^2 n)$ from $\log n$ merging networks, each of size $O(n \log n)$. There are $(2n)! = 2^{O(n \log n)}$ permutations on $2n$ elements, but $\binom{2n}{n} = 2^{O(n)}$ possible ways two sorted lists can be merged together. This gives combinatorial lower bounds of $\Omega(n \log n)$ and $\Omega(n)$ comparators for sorting and merging networks, respectively. Although an asymptotically optimal sorting network of size $O(n \log n)$ was later achieved by Ajtai,

Komlos, and Szemerédi [1], merging networks cannot achieve the combinatorial lower bound of n comparators. A merging network on lists of size $\Theta(n)$ require $\Omega(n \log n)$ comparators, as shown by Yao and Yao [26], and depth of $\Omega(\log n)$, as shown by Hong and Sedgewick [16].

RAM and PRAM. The classical merge algorithm requires $O(n)$ work on a RAM machine, see e.g., [21]. As discussed below, [10] uses an approach that is inspired by this classical merge algorithm, and achieves matching asymptotic work (albeit with $O(n)$ round-complexity).

A parallel RAM machine (PRAM) allows multiple processors to act on the same set of memory in parallel. We are in particular interested in Concurrent-Read-Exclusive-Write (CREW) PRAM, where all processors can read the same memory simultaneously, but processors cannot write to the same memory address at the same time, see e.g., [4] for more formal definitions and a discussion of various PRAM models. For PRAM machines, Valiant showed in [25] that $O(n)$ processors could merge two lists of size $O(n)$ in time $O(\log \log n)$. This was improved by Borodin and Hopcroft [4] to $O(n/\log \log n)$ processors, who also showed that the time bound of $O(\log \log n)$ is optimal when limited to $O(n)$ processors. As a rough heuristic, we expect the number of processors and total work done by a PRAM algorithm to serve as a lower bound for the round complexity and communication complexity of a corresponding secure protocol, motivating the following:

► **Conjecture.** *Any linear-time protocol for two parties to securely merge their lists (each of size $\Theta(n)$) requires $\Omega(\log \log n)$ rounds of communication.*

Notice that if the above conjecture is valid, then our secure merge protocol (§4) is asymptotically optimal in all three key metrics. To lend weight to this conjecture, we note that if a 2-party protocol Π securely realizes some functionality $\mathcal{F}$ in R rounds and C communication, and each party can execute each round of the protocol in $O(1)$ time on a CREW PRAM with C processors, then there exists an algorithm A that realizes $\mathcal{F}$ on a single CREW PRAM machine with C processors, $O(RC)$ total work, and $O(R)$ time. Indeed, A merely executes Π , playing the role of all parties. If a protocol for secure merge existed with $C = n$, $R = o(\log \log n)$, this would immediately imply an insecure merge algorithm with $O(n)$ processors and $o(\log \log n)$ time, contradicting the lower bound of [4] cited above. Thus the conjecture is true for the special case of secure merge protocols where each round of the protocol can be executed in $O(1)$ time on a CREW PRAM with $O(n)$ processors.

Both the Valiant and the Borodin-Hopcroft algorithms rely on the following basic construction: Split each list into blocks of size $\sqrt{n}$, with $\sqrt{n}$ medians. Running all pairwise comparisons between medians identifies which block of the opposite list each median is mapped to; then another round of pairwise comparisons identifies the exact position the median is mapped to within that block. This creates $\sqrt{n}$ subproblems of size $\sqrt{n}$, giving the recurrence $c(n) = \sqrt{n} \cdot c(\sqrt{n})$, if $c(n)$ is the cost of merging two lists of length n . With sufficiently many processors, this recurrence yields $O(\log \log n)$ run-time. Our protocols in §4 use similar techniques as a starting point, though many additional ideas are needed to handle the *secure* setting, e.g. securely handling the case where blocks from one list potentially span more than one block of the other list.

2.2 Secure Merge Algorithms

Notation. To aid the comparison to prior work, we introduce the following constants: κ is a computational security parameter (e.g., $\kappa = 128$ is standard), β_{HE} (resp. β_{FHE}) is the ciphertext expansion of an additively homomorphic (resp. fully homomorphic) cryptosystem, γ is the cost of decryption, and μ is the cost of multiplication in the FHE cryptosystem.

The parameters β_{HE} , β_{FHE} , γ , and μ depend on the cryptosystem and on κ . Asymptotically, $\kappa \gg \log n$ is necessary so that a random bit-string of length κ cannot be guessed in the time it takes to traverse the list, and (for suitable choice of cryptosystems) β_{HE} and β_{FHE} approach 1 as the plaintext size grows. In practice, fully homomorphic encryption schemes are more costly than additive-only homomorphic ones, so we expect $\beta_{\text{FHE}} > \beta_{\text{HE}}$ and $\mu > \gamma$.

We assume the objects to be sorted are contained in $O(1)$ memory words of size W bits. Unless explicitly stated otherwise, our communication and computational complexity numbers are given in terms of memory words and primitive operations on memory words.

Security via Generic Transformation. We explore here two naïve solutions for transforming an *insecure* merge algorithm to a secure one (see Table 1 for a succinct comparison of our secure merge protocol to these naïve solutions):

GARBLED CIRCUITS. By choosing any (insecure) sorting algorithm that can be represented as a circuit, the parties can use garbled circuits to (securely) sort their list in $O(1)$ rounds. As discussed above in §2, for comparison-based merging networks, $\Omega(n \log n)$ comparisons and $\Omega(\log n)$ depth are necessary, and achievable by the Batcher merging network [3].

We note that obtaining κ bits of computational security when merging lists whose elements have size W bits requires $\kappa \cdot W$ bits, or κ words of communication for each comparison. Thus, obtaining secure merge via a garbled circuit approach (for a circuit representing a merging network) would result in a constant-round protocol with $\Omega(\kappa \cdot n \log n)$ communication.

On the other hand, using GMW-style circuit evaluation (see [12]) instead of garbled circuits can reduce the overhead of each comparison (from κ down to constant), but it incurs a hit in round complexity (proportional to the depth of the circuit instead of constant-round).

FULLY HOMOMORPHIC ENCRYPTION (FHE). An $O(1)$ round protocol can also be constructed by having one party encrypt their inputs under FHE and send the result to the other party, who performs the desired calculations on ciphertexts. The other party then subtracts a vector of random values $\mathbf{r}$ from the (encrypted) merged list and sends the result back to the first party, who decrypts to obtain the merged list shifted by $\mathbf{r}$. Now the parties hold an additive sharing of the sorted list.

As with garbled circuits, however, the calculations the second party performs on the ciphertexts must be input-independent (in order to avoid information leakage), and so the calculation must be represented by a circuit. This means that communication is (asymptotically) lower than the garbled circuit approach, since the first party need only provide ciphertexts of his list (which correspond to *inputs* to the circuit), as opposed to providing information for each *gate*. Therefore the communication required for FHE is $O(\beta_{\text{HE}} n)$.

However, while communication in the FHE approach may be (asymptotically) reduced (compared to the garbled circuit approach), notice that the computation is still $\Omega(\mu n \log n)$, where μ is the cost of a multiplication under FHE, as the circuit requires $\Omega(n \log n)$ comparison (multiplication) operations. Additionally, since the circuit has depth $\Omega(\log n)$, FHE will require bootstrapping to avoid ciphertext blowup, which will be expensive in practice.

ORAM AND GRAM. ORAM incurs at least an $\Omega(\log n)$ overhead [13, 19]. Currently known GRAM constructions are built using ORAM as a building block. Therefore, if one is to take an insecure merge implementation and try to compile it into a secure circuit using ORAM or GRAM, both the computational and communication complexity will be $O(n \log n)$, and thus these techniques are inapplicable if we aim to achieve linear complexity.

Shuffle-Sort Paradigm. One challenge facing any comparison-based *secure* merge (or sort) protocol is that the results of each comparison must be kept secret from each party, or else security is lost. One approach to allowing the results of the comparisons to be known *without* information leakage is to first *shuffle* (in an oblivious manner) the input lists. However, since shuffling will destroy the property that the input lists are pre-sorted, this approach reduces a secure *merge* problem to a secure *sort* problem, hence incurring the $\log n$ efficiency loss.

Comparison-based sorting has $O(n \log n)$ communication and computation and $O(\log n)$ rounds, while secure shuffling requires $O(n)$ communication and $O(1)$ rounds, see e.g., [11, 15]. Therefore, a secure sort using the shuffle-sort paradigm requires $O(n \log n)$ communication and computation and $O(\log n)$ rounds, which is worse than our results in all three metrics.

Oblivious Sort. Because the constant from [1] is too large for practical applications, a number of other approaches to secure sorting have been explored. The shuffle-sort paradigm mentioned above is one example of a large family of *oblivious* sort protocols, which allow for a variable memory access pattern as long as it is independent of the underlying list values, or *data oblivious*. We mention here the radix sort of Hamada et al. [14], which achieves $O((W \log W + W)n + n \log n)$ communication (in memory words) and $O(1)$ round complexity in the three-party honest majority setting with constant bit lengths of elements. The communication complexity was later improved by Chida et. al [8], to $O(n \log n)$ memory words. However, the round complexity depends linearly on W , so when $W \approx \log n$, this matches the round complexity of the other protocols.

Secure Merge Protocols. There are several works that investigate secure merge directly. The first, due to Falk and Ostrovsky [11], achieves $O(n \log \log n)$ communication complexity with $O(\log n)$ round complexity. The second, due to Falk, Nema, Ostrovsky [10], achieves the asymptotically optimal $O(n)$ communication complexity (with small constants), but requires $O(n)$ rounds of communication. For many cryptographic applications, a high round complexity causes more of a bottleneck than a high communication complexity due to network latency. Therefore, the secure merge protocol of [10], while both simple and asymptotically optimal in terms of communication, may still not be practical due to high round complexity.

In a subsequent work [6], Chakraborty et al. present a series of secure merge protocols, optimizing for concrete efficiency. Some of their protocols use similar machinery to our work presented here – and indeed, their paper cites our work here as inspiring some of their ideas. In [6], the authors explore the trade-offs between communication and rounds, giving protocols with worse asymptotic performance than our protocols. See the full version for a discussion of the concrete efficiency of our protocols and comparisons with standard (insecure) merge protocols.

Three Party Sort and Merge protocols. The three party honest majority setting is a natural fit for real-world protocols in the client-server model, including ORAM and PSI protocols. Prior work in this area has shown how the use of three parties facilitates more efficient protocols, including through the use of one party to generate randomness for the other parties and more efficient shuffling protocols (see e.g., [11]). The oblivious sort protocols mentioned above [8, 14] use three parties for shuffling and to enable the use of a Shamir secret sharing scheme. In [7] Chan et al. give a three-server merge protocol in the course of building a three-server ORAM scheme. This merge protocol, which requires three servers and an honest client, is most similar to the FNO protocol [10], and similarly requires $O(n)$ round complexity.

2.3 Comparison of Results

In Table 1 we give the communication, computation, and round complexity of our secure merge protocols, in comparison with the approaches described above. Our main result is a secure merge protocol that has (optimal) communication and computation $O(n)$, and round complexity $O(\log \log n)$ (see Theorem 1). Notice that $O(\log \log n)$ round complexity is superior to all other protocols in Table 1 *except* the constant-round garbled circuit and FHE approaches, each of which is inferior in the other two metrics (computation and communication). We also describe an asymmetric merge protocol on lists of size (n^α, n) for fixed $\alpha < 1$ that achieves the same asymptotic complexity as our general secure merge protocol, but in only $O(1)$ rounds.

The parameters β_{HE} and γ arise out of the use of homomorphic encryption for two-party shuffle, and so are only relevant for comparing secure merge protocols against two-party merge networks with standard MPC. Namely, in the three-party setting (which is assumed for [8, 15]), we can set $\beta_{\text{HE}} = \gamma = 1$ for the last four protocols of Table 1.

■ **Table 1** Comparison of secure merge protocols, with parameters as in §2.2; namely: n is the list size, $\alpha < 1$ is any fixed constant, κ is a computational security parameter, μ is the cost of FHE multiplication, β_{HE} and β_{FHE} are ciphertext expansions, and γ is the decryption cost ($\beta_{\text{HE}} = \gamma = 1$ in the three-party setting). We set the word size $W = \Theta(\log n)$ to simplify the formulas.

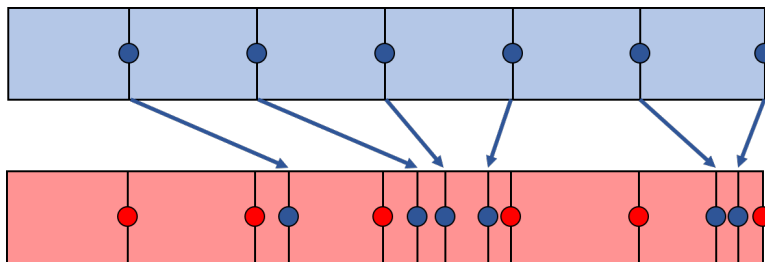
Protocol	Computation	Communication	Rounds
(Garbled) Merge Network [Folklore]	$O(\kappa \cdot n \log n)$	$O(\kappa \cdot n \log n)$	$O(1)$
(GMW) Merge Network [Folklore]	$O(n \log n)$	$O(n \log n)$	$O(\log n)$
(FHE) Merge Network [Folklore]	$O(\mu \cdot n \log n)$	$O(\beta_{\text{FHE}} n)$	$O(1)$
Shuffle-Sort [15]	$O(n \log n)$	$O(n \log n)$	$O(\log n)$
Oblivious Radix Sort [8]	$O(n \log n)$	$O(n \log n)$	$O(\log n)$
Secure Merge of [11]	$O(n \log \log n + \gamma n)$	$O(n \log \log n + \beta_{\text{HE}} n)$	$O(\log n)$
Secure Merge of [10]	$O(\gamma n)$	$O(\beta_{\text{HE}} n)$	$O(n)$
Our Secure (n^α, n) Merge [Fig. 5]	$O(2^{1/(1-\alpha)^3} \gamma n)$	$O(2^{1/(1-\alpha)^3} \beta_{\text{HE}} n)$	$O(1/(1-\alpha)^3)$
Our Secure (n, n) Merge [Fig. 4]	$O(\gamma n)$	$O(\beta_{\text{HE}} n)$	$O(\log \log n)$

While our primary focus in this paper is on asymptotic performance (rounds, communication, computation), our protocols have many degrees of freedom that allow customization for concrete performance as well. By making different choices based on actual conditions (list size, network bandwidth, etc.) – in terms of which sub-protocols to use and when to stop recursing between our high-level merge protocols – we can give specializations of our protocol that provide high concrete performance. For example, our protocol will outperform a generic instantiation of the secure merge protocol that is based on the Batcher odd-even merge network, in both round complexity and the number of secure comparisons – see the full version for details.

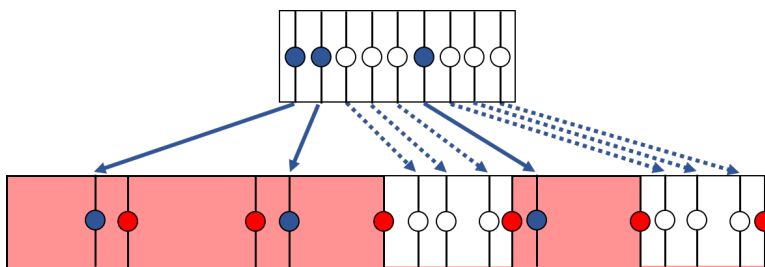
3 Overview of Techniques

At a high level, the strategy of our secure merge protocols is to *partition* the original lists into several blocks, then *align* these blocks (find the appropriate block(s) from the other list that span the “same” range of values); perform secure merge on these blocks; and finally combine (concatenate) the blocks together to obtain the final merged list. Ignoring for the moment issues in aligning the blocks (e.g. a block from one list spanning numerous blocks from the other list), this “partition-align” strategy – of using partitioning to achieve secure merge on

larger lists via several smaller secure merge subproblems – has the following appeal: If we partition the lists into k blocks, each with n/k elements, then for e.g. $k = n/\log \log n$, we could apply the linear protocol of [10] to each block. Since each block contains $n/k = \log \log n$ elements, each subproblem requires $O(\log \log n)$ communication, computation, and round-complexity. Furthermore, since each subproblem can be performed in parallel, the total cost across all $k = n/\log \log n$ blocks will be $O(n)$ computation and communication (and $O(\log \log n)$ rounds). This matches our target complexity in all three metrics.



■ **Figure 1** Merging the k medians (blue dots) from the top list into the correct locations with respect to the bottom list (the k medians of the bottom list are shown in red).



■ **Figure 2** Merging a smaller list (9 elements) into a larger list, and then classifying the elements in the first list and blocks from the second list that are “poorly-aligned.” Namely, poorly-aligned elements in the first list (depicted as white) means multiple (in this case three or more) elements from the first list map to the *same block* of the second list; and similarly the blocks in the second list that contain three or more elements from the first are poorly-aligned (and also depicted as white).

3.1 Identifying Poorly-Aligned Blocks

Of course, the above simplified strategy and analysis ignores several challenges that arise in practice:

Block Alignment. In order for the **partition-align** strategy to make sense when merging together two blocks (one from each list): Given a block (contiguous set of values) from one list, we must identify the appropriate block(s) from the other list that contains the “same” range of values.

Poorly-Aligned Blocks. If we partition say the first list into equal block sizes of n/k elements, the simplified analysis above assumed these blocks precisely align with exactly one block from the second list. In practice, a given block from the first list can align with arbitrarily many blocks from the second list, including e.g. the extreme case where a single block from the first list has a range of values that encompasses the entirety of the second list (see Figure 2).

Obliviousness of Alignment. Notice that being able to observe how the blocks from one list overlap with the blocks of the other list can leak information about the (relative) values on each list, thereby compromising overall security. In particular, any secure merge protocol employing the **partition-align** approach must hide all information regarding the nature of the alignments.

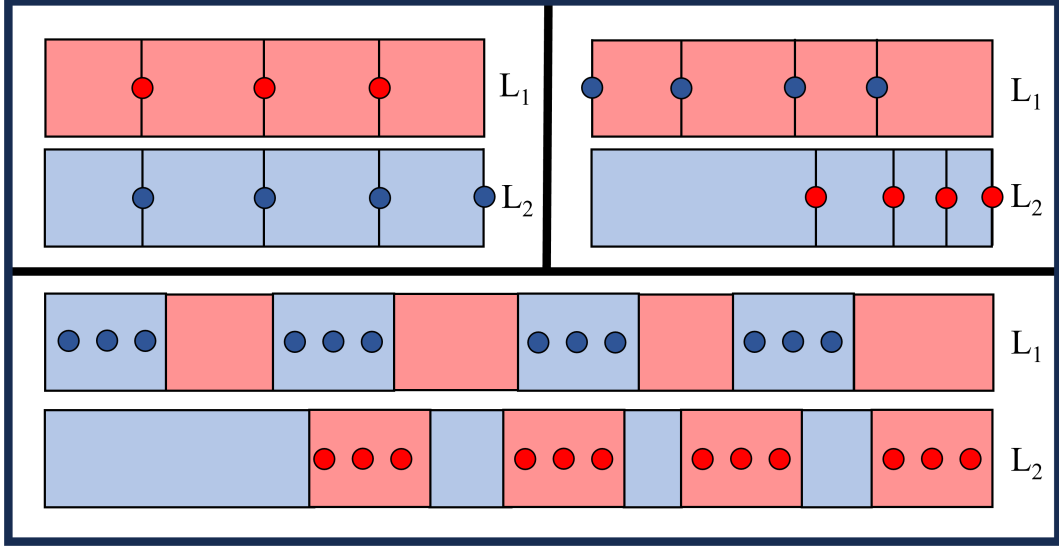
Our main contribution in this paper can be viewed as resolving the above three challenges:

1. We resolve the “Block Alignment” challenge by running a secure merge protocol to identify where the k partition points (“medians”) of one list belong within the second list. This requires a secure *asymmetric* (since one of the lists – of size k – is smaller than the other list – of size n) merge protocol. However, since this protocol is only invoked twice (once in each direction to merge the k medians of one list into the other), it can have complexities proportional to the *larger* list size n and still achieve our overall target metrics; we leverage this fact in our secure asymmetric merge protocol (Figure 5).
2. We resolve the “Poorly-Aligned Blocks” challenge as follows. First, we classify a block as “poorly-aligned” if (the range of values in) it spans “too many” blocks of the other list (or vice-versa). Then for merging poorly-aligned blocks, we again employ a secure asymmetric merge (merging one block from one list with multiple blocks from the other list). Depending on how “poor” the alignment (e.g. a single block from one list might span *all* blocks of the other list), this secure asymmetric merge protocol might have the larger list of size comparable to the original list.
3. We resolve the “Obliviousness of Alignment” challenge in three ways. First, we hide the classification of which blocks are well/poorly-aligned. Second, for the merging of “well-aligned” blocks, we apply a remarkable lemma (Lemma 2 below) about k -medians that allows the introduction of dummy elements to each block to ensure they are perfectly aligned, and then merge the resulting (slightly larger) blocks (see Figure 3). Third, for the poorly-aligned blocks, we observe that a “worst possible” alignment scenario can be defined, which provides a bound on the number and nature of the poorly-aligned blocks. In particular, we bound the number of highly *unbalanced* invocations of the secure asymmetric merge protocols – those in which a single block from one list spans *many* blocks from the other – which are costly to run. In particular, we always perform the same set of secure asymmetric merges (i.e. for a (fixed, known) set of list sizes) for the poorly-aligned blocks, regardless of how many blocks are actually classified as poorly-aligned (and the specific nature of their “poor” alignment).

► **Lemma 2.** *Let L_1 and L_2 denote two (sorted) lists of size n , and let $\mathcal{M}_{1,k}$ and $\mathcal{M}_{2,k}$ denote their k medians. Let $L'_1 = L_1 \sqcup \mathcal{M}_{2,k}$ denote the list (of size $2n$) resulting from merging $\mathcal{M}_{2,k}$ with L_1 , with each element in $\mathcal{M}_{2,k}$ duplicated n/k times in L'_1 . Let $\mathcal{M}'_{2k}$ denote the $2k$ medians of L'_1 . Then the $2k$ medians of L'_1 are exactly the (merged) k medians of L_1 and L_2 : $\mathcal{M}'_{2k} = \mathcal{M}_{1,k} \sqcup \mathcal{M}_{2,k}$.*

While several details need to be worked out – such as tuning parameters and specifying how to handle dummy values and superfluous merging of (phantom/ non-existent) poorly-aligned blocks, the above strategy describes our high level approach. After introducing the notation that will be used in the remainder of this paper, we next go into more detail on the specific techniques and subroutines that will be employed in our secure merge protocols.

For a given block from one list, we need to identify which block(s) from the other list span a similar range of values. We first count the number of blocks from the second list that span the range of values in the first list, and then classify the blocks as “poorly-aligned” if the number of blocks from the second list is too large (greater than some constant). We



■ **Figure 3** The *Expanding Medians* strategy for converting “well-aligned” to *perfect* alignment: The k -medians of L_1 and L_2 are identified (top-left) and merged into the opposite list (top-right). Then each median is duplicated in place $\frac{n}{k}$ times (bottom), and the resulting lists will consist of $2k$ blocks that are perfectly aligned (can be merged in parallel).

then collect all poorly-aligned blocks/elements from L_1 and L_2 (marked white in Figure 2) and treat them as an additional subproblem, after padding with dummy elements to avoid leaking information, and explain how to bound the number of poorly-aligned blocks in §5.2.

3.2 The Tag-Shuffle-Reveal Paradigm

We make repeated use of the *tag-shuffle-reveal* paradigm, which should be considered analogous to the shuffle-sort paradigm of prior sorting protocols, see e.g. [11, 15], or an extension of the shuffle-reveal paradigm of [14]. Succinctly, the tag-shuffle-reveal paradigm starts with each element of a list (obviously) tagged with some label. This label can be (a secret sharing of) its current index, or it can be the result of some multiparty computation, for example a bit representing the output of a comparison against another value. Then, after shuffling the list, the tag is revealed, and the list entries are rearranged accordingly. Because the shuffle step ensures that the tags are randomly ordered, the only requirement to ensure security is to ensure that the set of values the revealed tags take on do not depend on the underlying data. We use $\hat{x}_i$ to denote the element x in position i after shuffling.

3.3 Extraction Protocols

One central application of the tag-shuffle-reveal paradigm is our collection of *extraction protocols* for pulling marked elements from a larger list into a smaller list or set of lists. Marked elements can be extracted and kept in their original order ($\Pi_{\text{Ext-Ord}}$), or extracted and shuffled (Π_{Extract}), or extracted into bins based on a tag they are marked with ($\Pi_{\text{Ext-Bin}}$). Of course, each of these protocols should reveal nothing about the location or number of tagged elements, and so the outputs will be padded with dummy elements where necessary. We use extraction protocols in a black box way in §4-6; full descriptions and proofs of the Π_{Extract} , $\Pi_{\text{Ext-Ord}}$, and $\Pi_{\text{Ext-Bin}}$ protocols can be found in the full version.

3.4 Primitive Functionalities

We recall the secure merge protocol from [10] discussed above, which requires $O(n)$ communication and $O(n)$ rounds, which we call $\Pi_{\text{SM-FNO}}$ in this work. Additionally, we use the trivial $O(n^2)$ communication, $O(1)$ round merge protocol, which makes every possible pair of comparisons. We call this protocol $\Pi_{\text{SM-ALL}}$ (see the full version for a full description of this protocol and a proof of its properties).

The protocols we present below are realized with black box calls to five “primitive” functionalities: Π_{Reveal} , Π_{Comp} , Π_{Sel} , Π_{Shuffle} , Π_{Dup} that act on additively shared secret values. Oblivious transfer and the existence of an additively homomorphic rerandomizable public key cryptosystem are sufficient assumptions to realize these functionalities (and $\Pi_{\text{SM-FNO}}$), as described below.

In Π_{Reveal} , parties begin with secret-shares of a value $[x]$ and end with the value x .

In Π_{Comp} , parties have shares of two values x and y , and as output they receive shares of a bit denoting the result of a comparison operator $[x > y]$, $[x \geq y]$ or $[x == y]$. These comparisons can be computed by converting to shares of bits and applying garbled circuits, which requires at least $O(W \log W)$ boolean gates on words of W bits. A promising alternative is the GMW-based approach of Nishide and Ohta [22], which requires $O(W)$ bits of communication and $O(1)$ rounds of communication and is concretely efficient.

In Π_{Sel} , parties perform multiplication of two values $[b]$ and $[x]$, where b is either 0 or 1, and x can be any value. Equivalently, parties compute shares of the ternary operator $b ? x : 0$, where b is either XOR shared or additively shared over a larger field. Π_{Sel} can be realized using MPC multiplication or via string-OTs with rerandomizable encryption.

In Π_{Shuffle} , parties begin with shares of a list, and end with shares of the same list, in a new (unknown) order. To get the desired asymptotics, we require that the shuffle have $O(1)$ rounds and $O(n)$ total communication. Such protocols exist and can be realized via an additively homomorphic, semantically secure cryptosystem with constant ciphertext expansion. At a high level, the shuffle works by allowing each party to hold an encryption of the list under the other party’s secret key, and then shuffling and re-randomizing; see [11] for the full protocol and a more thorough treatment.

In Π_{Dup} , parties begin with shares of a list of size n in which exactly k elements are marked with a given “tag” value (the tag value and the number of elements k that have this value are both public knowledge). After this protocol, each element with the specified tag value has been duplicated m times (for arbitrary public integer m), so that the final list size is $n + km$. We demonstrate in the full version how Π_{Dup} can be instantiated via two invocations of Π_{Shuffle} and n invocations of Π_{Comp} .

3.5 Security Model

We provide security analyses of our protocols within the Universally Composable (UC) framework, against a semi-honest adversary corrupting one of the two parties. UC security is essential since our protocols are built on recursive calls to sub-protocols. We present the protocol in the input setting where each party holds shares of each list being sorted, although, as mentioned in the introduction, the protocol can be adapted to other, more specific settings. We remark that the adversary does not have direct access to the memory access pattern of the other party. However, outside of the shuffling protocol, both parties have identical memory access patterns, and so the adversary can deduce most of the memory accesses of the honest party, and our proofs of security show the adversary learns nothing from these memory accesses.

We prove UC security of the protocols in this paper against static semi-honest adversaries under the standard simulator definition of security, see e.g. [5]. It is straightforward to simulate the behavior of the adversary during the protocol in any environment, since the adversary is semi-honest and must follow the protocol. What remains to be checked is the behavior of the adversary on the input, i.e. that the adversary input is still extractable without the simulator being able to “rewind” the adversary. We address this in the standard way by requiring both parties to commit to their inputs under an extractable commitment, so that the adversary input can be recovered without rewinding. We omit the details.

An ideal functionality $\mathcal{F}$ for each protocol interacts with the parties in the following way:

- **Setup.** Each party sends their inputs to $\mathcal{F}$, who stores them plus an id sid .
- **Execution.** Each party sends the command $(\text{Execute}, sid)$ to $\mathcal{F}$, who computes the desired output and stores it.
- **Reveal.** Each party sends the command (Reveal, sid) to $\mathcal{F}$, who sends the output of the execute step to each party.

Note that we could instead combine multiple ideal functionalities into a black-box functionality $\mathcal{F}^{\text{black-box}}$ with a family of execution commands (Execute_i) corresponding to each of the protocols defined in this paper. This provides an alternate way to address composability, and guarantees that inputs to one protocol match outputs from another protocol.

3.6 Notation

For any sorted lists L_1 and L_2 , let $\sqcup$ denote the “merge” of two lists (i.e. $\sqcup$ is functionally equivalent to (multi-)set union followed by sort): $L_1 \sqcup L_2 = \text{Sort}(L_1 \cup L_2)$. For any sorted list L_j of size n , and for any $k|n$, let $\mathcal{M}_{j,k}$ denote the k “medians” of L_j . Namely, if list $L_j = \{u_1, \dots, u_n\}$, then: $\mathcal{M}_{j,k} := \{u_{\frac{\ell \cdot n}{k}}\}_{\ell=1}^k$. Basic properties that follow from the definition of medians can be found in the full version.

Throughout the paper we distinguish between secure *symmetric* merge Π_{SSM} and secure *asymmetric* merge Π_{SAM} . For symmetric merge, the lists are of roughly the same size or differ by a constant factor (since one list can be padded with dummies to match the length of the other). For asymmetric merge, the ratio of list sizes is larger than constant.

4 Description of Secure Merge Protocols

Our merge protocols come in two variants: symmetric merge, where the two lists are of equal size, and asymmetric merge, where one list is significantly smaller than the other. One useful technique that we employ in both settings is using black box calls to one flavor of merge to solve the other. This iterative process then terminates with a “base” version of each variant (symmetric and asymmetric merge), and we introduce in §6 two efficient protocols – $\Pi_{\text{SSM-log log}}$ and $\Pi_{\text{SAM-}n^{1/3}}$ – that can be used as the base protocols, allowing us to achieve our overall target metrics for secure merge as stated in Theorem 1. We summarize the dependency of our main protocols on these subprotocols in Table 2. That table, together with the asserted metrics of the primitive functionalities (such as Π_{Extract}), is sufficient to establish the stated asymptotics of Theorem 1.

In this section, we present our secure symmetric and asymmetric merge protocols. Each follows the *partition-align* approach, which has four phases:

- **Partition.** We invoke a subprotocol to determine where the partition points (medians) of one list lie within the other list (see e.g. Figure 1).
- **Align Blocks.** A (lightweight) MPC protocol is run to determine, for each block in one list, which block(s) in the other list span the same range of values (see e.g. Figure 2).

■ **Table 2** Protocol and subprotocol relations. The Cost column uses:

- For $\Pi_{\text{SSM}}(n)$: Set $k = \frac{n}{\log \log n}$, $\Pi_{\text{SAM}}(k, n) = \text{Fig. 5}$, $\Pi_{\text{SSM}}(\frac{n}{k}) = \Pi_{\text{SM-FNO}}$.
- For $\Pi_{\text{SAM}}(k, n)$: $\Pi_{\text{SSM}}(k) = \Pi_{\text{SSM-log log}}(k)$ (Fig. 7), $\Pi_{\text{SSM}}(\frac{n}{k}) = \Pi_{\text{SM-FNO}}$.

Name	Calls to Subprotocols	Cost
$\Pi_{\text{SSM}}(n)$ [Fig. 4]	$2 \cdot \Pi_{\text{SAM}}(k, n), 2k \cdot \Pi_{\text{SSM}}(n/k)$	$O(n)$
$\Pi_{\text{SAM}}(k, n)$ [Fig. 5]	$2 \cdot \Pi_{\text{SSM}}(k), k \cdot \Pi_{\text{SSM}}(n/k)$	$O(n + k \log \log k)$
$\Pi_{\text{SSM-log log}}(n)$ [Fig. 7]	$O(\log \log n) \cdot \Pi_{\text{Shuf}}(n), O(n) \cdot \Pi_{\text{Rev,Comp,Sel}}$	$O(n \log \log n)$
$\Pi_{\text{SAM-n}^{1/3}}(n^{1/3}, n)$ [Fig. 9 (full version)]	$2n^{1/3} \cdot \Pi_{\text{SM-ALL}}(n^{1/3}, n^{1/3})$	$O(n)$

- **Merge Blocks.** This phase involves the merging of both the “well-aligned” blocks and the “poorly-aligned” blocks (see e.g. Figure 3 for well-aligned blocks).
- **Combine Blocks.** The results from the previous step are combined, and any dummy elements that were added are removed, producing the final merged list.

4.1 Secure Symmetric Merge

Our *Secure Symmetric Merge* protocol $\Pi_{\text{SSM}}(n)$ is sketched in Figure 4. For the Partition phase, it uses a secure asymmetric merge protocol to merge the k -medians of one list into the other list (and vice-versa). For the Align Blocks phase, we expand each median (in its merged location within the other list) into k -copies; which by Lemma 2 guarantees that each block of n/k elements is perfectly aligned (see Figure 3). Thus, there are no “poorly-aligned” blocks for the Merge Blocks phase, and each block is merged via a secure symmetric merge protocol (for lists of size n/k). Specification of our “abstract” secure symmetric merge protocol is presented in Figure 4; this protocol is made concrete by specifying choices for partition/block size k and the specific merge sub-protocols used in the Partition and Merge Blocks phases. In particular, the metrics of Theorem 1 are obtained by setting $k = n / \log \log n$ and using $\Pi_{\text{SSM}'}(n/k) = \Pi_{\text{SSM-FNO}}$ and $\Pi_{\text{SAM}}(k, n) = \Pi_{\text{SAM}}(k, n, \Pi_{\text{SSM-FNO}}, \Pi_{\text{SSM-log log}})$, which refers to the secure asymmetric merge protocol of Figure 5, using for its subprotocols $\Pi_{\text{SSM}'}(n/k) = \Pi_{\text{SSM-log log}}$ and $\Pi_{\text{SSM}''}(k) = \Pi_{\text{SSM-FNO}}$.

Secure Symmetric Merge Protocol

Input. Two parties $\mathcal{P}_1, \mathcal{P}_2$ (additively) secret-share two *sorted* lists L_1 and L_2 , each of size n . Also as input, a parameter k with $k|n$, and specifications of subprotocols $\Pi_{\text{SSM}'}(n/k)$ and $\Pi_{\text{SAM}}(k, n)$.

Output. The two lists have been merged into an output list $L_1 \sqcup L_2$, which has size $2n$ and is (additively) secret-shared amongst the two parties.

Protocol (sketch).

1. **Partition.** Invoke secure asymmetric merge protocol $\Pi_{\text{SAM}}(k, n)$ (Fig. 5) to merge the k medians of L_2 with list L_1 (and vice-versa).
2. **Align Blocks.** Expand Medians (Fig. 3) by running the duplicate values Π_{Dup} protocol twice, which expands the sizes of the output lists from Step 1 to be $2n$ and ensures they are “aligned”.
3. **Merge Blocks.** Run the secure symmetric merge protocol $\Pi_{\text{SSM}'}(n/k)$, in parallel, on each of the $2k$ (aligned) blocks.
4. **Combine Blocks.** Concatenate the results of the $2k$ parallel invocations of secure symmetric merge from the previous step, and run the secure ordered extract $\Pi_{\text{Ext-Ord}}$ protocol to remove dummy elements.

■ **Figure 4** Overview of our Secure Symmetric Merge Protocol. For the top-level protocol, use $k = n / \log \log n$, and use $\Pi_{\text{SAM}}(k, n, \Pi_{\text{SSM-FNO}}, \Pi_{\text{SSM-log log}})$ for the asymmetric merge protocol in Step 1, and use $\Pi_{\text{SSM-FNO}}$ for each of the symmetric merge protocols run in parallel in Step 3.

4.2 Secure Asymmetric Merge

Our *Secure Asymmetric Merge* protocol $\Pi_{\text{SAM}}(k, n)$ is sketched in Figure 5. For the Partition phase, it uses a secure symmetric merge protocol to merge the k medians of the larger list with the (entirety of the) smaller list. For the Align Blocks phase, we use the $\Pi_{\text{Ext-Bin}}$ protocol (§3.3) to securely classify elements/blocks as poorly-aligned vs. well-aligned. For the Merge Blocks phase: for the well-aligned items, we apply a secure symmetric merge protocol to merge (at most) n/k elements from L_1 into the appropriate block of n/k elements of L_2 . For the poorly-aligned items, we first argue that the cumulative number of elements in L_2 that lie in a poorly-aligned block is at most k (and the same is trivially true for L_1 , which only has k elements), and therefore we use another secure symmetric merge protocol (for lists of size k) to merge poorly-aligned elements from L_1 into poorly-aligned blocks of L_2 .

The “abstract” secure asymmetric merge protocol in Figure 5 is made concrete by specifying choices for the specific merge sub-protocols used in the Partition and Merge Blocks phases. In particular, the metrics of Theorem 1 are obtained by using $\Pi_{\text{SSM}'}(n/k) = \Pi_{\text{SSM-FNO}}$ and $\Pi_{\text{SSM}''}(k) = \Pi_{\text{SSM-log log}}$.

5 Analyses of Protocols in Section 4

In analyzing the performance of a protocol, RCost will denote the round-complexity, and CCost the computation and overall communication complexity.

5.1 Analysis of Abstract Secure Symmetric Merge Protocol of §4.1

Security. The security of the $\Pi_{\text{SSM}}(n)$ protocol follows immediately from the security of the underlying Π_{Dup} , $\Pi_{\text{Ext-Ord}}$, $\Pi_{\text{SSM}'}$, and Π_{SAM} protocols.

Correctness. Assuming correctness of $\Pi_{\text{SSM}'}(n/k)$, $\Pi_{\text{SAM}}(k, n)$, Π_{Dup} , and $\Pi_{\text{Ext-Ord}}$ subprotocols, we need only demonstrate that the concatenation done in Step 4 above is correct, that is, that the blocks “align” as per the partitioning. Namely, this follows from Lemma 2, which demonstrates that lists L_1'' and L_2'' have the same list of $2k$ medians (both equal $\mathcal{M}_{1,k} \sqcup \mathcal{M}_{2,k}$).

Cost. Step (1) invokes the $\Pi_{\text{SAM}}(k, n)$ protocol (Figure 5) twice; Step (2) invokes the $\Pi_{\text{Dup}}(k, n/k)$ protocol twice; Step (3) invokes the $\Pi_{\text{SSM}'}(n/k)$ protocol $2k$ times; and Step (4) invokes the secure ordered extract $\Pi_{\text{Ext-Ord}}(4n, 2n)$ protocol. Using the $\Pi_{\text{Dup}}(k, n/k)$ and the $\Pi_{\text{Ext-Ord}}(4n, 2n)$ protocols, and assuming constant-round and linear secure protocols for Π_{Comp} , Π_{Reveal} , and Π_{Shuffle} , adding up these costs yields:

$$\begin{aligned} \text{RCost}(\Pi_{\text{SSM}}(n)) &= \text{RCost}(\Pi_{\text{SAM}}(k, n)) + \text{RCost}(\Pi_{\text{Dup}}(k, n/k)) + \\ &\quad \text{RCost}(\Pi_{\text{SSM}'}(n/k)) + \text{RCost}(\Pi_{\text{Ext-Ord}}(4n, 2n)) \\ &= O(1) + \text{RCost}(\Pi_{\text{SAM}}(k, n)) + \text{RCost}(\Pi_{\text{SSM}'}(n/k)) \\ \text{CCost}(\Pi_{\text{SSM}}(n)) &= 2 \cdot \text{CCost}(\Pi_{\text{SAM}}(k, n)) + 2 \cdot \text{CCost}(\Pi_{\text{Dup}}(k, n/k)) + \\ &\quad 2k \cdot \text{CCost}(\Pi_{\text{SSM}'}(n/k)) + \text{CCost}(\Pi_{\text{Ext-Ord}}(4n, 2n)) \\ &= 2 \cdot \text{CCost}(\Pi_{\text{SAM}}(k, n)) + 2k \cdot \text{CCost}(\Pi_{\text{SSM}'}(n/k)) + O(n) \end{aligned}$$

Using $\Pi_{\text{SSM-FNO}}$ for $\Pi_{\text{SSM}'}(n/k)$, and using for $\Pi_{\text{SAM}}(k, n)$ our protocol of Fig. 5 – with subprotocols $\Pi_{\text{SSM-FNO}}$ for $\Pi_{\text{SSM}'}(n/k)$ and $\Pi_{\text{SSM-log log}}$ for $\Pi_{\text{SSM}''}(k)$ – and using $k = n/\log \log n$, the cost becomes:

Secure Asymmetric Merge Protocol $\Pi_{\text{SAM}}(k, n)$

Input. Two parties $\mathcal{P}_1, \mathcal{P}_2$ (additively) secret-share *sorted* list L_1 of size k and L_2 of size n . Also, specification of subprotocols $\Pi_{\text{SSM}'}(n/k)$ and $\Pi_{\text{SSM}''}(k)$.

Output. The two lists have been merged into an output list $L_1 \sqcup L_2$, which has size $k + n$ and is (additively) secret-shared amongst the two parties.

Protocol.

1. **Partition**(Fig. 1). Run $\Pi_{\text{SSM}''}(L_1, \mathcal{M}_{2,k})$ to merge L_1 and the k medians of L_2 .
2. a. **Align Blocks: Label**(Fig. 2). Partition L_2 into k blocks, each of size n/k (the k medians of L_2 define the right-boundary of each block). Define a block of L_2 to be “well-aligned” if it has fewer than n/k elements of L_1 that map to it. Well-aligned blocks are identified by doing (in parallel, via Π_{Comp}) a linear scan of the merge positions of L_1 (from Step 1), and comparing whether elements i and $(i + n/k + 1)$ are in the same block. Meanwhile, define elements of L_1 to be “well-aligned” if they lie in a well-aligned block.
 - b. **Align Blocks: Extract Poorly-Aligned**(Fig. 2). Using the merge positions from the previous step, run the $\Pi_{\text{Ext-Ord}}$ protocol on L_1 (respectively on L_2) to extract all “poorly-aligned” (i.e. *not* well-aligned) elements of L_1 (resp. L_2), and let L_1^{PA} (resp. L_2^{PA}) denote the extracted elements. Note that there are at most k poorly-aligned elements from each list (Observation 4), so L_1^{PA} and L_2^{PA} will each have size k , with $\Pi_{\text{Ext-Ord}}$ extracting dummy-elements to pad each list to exactly this size.
 - c. **Align Blocks: Extract Well-Aligned**(Fig. 2). For the well-aligned elements of L_1 , run the $\Pi_{\text{Ext-Bin}}$ protocol to extract the well-aligned elements of L_1 into the separate lists (each of size n/k), based on which block of L_2 they lie in (this information is available from the merge done in Step 1). By definition of “well-aligned,” there are at most n/k such elements for each block, and $\Pi_{\text{Ext-Bin}}$ extracts exactly this many elements into each list, padding with dummy elements when necessary. Let $\{L_{1,m}^{WA}\}_m$ denote the k output lists of the $\Pi_{\text{Ext-Bin}}$ protocol. Meanwhile, for L_2 , we simply extract all elements of the i^{th} block into $L_{2,i}^{WA}$ if and only if block i is well-aligned (otherwise $L_{2,i}^{WA}$ is filled with dummy elements). Using the labels created in Step 2, this can be done by running the $\Pi_{\text{Ext-Ord}}$ protocol on each block of L_2 .
3. a. **Merge Blocks: Poorly-Aligned.** Run $\Pi_{\text{SSM}''}(k)$ on L_1^{PA} and L_2^{PA} .
 - b. **Merge Blocks: Well-Aligned.** Run $\Pi_{\text{SSM}'}(n/k)$, in parallel, on each of the k (aligned) blocks $\{L_{1,m}^{WA}\}_m$ and $\{L_{2,m}^{WA}\}_m$.
4. **Combine Blocks.** The final index of any element in the merged list is:

$$(\#Left_1^{WA}) + (\#Left_1^{PA}) + (\#Left_2^{WA}) + (\#Left_2^{PA}) + \text{Block_Index}$$

where e.g. $\#Left_1^{WA}$ denotes the number of well-aligned elements of L_1 that lie in a block to the left of this element’s block, and Block_Index is this element’s index within its block. Each element’s final index is computed from the outputs of the merges done in Steps 1, 3a, and 3b, as per the analysis in the Correctness argument (Section 5.2).

■ **Figure 5** Secure Asymmetric Merge Protocol. In our main protocol, we take $k = n/\log \log n$, $\Pi_{\text{SSM}'} := \Pi_{\text{SSM-FNO}}$, and $\Pi_{\text{SSM}''} := \Pi_{\text{SSM-log log}}$.

$$\begin{aligned} \text{RCost}(\Pi_{\text{SSM}}(n)) &= [\text{RCost}(\Pi_{\text{SSM-log log}}(k) + \text{RCost}(\Pi_{\text{SSM-FNO}}(n/k))] + \\ &\quad \text{RCost}(\Pi_{\text{SSM-FNO}}(n/k)) \\ &= O(\log \log k) + O(n/k) = O(\log \log n). \\ \text{CCost}(\Pi_{\text{SSM}}(n)) &= 2 \cdot [2 \cdot \text{CCost}(\Pi_{\text{SSM-log log}}(k) + k \cdot \text{CCost}(\Pi_{\text{SSM-FNO}}(n/k))] + \\ &\quad 2k \cdot \text{CCost}(\Pi_{\text{SSM-FNO}}(n/k)) + O(n) \\ &= [O(k \log \log k) + k \cdot O(n/k)] + 2k \cdot O(n/k) + O(n) = O(n). \end{aligned}$$

5.2 Analysis of Abstract Secure Asymmetric Merge Protocol of §4.2

For all steps in this subsection, we refer to Figure 5.

Security. For Steps 1, 2a, 2b, 2c, 3a, and 3b: security follows from the security of the underlying subprotocols. Namely, a simulator for the protocol for either party calls the simulator for each subprotocol: $\Pi_{SSM''}$, Π_{Comp} , $\Pi_{Ext-Ord}$, $\Pi_{Ext-Bin}$, $\Pi_{SSM''}$, and $\Pi_{SSM'}$, respectively. Meanwhile, the correctness property ensures that the indices revealed in Step 4 are unique and are a (random) permutation of the values in $[1, \dots, (k+n)]$. Therefore, the simulator for Step 4 generates a random permutation of $(k+n)$ elements.

Correctness. We first clarify that the merges done in Steps 1, 3a, and 3b are done “in-place”: rather than actually *constructing* an output merged list, these merges instead determine each element’s *index* in what would be the merged list, and then append (shares of) this index as a tag applied to the element (in its original list). In this way, we may manipulate (add, subtract, etc.) the indices produced by the merges in Steps 1, 3a, and 3b, to compute each element’s index in the final merged list based on its indices in the outputs of the earlier “in-place” merges.

To verify correctness, it will be useful to set notation corresponding to the formula in Step 4 of Figure 5 as follows:

$$\begin{aligned} (U, V) &= (\#Left_1^{WA}, \#Left_1^{PA}): (\#(L1.WA), \#(L1.PA)) \text{ in all blocks to the left} \\ (W, X) &= (\#Left_2^{WA}, \#Left_2^{PA}): (\#(L2.WA), \#(L2.PA)) \text{ in all blocks to the left} \\ (Y, Z) &= (\#Same_1, \#Same_2): (\#L_1, \#L_2) \text{ in same block but to the left} \end{aligned}$$

where $Block_Index = Y + Z$ denotes the index of an element in its own block. Thus, any element’s final index in the merged list is: $U + V + W + X + Y + Z$. This quantity can be computed for each element based on information available from Steps 1-3, as follows:

$L_1.WA$: $(U + V + Y)$ is available from each element’s original position in L_1 ; $(W + X)$ is available as $j \cdot n/k$, where j is the block of L_2 that this element maps to (from Step 1); $(Y + Z)$ is the output index from Step 3b; and $-Y$ is this element’s position after merging its well-aligned block (Step 2c).

$L_2.WA$: $(U + V)$ is available from Step 2a; $(W + X)$ is available as $j \cdot n/k$, where j is the block of L_2 that this element lies in; and $(Y + Z)$ is the output index from Step 3b.

$L_1.PA$: $(U + V + Y)$ is available from each element’s original position in L_1 ; $(W + X)$ is available as $j \cdot n/k$, where j is the block of L_2 that this element maps to (from Step 1); $(V + X + Y + Z)$ is the output index from Step 3a; $(-V - Y)$ is from Step 2b; and $-X$ is n/k times the number of poorly-aligned blocks to the left, which is computable from the info in Step 2a.

$L_2.PA$: $(U + V)$ is available from Step 2a, while $-V$ is available by using the information from Step 1 applied just to the poorly-aligned items extracted in Step 2b; $(W + X)$ is available as $j \cdot n/k$, where j is the block of L_2 that this element lies in; $(V + X + Y + Z)$ is the output index from Step 3a; and $-X$ is n/k times the number of poorly-aligned blocks to the left, which is computable from the info in Step 2a.

It remains to show that the pre-conditions of the $\Pi_{Ext-Ord}$ and $\Pi_{Ext-Bin}$ protocols are satisfied. Namely, how to (efficiently) construct the input parameters C and T' of the $\Pi_{Ext-Bin}$ protocol; and that at most n/k elements are extracted from each list into each well-aligned block and at most k elements are extracted from each list into the poorly-aligned block. These are stated and proved in the following observations:

► **Observation 3.** (*Shares of*) the parameters C and T' to the $\Pi_{Ext-Bin}$ protocol used in Step 2c can be securely computed locally in $O(n)$ computation.

Proof. For each $i \in [1..k]$, let ι_i denote the position of the k^{th} median of L_2 in the output list of Step 1; and let δ_i be an indicator on whether $t_i > 0$. Then formulas for $C = \{c_j\}$ can be expressed iteratively as: $c_1 = \delta_1$ and $c_{j+1} = \delta_{j+1} \cdot (1 + \sum_{i \leq j} t'_i)$; and for $T' = \{t'_j\}$ as: $t'_1 = \iota_1 - 1$ and $t'_{j+1} = \iota_{j+1} - j - \sum_{i \leq j} t'_i$. While these expressions can be computed securely without introducing additional overhead to the overall protocol, the multiplication by δ_i in the expression of c_{j+1} would require a secure (non-local) computation. However, we leverage the fact that the $\Pi_{\text{Ext-Bin}}$ protocol (see full version) works correctly even if c_j is an arbitrary value when $t'_j = 0$. Thus, the δ_i terms can be dropped from the above expression, and then, by linearity of the secret-sharing scheme, each party can locally compute their shares of C and T' by using their shares of the relevant variables on the RHS of each of the above expressions for t'_j and c_j . $\blacktriangleleft$

► **Observation 4.** For both L_1 and L_2 , the number of poorly-aligned elements in Step 2b of the $\Pi_{\text{SAM}}(k, n)$ protocol of Figure 5 is at most k .

Proof. Since L_1 has size k , this statement is trivially true for L_1 . Meanwhile, for each poorly-aligned segment of L_2 , by definition there are at least $n/k + 1$ elements from L_1 that are assigned this segment. Thus, the k elements of L_1 can cause at most $m = k/(n/k + 1) < k^2/n$ poorly-aligned segments. Since each segment has size n/k , this corresponds to at most $m \cdot n/k < k$ elements of L_2 . $\blacktriangleleft$

► **Observation 5.** For both L_1 and L_2 , the number of well-aligned elements in each block of Step 2c of the $\Pi_{\text{SAM}}(k, n)$ protocol of Figure 5 is at most n/k .

Proof. Blocks are defined as the n/k elements to the left of (and including) each (of the k) median of L_2 , so this is trivially true for L_2 . Meanwhile, for L_1 , any block that contains more than n/k elements of L_1 will be a poorly-aligned block, and consequently the corresponding elements from L_1 will all be labelled “poorly-aligned,” which means *no* elements from L_1 will be extracted by $\Pi_{\text{Ext-Bin}}$ in Step 2c for this block. $\blacktriangleleft$

Cost.

- Step (1) has $\text{RCost}(\Pi_{\text{SSM}''}(k))$ and $\text{CCost}(\Pi_{\text{SSM}''}(k))$.
- Step (2a) has $\text{RCost}(\Pi_{\text{Comp}})$ and $\Theta(k) \cdot \text{CCost}(\Pi_{\text{Comp}})$.
- Step (2b) has $\text{RCost}(\Pi_{\text{Ext-Ord}}(k, k)) + \text{RCost}(\Pi_{\text{Ext-Ord}}(n, k))$ and $\text{CCost}(\Pi_{\text{Ext-Ord}}(k, k)) + \text{CCost}(\Pi_{\text{Ext-Ord}}(n, k))$.
- Step (2c) has $\text{RCost}(\Pi_{\text{Ext-Bin}}(k, k, n/k)) + k \cdot \text{RCost}(\Pi_{\text{Ext-Ord}}(n/k, n/k))$ and $\text{CCost}(\Pi_{\text{Ext-Bin}}(k, k, n/k)) + k \cdot \text{CCost}(\Pi_{\text{Ext-Ord}}(n/k, n/k))$.
- Step (3a) has $\text{RCost}(\Pi_{\text{SSM}''}(k))$ and $\text{CCost}(\Pi_{\text{SSM}''}(k))$.
- Step (3b) has $\text{RCost}(\Pi_{\text{SSM}'}(n/k))$ and $k \cdot \text{CCost}(\Pi_{\text{SSM}'}(n/k))$.
- Step (4) has $\text{RCost}(\Pi_{\text{Extract}}(2(k+n), k+n)) + \text{RCost}(\Pi_{\text{Reveal}})$ and $\text{CCost}(\Pi_{\text{Extract}}(2(k+n), k+n)) + (k+n) \cdot \text{CCost}(\Pi_{\text{Reveal}})$.

Using the costs of subprotocols Π_{Extract} , $\Pi_{\text{Ext-Ord}}$, and $\Pi_{\text{Ext-Bin}}$, and assuming constant-round and linear secure protocols for Π_{Comp} , Π_{Reveal} , and Π_{Shuffle} , we add up the contributions from each step to obtain:

$$\begin{aligned}
\text{RCost}(\Pi_{\text{SAM}}(k, n)) &= \text{RCost}(\Pi_{\text{SSM}''}(k)) + \text{RCost}(\Pi_{\text{SSM}'}(n/k)) + \\
&\quad \text{RCost}(\Pi_{\text{Comp}}) + \text{RCost}(\Pi_{\text{Reveal}}) + \\
&\quad \text{RCost}(\Pi_{\text{Ext-Ord}}(k, k)) + \text{RCost}(\Pi_{\text{Ext-Ord}}(n, k)) \\
&\quad \text{RCost}(\Pi_{\text{Ext-Bin}}(k, k, n/k)) + \text{RCost}(\Pi_{\text{Ext-Ord}}(n/k, n/k)) \\
&\quad \text{RCost}(\Pi_{\text{Extract}}(2(k+n), k+n)) \\
&= O(1) + \text{RCost}(\Pi_{\text{SSM}''}(k)) + \text{RCost}(\Pi_{\text{SSM}'}(n/k)) \\
\text{CCost}(\Pi_{\text{SAM}}(k, n)) &= 2 \cdot \text{CCost}(\Pi_{\text{SSM}''}(k)) + k \cdot \text{CCost}(\Pi_{\text{SSM}'}(n/k)) + \\
&\quad \Theta(k) \cdot \text{CCost}(\Pi_{\text{Comp}}) + (k+n) \cdot \text{CCost}(\Pi_{\text{Reveal}}) + \\
&\quad \text{CCost}(\Pi_{\text{Ext-Ord}}(k, k)) + \text{CCost}(\Pi_{\text{Ext-Ord}}(n, k)) \\
&\quad \text{CCost}(\Pi_{\text{Ext-Bin}}(k, k, n/k)) + k \cdot \text{CCost}(\Pi_{\text{Ext-Ord}}(n/k, n/k)) \\
&\quad \text{CCost}(\Pi_{\text{Extract}}(2(k+n), k+n)) \\
&= O(k+n) + 2 \cdot \text{CCost}(\Pi_{\text{SSM}''}(k)) + k \cdot \text{CCost}(\Pi_{\text{SSM}'}(n/k))
\end{aligned}$$

Using $\Pi_{\text{SAM-FNO}}$ for $\Pi_{\text{SSM}'}(n/k)$ and $\Pi_{\text{SSM-log log}}$ for $\Pi_{\text{SSM}''}(k)$, the cost is:

$$\begin{aligned}
\text{RCost}(\Pi_{\text{SAM}}(k, n)) &= O(\log \log k) + O(n/k) = O(n/k + \log \log k) = O(\log \log n) \\
\text{CCost}(\Pi_{\text{SAM}}(k, n)) &= O(k+n) + O(k \log \log k) + O(n) = O(n + k \log \log k) = O(n)
\end{aligned}$$

where the final equality for costs comes as a result of setting $k = n / \log \log n$.

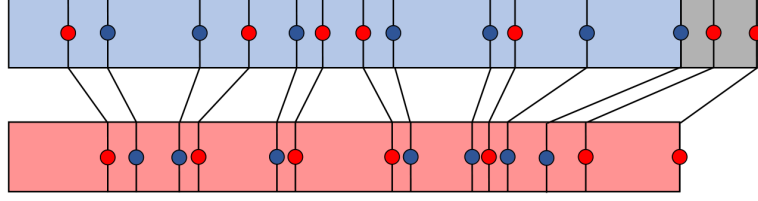
6 Description of Base Protocols: $\Pi_{\text{SSM-log log}}$ and $\Pi_{\text{SAM-n}^{1/3}}$

In this section, we present our $\Pi_{\text{SSM-log log}}$ and $\Pi_{\text{SAM-n}^\alpha}$ base protocols (for $\alpha = 1/3$; the general case for $\alpha < 1$ can be found in the full version).

6.1 Secure Symmetric Merge with $O(n \log \log n)$ Communication

As an ingredient for $\Pi_{\text{SAM}}(k, n)$, we need a secure symmetric merge with $O(n \log \log n)$ communication and $O(\log \log n)$ rounds, which we call $\Pi_{\text{SSM-log log}}$. We give this protocol in Figure 7. For the Partition phase, it uses a secure asymmetric merge protocol (namely, the $\Pi_{\text{SAM-n}^{1/3}}$ in the next section) to merge the $k = n^{1/3}$ medians of each list into the other list. The $2k$ medians of both lists thus partition each list into $2k$ blocks of size at most n/k . For the Align Blocks phase, we group each block from L_1 with the corresponding block from L_2 , as shown in Figure 6, and pad each block with dummies so that it has length exactly n/k . For the Merge Blocks phase, each pair of matching blocks are merged together; and then the Combine Blocks phase simply concatenates and removes dummy elements that were introduced to hide the alignment of blocks.

The protocol is recursive, and, if implemented naïvely, the problem sizes would double at each step of the recursion, for $O(n \log n)$ computation (instead of the desired $O(n \log \log n)$), because there are $\log \log n$ steps total. In Fig. 7, we show how to discard sub-problems along the way to avoid this blow-up.



■ **Figure 6** Each list is partitioned into $2k$ blocks by the k medians from both lists. The two grey blocks appended to the end of the top list are made up entirely of dummy elements, and show how to handle medians from one list that merge to a location outside the other list. Each block is padded with the appropriate number of dummy elements to ensure it has n/k elements. Each pair of aligned blocks (visualized via the connecting line segments) are then merged in $2k$ sub-problems (run in parallel) for $\Pi_{\text{SSM}}(n/k)$, with the final result obtained by concatenating the results of these $2k$ merges and removing dummy elements.

6.2 Analysis of the $\Pi_{\text{SSM}-\log \log(n)}$ Protocol of §6.1

Security. To simulate either player's view during an execution of the protocol, a simulator can call the simulator of the sub-protocols on every step except for Step 11, since that is the only step where values are revealed to the parties. For Step 11, the parties only see a random permutation of $\{1, 2, \dots, 2n\}$, so their views can be simulated by randomly sampling such a permutation.

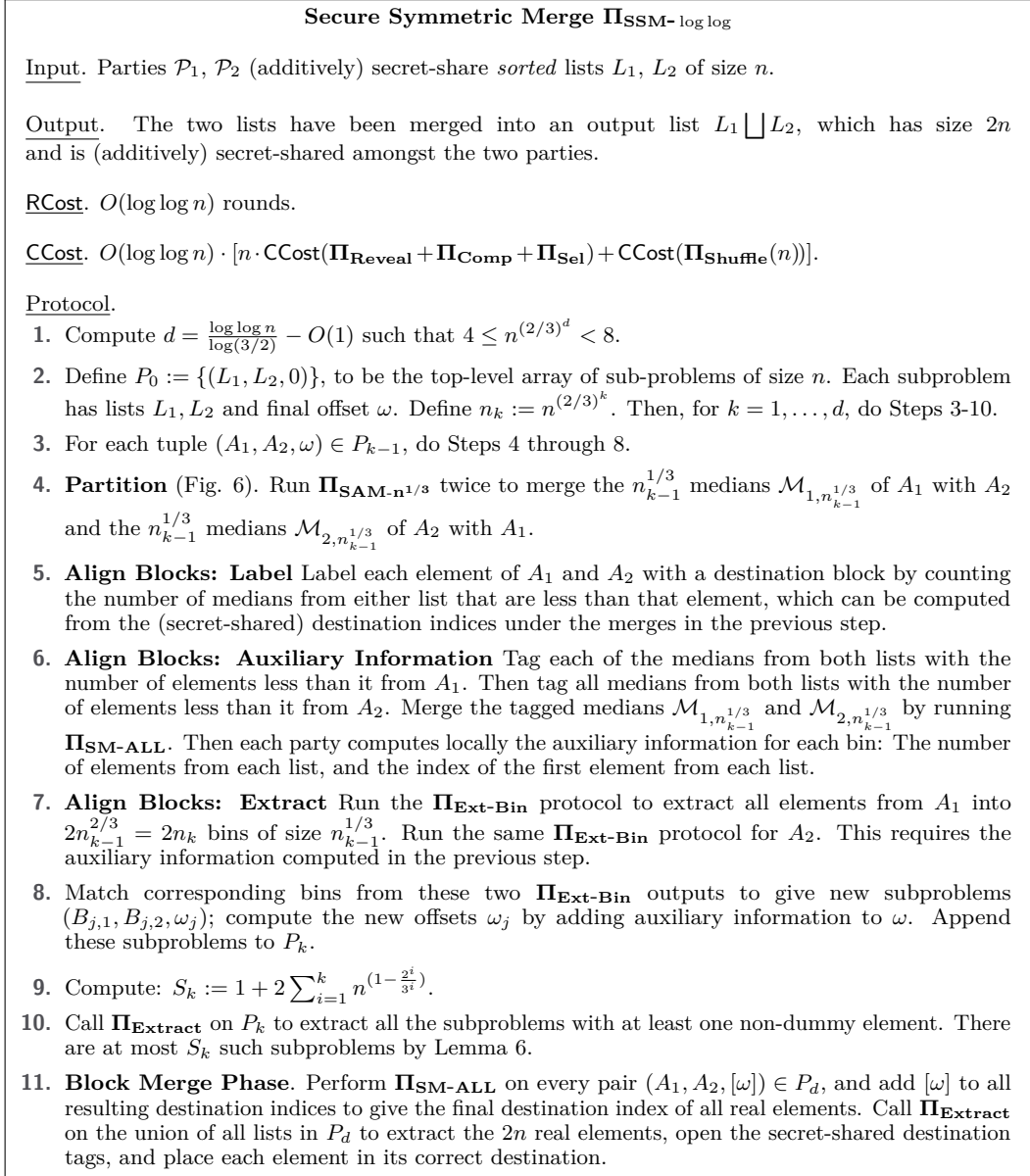
Correctness. The overall goal will be to show that the subproblems extracted in Step 10 contain a partition of the $2n$ elements into well-ordered blocks, so that the real elements of each block are either all greater than or all less than the real elements from another block. We must show this well-orderedness property and that all elements are included, so that it is truly a partition. We proceed step-by-step.

In Step 4, the $\Pi_{\text{SAM-}n^{1/3}}$ protocols are performed “in-place”, so that each element gets a (secret-shared) tag of their destination under this merge. These tags can be transformed into shares of the number of medians from the opposite list less than a given element, or shares of the number of elements from the opposite list less than a given median.

In Step 5, the shares of the destination block are computed by adding the shares of the number of medians of the opposite list less than a given element to the floor of the element's index divided by $n_{k-1}^{2/3}$, which counts the number of medians from the same list less than a given element, and is a public value.

In Step 6, again, the number of elements less than a given median from the opposite list is a secret shared value generated in Step 4, and the number of elements less than a given median in the same list is a public value. This time, when we run $\Pi_{\text{SM-ALL}}$, we do not run it “in-place”, but return the output list of tagged medians. Note that we tag all medians first with an A_1 -tag (counting elements from A_1 less than that median), then with an A_2 -tag, so that after merging, the A_1 medians and A_2 medians are indistinguishable, and we can compute the correct auxiliary information without leaking information. The index of the first element of A_1 (resp. A_2) in the j -th bin is the A_1 -tag (resp. A_2 -tag), since the first element in the j -th bin is the first element not less than the j -th median. The number of elements from A_1 (resp. A_2) in the j -th bin is equal to the $(j+1)$ -th A_1 -tag (resp. A_2 -tag) minus the j -th tag, since this is the difference between the start index of two adjacent bins.

For Step 7, we have just shown the correctness of the auxiliary information generated for $\Pi_{\text{Ext-Bin}}$. It remains to show that the assumed bound on the size of each bin holds. The union of the values from $\mathcal{M}_{1, n_{k-1}^{1/3}}$ and $\mathcal{M}_{2, n_{k-1}^{1/3}}$ will partition each list A_1 and A_2 into $2n_{k-1}^{1/3}$



■ **Figure 7** $O(n \log \log n)$ Secure Symmetric Merge Protocol

blocks (since the list of both medians includes the maximum of both lists, no non-median element can lie to the right of all medians, see Fig. 6). Each block will have at most n_k real elements, since each block is either composed of the interval between two adjacent medians of the same list, which has exactly n_k elements, or a sub-interval of such an interval.

In Step 8, we obtain the new offset ω_j by adding ω (the offset for the current subproblem) to the j -th A_1 -tag and the j -th A_2 tag. These two tags count the number of elements from A_1 and A_2 less than the j -th median, but we must verify that all of these elements are real. But all dummy elements are sorted to the right of all real-valued medians, so either the entire subproblem will be dummy, and the subproblem will be discarded in Step 10, or all elements counted by the A_1 -tag and the A_2 -tag are non-dummy, and our computation of ω_j is correct.

The value S_k and the bound on non-dummy subproblems in Step 10 follow from Lemma 6, below. The correctness of the final indices in Step 11 follows from the correctness of the offsets ω_j computed along the way and the correctness of $\Pi_{\text{SM-ALL}}$, and since we have verified that P_d contains a partition of the $2n$ real elements, the bound on real elements for the final Π_{Extract} call is also correct.

► **Lemma 6.** *At a depth of d , there are at most $S_d := 1 + 2 \sum_{i=1}^d n^{1-\frac{2^i}{3^i}}$ sub-problems without all elements dummy. For $d = \frac{\log \log n}{\log(3/2)} - O(1)$, $S_d \leq 4n^{1-(2/3)^d}$.*

Proof. Let N_k be the number of subproblems at depth k with at least one non-dummy element, that is, the cardinality of the set $|P_k|$ after the extraction in Step 10. Then we have $N_0 = 1$ and we will prove that $N_{k+1} \leq N_k + 2n^{1-(2/3)^k}$, which is sufficient to prove the lemma. To prove this recurrence relation, we will divide the subproblems generated before the extraction in Step 10 into three categories: A group of subproblems that have an average density of at least $1/2$ real elements, one special subproblem, and a group of subproblems made up entirely of dummy elements. Since subproblems at level k have $2n^{(2/3)^k}$ elements, and there are $2n$ real elements total, there can be at most $2n/n^{(2/3)^k} = 2n^{1-(2/3)^k}$ subproblems in the first category. There will also be, of course, N_k special subproblems, which will prove the recurrence. It remains to describe this categorization.

Let (A_1, A_2, ω) be a subproblem at level $k-1$, and let m_1 and m_2 be the number of median elements that are real in A_1 and A_2 respectively. Then the first $m_1 + m_2$ blocks (as determined in Step 5) will include at least the first $m_1 n^{(2/3)^k}$ real elements of A_1 and the first $m_2 n^{(2/3)^k}$ real elements of A_2 . Thus these $m_1 + m_2$ subproblems will have $2(m_1 + m_2)n^{(2/3)^k}$ elements and at least $(m_1 + m_2)n^{(2/3)^k}$ real elements, which proves that their average density is at least $1/2$. Because the next median of each list is a dummy element, all remaining real elements of both A_1 and A_2 will be mapped into the $m_1 + m_2 + 1$ -th block, which is the special subproblem.

For the final bound on S_d , note that there exists a unique value of d , with $d < \frac{\log \log n}{\log(3/2)}$, such that $4 \leq n^{(2/3)^d} < 8$. We thus have for $k < d$ the desired bound:

$$\frac{n^{1-(2/3)^k}}{n^{1-(2/3)^{k+1}}} = n^{-\frac{2^k}{3^{k+1}}} \leq \frac{1}{2} \quad \Rightarrow \quad S_d \leq 2n^{(2/3)^d} \sum_{i=0}^d \left(\frac{1}{2}\right)^{d-i} < 4n^{(2/3)^d}. \quad \blacktriangleleft$$

Cost. At a depth k , Steps 4 through 8 of Figure 7 require $O(n^{(2/3)^k})$ communication and computation and $O(1)$ rounds for each element of P_k , since the computations in each of Steps 4 through 8 are linear. By Lemma 6, at a depth of k , there are at most $S_k = 2n^{1-(2/3)^k} + O(n^{1-(2/3)^{k+1}}) = O(n^{1-(2/3)^k})$ subproblems, so the total work for Steps 4 through 8 is $O(n)$ communication and computation and $O(1)$ rounds. Steps 4 through 8 have the effect of doubling the total number of elements, so the total size of the lists in P_k before the extraction in Step 10 is $2S_k n^{1-(2/3)^k} = O(n)$, and so Step 10 requires $O(n)$ work each time it is called. Since Steps 3-10 are called $\log \log n - O(1)$ times, the total communication is $O(n \log \log n)$ and the round complexity is $O(\log \log n)$, as desired.

6.3 Secure Asymmetric Merge on $(n^{1/3}, n)$

For the special case where $|L_1| = O(n^{1/3})$ and $|L_2| = O(n)$, succinctly describe

an asymmetric merge protocol with communication complexity $O(n)$ and round complexity $O(1)$. By calling $\Pi_{\text{SM-ALL}}$ on L_1 and the $n^{2/3}$ medians of L_2 , we can identify every block of L_2 which contains an element of L_1 after merging the two lists. Because there are $O(n^{1/3})$

elements of L_1 , there are $O(n^{1/3})$ such blocks of L_2 . After extracting these blocks, we run $\Pi_{\text{SM-ALL}}$ again on L_1 and the $O(n^{2/3})$ elements of the extracted blocks of L_2 . After some careful accounting of the index shifts during these steps, we get the destination indices for all the elements, which gives the desired merge protocol. Due to space considerations, we refer the reader to Fig. 9 of the full version for details.

References

- 1 M. Ajtai, J. Komlós, and E. Szemerédi. An $o(n \log n)$ sorting network. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, pages 1–9, New York, NY, USA, 1983. Association for Computing Machinery. doi:10.1145/800061.808726.
- 2 Arvind Arasu and Raghav Kaushik. Oblivious query processing. In *Proc. 17th International Conference on Database Theory (ICDT)*, Athens, Greece, March 24–28, 2014, pages 26–37. OpenProceedings.org, 2014. doi:10.5441/002/icdt.2014.07.
- 3 K. E. Batcher. Sorting networks and their applications. In *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*, AFIPS '68 (Spring), pages 307–314, New York, NY, USA, 1968. Association for Computing Machinery. doi:10.1145/1468075.1468121.
- 4 Allan Borodin and John E. Hopcroft. Routing, merging, and sorting on parallel models of computation. *J. Comput. Syst. Sci.*, 30(1):130–145, 1985. doi:10.1016/0022-0000(85)90008-X.
- 5 Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptol.*, 13(1):143–202, 2000. doi:10.1007/s001459910006.
- 6 Suvaradip Chakraborty, Stanislav Peceny, Srinivasan Raghuraman, and Peter Rindal. Logstar: Efficient linear* time secure merge. *IACR Cryptol. ePrint Arch.*, page 159, 2024. URL: <https://eprint.iacr.org/2024/159>.
- 7 T.-H. Hubert Chan, Jonathan Katz, Kartik Nayak, Antigoni Polychroniadou, and Elaine Shi. More is less: Perfectly secure oblivious algorithms in the multi-server setting. In *Advances in Cryptology – ASIACRYPT 2018*, volume 11274 of *Lecture Notes in Computer Science*, pages 158–188. Springer, 2018. doi:10.1007/978-3-030-03332-3_7.
- 8 Koji Chida, Koki Hamada, Dai Ikarashi, Ryo Kikuchi, Naoto Kiribuchi, and Benny Pinkas. An efficient secure three-party sorting protocol with an honest majority. *IACR Cryptol. ePrint Arch.*, page 695, 2019. URL: <https://eprint.iacr.org/2019/695>.
- 9 Saba Eskandarian and Matei Zaharia. Oblidb: Oblivious query processing for secure databases. *Proc. VLDB Endow.*, 13(2):169–183, October 2019. doi:10.14778/3364324.3364331.
- 10 Brett Hemenway Falk, Rohit Nema, and Rafail Ostrovsky. A linear-time 2-party secure merge protocol. In *Cyber Security, Cryptology, and Machine Learning - 6th International Symposium, CSCML 2022, Be'er Sheva, Israel, June 30 - July 1, 2022, Proceedings*, volume 13301 of *Lecture Notes in Computer Science*, pages 408–427. Springer, 2022. doi:10.1007/978-3-031-07689-3_30.
- 11 Brett Hemenway Falk and Rafail Ostrovsky. Secure merge with $o(n \log \log n)$ secure operations. In *2nd Conference on Information-Theoretic Cryptography (ITC 2021)*, volume 199 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:29, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITC.2021.7.
- 12 O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 218–229, New York, NY, USA, 1987. Association for Computing Machinery. doi:10.1145/28395.28420.
- 13 Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious RAMs. *J. ACM*, 43(3):431–473, 1996. doi:10.1145/233551.233553.
- 14 Koki Hamada, Dai Ikarashi, Koji Chida, and Katsumi Takahashi. Oblivious radix sort: An efficient sorting algorithm for practical secure multi-party computation. *IACR Cryptol. ePrint Arch.*, page 121, 2014. URL: <http://eprint.iacr.org/2014/121>.
- 15 Koki Hamada, Ryo Kikuchi, Dai Ikarashi, Koji Chida, and Katsumi Takahashi. Practically efficient multi-party sorting protocols from comparison sort algorithms. In *International*

- Conference on Information Security and Cryptology*, pages 202–216. Springer, 2013. doi:10.1007/978-3-642-37682-5_15.
- 16 Zhu Hong and Robert Sedgewick. Notes on merging networks (preliminary version). In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, pages 296–302, New York, NY, USA, 1982. Association for Computing Machinery. doi:10.1145/800070.802204.
 - 17 Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*. The Internet Society, 2012. URL: <https://www.ndss-symposium.org/ndss2012/private-set-intersection-are-garbled-circuits-better-custom-protocols>.
 - 18 Yuval Ishai, Eyal Kushilevitz, Steve Lu, and Rafail Ostrovsky. Private large-scale databases with distributed searchable symmetric encryption. In *Topics in Cryptology - CT-RSA 2016*, pages 90–107, Cham, 2016. Springer International Publishing. doi:10.1007/978-3-319-29485-8_6.
 - 19 Kasper Green Larsen and Jesper Buus Nielsen. Yes, there is an oblivious ram lower bound! In *Advances in Cryptology – CRYPTO 2018*, pages 523–542, Cham, 2018. Springer International Publishing. doi:10.1007/978-3-319-96881-0_18.
 - 20 Steve Lu and Rafail Ostrovsky. Distributed oblivious ram for secure two-party computation. In *Theory of Cryptography*, pages 377–396, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. doi:10.1007/978-3-642-36594-2_22.
 - 21 Kurt Mehlhorn and Peter Sanders. *Algorithms and Data Structures: The Basic Toolbox*. Springer Publishing Company, Incorporated, 1 edition, 2008. doi:10.1007/978-3-540-77978-0.
 - 22 Takashi Nishide and Kazuo Ohta. Constant-round multiparty computation for interval test, equality test, and comparison. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 90-A(5):960–968, 2007. doi:10.1093/ietfec/e90-a.5.960.
 - 23 Benny Pinkas and Tzachy Reinman. Oblivious ram revisited. In *Advances in Cryptology – CRYPTO 2010*, pages 502–519, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. doi:10.1007/978-3-642-14623-7_27.
 - 24 Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on ot extension. *ACM Trans. Priv. Secur.*, 21(2), January 2018. doi:10.1145/3154794.
 - 25 Leslie G. Valiant. Parallelism in comparison problems. *SIAM Journal on Computing*, 4(3):348–355, 1975. doi:10.1137/0204030.
 - 26 Andrew Chi-Chih Yao and Foong Frances Yao. Lower bounds on merging networks. *J. ACM*, 23(3):566–571, 1976. doi:10.1145/321958.321976.
 - 27 Wenting Zheng, Ankur Dave, Jethro G. Beekman, Raluca Ada Popa, Joseph E. Gonzalez, and Ion Stoica. Opaque: An oblivious and encrypted distributed analytics platform. In *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*, pages 283–298. USENIX Association, 2017. URL: <https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/zheng>.

On the Definition of Malicious Private Information Retrieval

Bar Alon 

Department of Computer Science, Georgetown University, Washington DC, USA

Amos Beimel 

Department of Computer Science, Ben Gurion University, Be'er-Sheva, Israel

Abstract

A multi-server private information retrieval (PIR) protocol allows a client to obtain an entry of its choice from a database, held by one or more servers, while hiding the identity of the entry from small enough coalitions of servers. In this paper, we study PIR protocols in which some of the servers are malicious and may not send messages according to the pre-described protocol. In previous papers, such protocols were defined by requiring that they are correct, private, and robust to malicious servers, i.e., by listing 3 properties that they should satisfy. However, 40 years of experience in studying secure multiparty protocols taught us that defining the security of protocols by a list of required properties is problematic.

In this paper, we rectify this situation and define the security of PIR protocols with malicious servers using the real vs. ideal paradigm. We study the relationship between the property-based definition of PIR protocols and the real vs. ideal definition, showing the following results:

- We prove that if we require full security from PIR protocols, e.g., the client outputs the correct value of the database entry with high probability even if a minority of the servers are malicious, then the two definitions are equivalent. This implies that constructions of such protocols that were proven secure using the property-based definition are actually secure under the “correct” definition of security.
- We show that if we require security-with-abort from PIR protocols (called PIR protocols with error-detection in previous papers), i.e., protocols in which the user either outputs the correct value or an abort symbol, then there are protocols that are secure under the property-based definition; however, they do not satisfy the real vs. ideal definition, that is, they can be attacked allowing selective abort. This shows that the property-based definition of PIR protocols with security-with-abort is problematic.
- We consider the compiler of Eriguchi et al. (TCC 22) that starts with a PIR protocol that is secure against semi-honest servers and constructs a PIR protocol with security-with-abort; this compiler implies the best-known PIR protocols with security-with-abort. We show that applying this compiler does *not* result in PIR protocols that are secure according to the real vs. ideal definition. However, we prove that a simple modification of this compiler results in PIR protocols that are secure according to the real vs. ideal definition.

2012 ACM Subject Classification Security and privacy → Cryptography

Keywords and phrases Private information retrieval, secure multiparty computation

Digital Object Identifier 10.4230/LIPIcs.ITC.2025.8

Related Version *Full Version:* <https://eprint.iacr.org/2025/683>

Funding *Amos Beimel:* Partially supported by ISF grant 391/21. Co-funded by the European Union (ERC, NFITSC, 101097959). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.



© Bar Alon and Amos Beimel;
licensed under Creative Commons License CC-BY 4.0
6th Conference on Information-Theoretic Cryptography (ITC 2025).
Editor: Niv Gilboa; Article No. 8; pp. 8:1–8:23



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

1 Introduction

A private information retrieval (PIR) protocol [16] allows a client to obtain an entry of its choice from a database held by one or more servers, such that nothing is revealed to any small enough coalition of servers about the item being revealed. For example, an investor might want to know the value of a specific stock without revealing which stock they are interested in. This is modeled by considering an N -bit string D held by each server, and the client holding a retrieval index i wishing to learn the i^{th} entry of D , i.e., D_i , without revealing i . The trivial solution is to let a single server send the entire database to the client: Chor et al. [16] showed that this is optimal in the information-theoretic setting when there is a single server; however, better protocols are known if the database is held by $\ell \geq 2$ servers. PIR protocols were used to construct secure multiparty protocols [30, 7], locally decodable codes [34], and inspired constructions of conditional disclosure of secrets [39, 40]. Most constructions of PIR protocols guarantee that any single server will not learn any information on the client's retrieval index [16, 2, 30, 31, 6, 8, 35, 46, 48, 22, 33, 14, 20, 27, 1], e.g., there is a 3-server PIR protocol secure against a single server with communication complexity $2^{\tilde{O}(\sqrt[3]{\log N})}$ [27]. When there are many servers, it is natural to require that the protocol guarantees the privacy of the client against colluding subsets of the servers. Constructions of such PIR protocols are given in [30, 6, 8, 46, 5]. It is also natural to consider *malicious* servers that may deviate from the protocol. In the stock market example above, one may consider the case where some of the servers try to give the wrong information on the stock. Such PIR protocols were first discussed in [9] and further studied in [47, 46, 28, 18, 49, 43, 44, 37, 4, 23, 24, 25].

The security of PIR protocols in the literature is defined by listing the desired properties from the protocol. However, PIR is a special case of *secure multiparty computation (MPC)*, where it is well-known that separating the security properties is problematic. Indeed, there are MPC protocols that satisfy the natural security properties (such as privacy and correctness) yet should clearly be considered insecure [42, 13, 29]. Instead, the security of MPC protocols is defined using the *real vs. ideal world* paradigm. Intuitively, we consider an ideal process for computing a function via a trusted party that the adversary cannot corrupt. It is then required that any attack performed in the real world can be simulated in the ideal world, where the adversary is more limited. Different ideal processes capture different security properties, e.g., full security and security-with-abort.

Defining malicious security of PIR protocol via properties is problematic and it is unclear if there are other security properties that can be attacked in such protocols. In this work, we rectify this situation and define the security of PIR protocols using the real vs. ideal world paradigm. Specifically, we are interested in the notions of *full security* (also known as guaranteed output delivery) and *security-with-abort*. The former captures the requirement that the client always obtain the “correct” output, and the latter allows the client to abort but never output an “incorrect” value. We ask how such definitions relate to the property-based definitions used so far, and if there are efficient PIR protocols satisfying the real vs. ideal security definition. We consider the perfect setting, the statistical setting, and the computational setting.

1.1 Our Contributions

Our conceptual contribution is defining the ideal processes of the PIR functionality for full security and security-with-abort against malicious adversaries (see Section 3 for the formal definitions). We then compare the simulation-based security definition to the property-based security definition.

1.1.1 The Ideal World Processes

In this section, we describe the ideal world process for each of the desired security notions. A PIR protocol is then considered secure according to an ideal world if any adversary corrupting a subset of the servers can be simulated in the ideal world. We stress that definitions for PIR protocols do not have any privacy requirements from the client; in particular, the client may learn the whole database. Thus, we only consider adversaries that do not corrupt the client.

1.1.1.1 Full Security of PIR Protocols

We first describe the definition of fully secure PIR protocols. Roughly, the definition captures the requirement that the client outputs the correct value D_i , regardless of the messages sent by the malicious servers. Observe that this only makes sense when the majority of the servers are honest.¹ The ideal world for full security proceeds as follows. The client sends i to the trusted party and each honest server sends D to the trusted party. The corrupted servers may change their databases (based on the instruction of the ideal-world adversary) and send them to the trusted party. Let D' denote the database that was sent by more than $1/2$ of the servers (note that $D' = D$ if there is an honest majority). The trusted party then sends D'_i to the client. The ideal functionality is only defined when there is an honest majority.

1.1.1.2 Secure-With-Abort PIR Protocols

We next describe the definition of secure-with-abort PIR protocols. Here, we allow the client to output $\perp$ (indicating abort). The security definition captures the requirement that the client never outputs $1 - D_i$. Note that, unlike the ideal world for full security, here the definition allows a majority of corrupted servers.

The ideal-world computation proceeds as follows. The client sends i to the trusted party and each honest server sends D to the trusted party. The corrupted servers may change their databases (based on the instruction of the ideal-world adversary) and send them to the trusted party. If the trusted party receives the same database D from all servers, then it sends D_i to the client; otherwise, it sends $\perp$ to the client.

1.1.2 Comparing Simulation-Based Security to Property-Based Security

We compare the simulation-based security definitions to the property-based security definitions. In the following, we fix the total number of servers to be ℓ , and let t bound the number of corrupted servers. We refer the reader to Section 4 for the formal statements and proofs of the theorems.

1.1.2.1 Full Security

We first compare full security with the property-based definition. In addition to correctness and t -privacy, here we also consider the notion of t -Byzantine robust² PIR protocols defined by Beimel and Stahl [9]. Roughly, it requires that the client outputs D_i even if t malicious servers arbitrarily deviate from the protocol. Note that similarly to full security, this definition only makes sense when the majority of the servers are honest. As would be expected, full security implies all 3 security properties. We show that the converse also holds.

¹ If the number of servers is ℓ and t of them might be corrupted, where $2t \geq \ell$, then the client cannot distinguish an execution with the first t servers being corrupted and behaving honestly on input 0^N and the $\ell - t$ remaining honest servers holding 1^N , from an execution with the last $\ell - t \leq t$ servers being corrupted and behaving honestly on input 1^N and the first t servers being honest and holding 0^N .

² Also known as t -error-correcting [24].

► **Theorem 1** (Informal, equivalence of full security and the property-based definition). *Any ℓ -server PIR protocol is t -fully secure if and only if it is correct, t -private, and t -Byzantine robust.*

Thus, to prove full security it suffices to show that each property holds individually. In particular, this shows that previous Byzantine robust constructions, e.g., [9, 47, 46, 43, 44, 37, 4, 23, 24], are also fully secure.

1.1.2.2 Secure-With-Abort

Finally, we consider security-with-abort. Here, instead of t -Byzantine robustness, we consider the notion of t -error-detecting PIR protocols defined by Eriguchi et al. [23, 24]. In a t -error-detecting PIR protocol, the client can abort (i.e., by outputting $\perp$) but security requires that no set of t malicious servers can force the client to output $1 - D_i$. Clearly, error-detecting is weaker than Byzantine robust; however, it is possible to construct t -error-detecting PIR protocols when $t \geq \ell/2$ [23, 24].

Similarly to the previous definitions, the simulation-based definition implies the property-based definition. However, we show that here the converse is *not* true.

► **Theorem 2** (Informal, security-with-abort and the property-based definition). *Any ℓ -server PIR protocol that is t -secure-with-abort is correct, t -private, and t -error-detecting. However, the converse does not hold.*

To see why the properties do not imply simulation-based security, consider the following protocol, in which a single malicious server can cause a “selective abort”, i.e., the user will abort if and only if $i = 1$. In the protocol, each server sends D to the client and additionally, the first server sends an additional bit $b \in \{0, 1\}$. An honest server sends $b = 0$. If the client received different databases, or if it has input $i = 1$ and it received $b = 1$, then it outputs $\perp$. Otherwise, the client only receives the database D , and it outputs D_i . Clearly, Π is correct, 1-private, and 1-error-detecting. However, it is *not* 1-secure-with-abort. This is because the first server can force a different probability for the client to abort on different indices i . Specifically, if the server sends $b = 1$, then the client always aborts on index $i = 1$, and it never aborts on index $i \neq 1$. Note that this holds even though the server does not know anything about i . See Example 18 for the formal argument. In Example 19, we show that even if we require the same probability for abort on all indexes i , then it is still insufficient to claim security-with-abort.

Colombo et al. [17] noticed that standard definitions for the security of PIR might allow for selective abort (however, they did not provide any concrete example). This motivated them to define *authenticated PIR* to capture security against such attacks. However, their definition is property-based, and it is unclear whether their definition is equivalent to our simulation-based security, or whether it captures security against attacks not mentioned in their paper.

1.1.2.3 A Compiler From a Private PIR Protocol to a Secure-With-Abort Protocol

Eriguchi et al. [24] showed a compiler that takes any ℓ -server correct and t -private PIR protocol, and transforms it into a correct, t -private, and t -error-detecting ℓ -server PIR protocol. In light of Theorem 2, it is natural to ask whether their compiler satisfies the stronger requirement of t -security-with-abort. We show that this is *not* the case; however, we prove that a simple modification to their construction results in a secure-with-abort protocol.

■ **Table 1** A summary of the best PIR protocols tolerating t corrupted servers for various parameters. The results in the second row are obtained by applying our compiler on the protocols from the first row. These results are also obtained by [24] for the weaker property-based security. The last row follows from [24] and Theorem 1.

	2^t server PIR from [5] + [20]	3^t server PIR from [5] + [22]	The ℓ server PIR, $\ell = O(1)$ [46]
t -private	$2^{\tilde{O}(t\sqrt{\log N})}$	$\max \left\{ t \cdot 2^{\tilde{O}(\sqrt{\log N})}, 2^{O(t)} \right\}$	$N^{1/\lfloor (2\ell-1)/t \rfloor}$
t -security with-abort	$2^{\max \left\{ \tilde{O}(t\sqrt{\log N}), t \log t \right\}}$	$3^{2t} \cdot \max \left\{ t 2^{\tilde{O}(\sqrt{\log N})}, t 3^t \right\}$	$N^{\frac{1}{\lfloor (2\ell-1)/t \rfloor} + o(1)}$
t -full security	$2^{t^2 + \max \left\{ \tilde{O}(t\sqrt{\log N}), t \log t \right\}}$	$3^{t^2 + 2t} \cdot \max \left\{ t 2^{\tilde{O}(\sqrt{\log N})}, 2^{O(t)} \right\}$	$N^{\frac{1}{\lfloor (2\ell-1)/t \rfloor - 2} + o(1)}$

► **Theorem 3** (Informal, compiler to secure-with-abort). *There exists a compiler that transforms any ℓ -server correct and t -private PIR protocol Π into an ℓ -server t -secure-with-abort PIR protocol. The total communication complexity (i.e., the total number of bits sent) of the new protocol is $\ell^2 \cdot \omega(\log N) \cdot (c + \ell \log \ell)$, where c is the communication complexity of Π .*

This result implies that for a constant number of servers, the communication complexity of t -secure-with-abort PIR protocols is equivalent to PIR protocols secure against t -semi-honest servers (up to a factor of $\omega(\log N)$). We refer to Section 5 for the formal statement and proof. In Table 1 we summarize the PIR protocols obtained from our results.

1.2 Our Techniques

To illustrate our techniques, we next show that t -semi-honest security is equivalent to correctness and t -privacy. Since simulation-based security is clearly stronger than property-based security, we only show that correctness and t -privacy imply t -semi-honest security. We do it for statistical security (other cases are handled similarly). This is a special case of a known result that the property-based definition is equivalent to the simulation-based definition for deterministic functionalities with semi-honest security against unbounded adversaries.

Fix a real-world adversary $\mathcal{B}$ corrupting a set $\mathcal{I}$ of at most t servers. We define its simulator Sim running the client on index $i = 1$, and outputting the queries that correspond to the corrupted servers. We now show that the statistical distance between the real and ideal worlds is negligible. For a retrieval index $i \in [N]$ and a database $D \in \{0, 1\}^N$ let $y_{D,i}$ denote the output of the client in the real world, and let $\mathbf{q}_{\mathcal{I}}^i$ denote the queries the corrupted server receives. By construction of the simulator and the fact that the client always outputs D_i in the ideal world, we need to show that $(\mathbf{q}_{\mathcal{I}}^i, y_{D,i})$ and $(\mathbf{q}_{\mathcal{I}}^1, D_i)$ are statistically close. First, by the t -privacy of the protocol, $(\mathbf{q}_{\mathcal{I}}^1, D_i)$ and $(\mathbf{q}_{\mathcal{I}}^i, D_i)$ are statistically close. Second, observe that correctness implies that $(\mathbf{q}_{\mathcal{I}}^i, D_i)$ and $(\mathbf{q}_{\mathcal{I}}^i, y_{D,i})$ are statistically close. Thus, $(\mathbf{q}_{\mathcal{I}}^i, y_{D,i})$ and $(\mathbf{q}_{\mathcal{I}}^1, D_i)$ are statistically close.

1.3 Related Works

1.3.1 1-Private PIR

We next state the best known PIR protocols. Efremenko [22] constructed a 3-server 1-private PIR protocol with query length $2^{\tilde{O}(\sqrt{\log N})}$ and 2^r -server 1-private PIR protocols with query length $2^{\tilde{O}(\sqrt[r]{\log N})}$; the answer length in these protocols is $O(1)$. Dvir and Gopi [20] constructed a PIR protocol with query and answer length $2^{\tilde{O}(\sqrt{\log N})}$. Very recently, Ghasemi, Kopparty, and Sudan [27] constructed a 3-server PIR protocol with query and answer length $2^{\tilde{O}(\sqrt[3]{\log N})}$. These constructions were simplified by Alon, Beimel, and Lasri [1]. The communication complexity of ℓ -server PIR protocols for $\ell \geq 6$ was improved by Itoh and Suzuki [32, 33], Chee, Feng, Ling, Wang, and Zhang [14], Dvir and Gopi [20], and Ghasemi et al. [27]. For example, there is a 6-server PIR protocol with communication complexity $2^{\tilde{O}(\sqrt[6]{\log N})}$ [32, 3, 27, 1]. The best known lower bound on the total communication complexity of 2-server PIR protocols is $5 \log n$, proved by Wehner and de Wolf [45] (improving on [41, 34, 36]).

1.3.2 t -Private PIR

t -private PIR protocols were constructed in [16, 2, 30, 6, 8, 46]. In particular, Woodruff and Yekhanin [46] presented a t -private ℓ -server PIR construction for general ℓ and t with communication complexity $\frac{\ell^2}{t} \cdot \log \ell \cdot N^{1/\lfloor (2\ell-1)/t \rfloor}$. Barkol, Ishai, and Weinreb [5] presented a general transformation from 1-private PIR protocols to t -private PIR protocol; given a 1-private ℓ -server PIR protocol with query length m_q and answer length m_a , they constructed a t -private ℓ^t -server PIR protocol with query length $O(tm_q)$ and answer length $O(m_a^t)$. When t is small compared to ℓ , this gives better protocols than [46], e.g., a t -private 2^t -server PIR protocol with communication complexity $2^{\tilde{O}(t\sqrt{\log(N)})}$ (using [20]) and a t -private 3^t -server PIR protocol with query length $t \cdot 2^{\tilde{O}(\sqrt{\log(N)})}$ and answer length $2^{O(t)}$ (using [22]).

1.3.3 Robust and Byzantine-Robust PIR

Beimel and Stahl [9] introduced robust and Byzantine robust PIR protocols. A PIR protocol is t -robust if the client can recover the correct value even if t of the servers go offline. The generalized notion of t -Byzantine robust requires robustness to hold even if t of the servers are malicious. This was further studied in subsequent works [46, 37, 23, 24]. In particular, Eriguchi et al. [24] showed that for a constant number of servers:

- The communication complexity of perfect t -Byzantine robust and t -private ℓ -server PIR is equivalent to the communication complexity of t -private $(\ell - 2t)$ -server PIR protocol;
- The communication complexity of statistical t -Byzantine robust and t -private ℓ -server PIR is equivalent to t -private $(\ell - t)$ -server PIR protocol.

Combining the above results and [46] yields a t -private and t -Byzantine robust ℓ -server statistical PIR protocol with total communication $\omega(\log N) \cdot N^{\frac{1}{\lfloor (2\ell-1)/t \rfloor - 2}}$. When t is relatively small compared to ℓ , the results of [5, 20, 22] yield a t -private and a t -Byzantine robust 2^t -server statistical PIR protocol with total communication complexity $2^{t^2 + \max\{\tilde{O}(t\sqrt{\log N}), t \log t\}}$ and a t -Byzantine robust 3^t -server statistical PIR protocol with total communication complexity $3^{t^2 + 2t} \cdot \max\{t 2^{\tilde{O}(\sqrt{\log N})}, t 3^t\}$. Byzantine robust PIR protocols where the database entries are large were considered in [47, 43, 44, 4]. They measure efficiency compared to the size of the entries rather than the size of the database (as we do in our work).

Eriguchi et al. [23, 24] introduced error-detecting PIR, where the client may abort and it is required that it never outputs the wrong value. In [23], they showed that error-detecting with perfect security and communication that is sublinear in the database size is impossible. In particular, this shows that there is no non-trivial perfect security-with-abort PIR protocol. In [24], they constructed a compiler transforming any t -private ℓ -server PIR protocol with communication c_0 to a t -error-detecting one with total communication $\ell^2 \cdot \omega(\log N) \cdot (c_0 + \ell \log \ell)$. When t is very small compared to ℓ , this has significantly better communication complexity compared to the Byzantine robust protocols discussed above. For example, there is a t -error-detecting 2^t -server PIR protocol with total communication complexity $2^{\max\{\tilde{O}(t\sqrt{\log N}), t \log t\}}$ and a t -error-detecting 3^t -server PIR protocol with total communication complexity $3^{2t} \cdot \max\{t2^{\tilde{O}(\sqrt{\log N})}, t3^t\}$. Additional constructions of error-detecting PIR protocols appear in Eriguchi et al. [25].

Colombo et al. [17] also observed that standard PIR security definitions allow for selective abort attacks. This motivated them to define *authenticated PIR protocols*, where, similarly to error-detection, it is required that either the client aborts or its output is consistent with the honest server's database. Additionally, the privacy of the client's retrieval index is required to hold even if the adversary knows whether the client aborts or not, which prevents selective abort attacks. Unlike our definitions, Colombo et al. [17] defined security by listing the desired security properties.

1.3.4 Computational PIR

Computational PIR protocols consider computationally-bounded servers. This model was first considered by Chor and Gilboa [15]. Computational single-server PIR protocols can be constructed with non-trivial communication complexity [38, 12, 26]. Computational PIR can be seen as a special case of fully homomorphic encryption; the result of Brakerski and Vaikuntanathan [11] used this to construct a single-server PIR protocol with total communication $\text{poly}(\kappa) + \log N$, where κ is the security parameter.

1.3.5 Spooky PIR

Dwork et al. [21] studied two-round succinct arguments for NP by composing PCP proofs with computational single-server PIR protocols. They showed that such heuristics may be insecure. More generally, they argued that executing PIR protocols in parallel may introduce “spooky interactions”: even though the queries made by the client are independent and the server knows nothing about the indexes of the client, the server may introduce correlations between the queries and outputs of the client. Dodis et al. [19] later showed that this holds even when using multi-prover interactive proofs instead of PCPs. These results show the necessity of defining the security of PIR protocols via the real vs. ideal world paradigm.

1.3.6 Verifiable PIR

Ben-David et al. [10] introduced verifiable PIR, where a (single) server should be able to prove that the database satisfies various properties. Although one of their definition follows the real vs. ideal world paradigm, it does not require privacy or security against selective abort attacks.

2 Preliminaries

2.1 Notations

We use bold characters to denote vectors. For $n \in \mathbb{N}$, let $[n] = \{1, 2, \dots, n\}$. For a set $\mathcal{S}$, we write $s \leftarrow \mathcal{S}$ to indicate that s is selected uniformly at random from $\mathcal{S}$. Given a random variable (or a distribution) X , we write $x \leftarrow X$ to indicate that x is selected according to X . PPT stands for probabilistic polynomial time. A function $\varepsilon(\cdot)$ is called negligible if for every positive polynomial $p(\cdot)$ and all sufficiently large n , $\varepsilon(n) < 1/p(n)$. For a vector $\mathbf{v}$ of dimension n , we write v_i for its i^{th} coordinate, and for $\mathcal{S} \subseteq [n]$ we write $\mathbf{v}_{\mathcal{S}} = (v_i)_{i \in \mathcal{S}}$.

A *distribution ensemble* $X = \{X_{a,n}\}_{a \in \mathcal{D}_n, n \in \mathbb{N}}$ is an infinite sequence of random variables indexed by $a \in \mathcal{D}_n$ and $n \in \mathbb{N}$, where $\mathcal{D}_n$ is a domain that might depend on n . Computational indistinguishability is defined as follows.

► **Definition 4.** Let $X = \{X_{a,n}\}_{a \in \mathcal{D}_n, n \in \mathbb{N}}$ and $Y = \{Y_{a,n}\}_{a \in \mathcal{D}_n, n \in \mathbb{N}}$ be two ensembles. We say that X and Y are computationally indistinguishable, denoted $X \stackrel{c}{\equiv} Y$, if for every PPT distinguisher D , there exists a negligible function $\varepsilon(\cdot)$, such that for all $n \in \mathbb{N}$ and $a \in \mathcal{D}_n$,

$$\left| \Pr[D(X_{a,n}) = 1] - \Pr[D(Y_{a,n}) = 1] \right| \leq \varepsilon(n).$$

► **Definition 5.** The statistical distance between two finite random variables X and Y is defined as

$$\text{SD}(X, Y) = \max_{\mathcal{T}} (\Pr[X \in \mathcal{T}] - \Pr[Y \in \mathcal{T}]).$$

Two ensembles $X = \{X_{a,n}\}_{a \in \mathcal{D}_n, n \in \mathbb{N}}$ and $Y = \{Y_{a,n}\}_{a \in \mathcal{D}_n, n \in \mathbb{N}}$ are said to be statistically close, denoted $X \stackrel{s}{\equiv} Y$, if there exists a negligible function $\varepsilon(\cdot)$, such that for all n and $a \in \mathcal{D}_n$,

$$\text{SD}(X_{a,n}, Y_{a,n}) \leq \varepsilon(n).$$

We say that X and Y are identically distributed, denoted $X \stackrel{p}{\equiv} Y$, if $\varepsilon(n) = 0$ for all $a \in \mathcal{D}_n$ and $n \in \mathbb{N}$.

► **Theorem 6** (Hoeffding's inequality for the hypergeometric distribution). Let $n \in \mathbb{N}$, $m, k \in [n]$, and $A \in \binom{[n]}{m}$. Then for every $t > 0$,

$$\Pr_{X \leftarrow \binom{[n]}{k}} \left[\left| X \cap A \right| - \frac{km}{n} \geq t \right] \leq e^{-2t^2/k}$$

and

$$\Pr_{X \leftarrow \binom{[n]}{k}} \left[\left| X \cap A \right| - \frac{km}{n} \leq -t \right] \leq e^{-2t^2/k}.$$

2.2 Private Information Retrieval

In this section, we start with the definition of single-round private information retrieval (PIR) protocols [16]. We then define property-based security taken from previous papers [9, 23, 24]. To simplify the presentation, we present the definition for a constant number of servers (i.e., independent of the database size). The definition readily extends to a non-constant number of servers. Intuitively, the PIR protocol starts with the client running a query algorithm $\mathcal{Q}$ and sending the j^{th} output q_j to the j^{th} server S_j . The j^{th} server, holding the database D and the query q_j , responds with the answer $\mathcal{A}(j, q_j, D)$. Finally, the client computes the output by applying the reconstruction algorithm $\mathcal{C}$. We next formalize this.

► **Definition 7** (PIR protocols). Let $\ell \in \mathbb{N}$. An ℓ -server PIR protocol is given by a 3-tuple $\Pi = (\mathcal{Q}, \mathcal{A}, \mathcal{C})$ of algorithms with the following syntax.

1. The query algorithm $(q_1, q_2, \dots, q_\ell, \mathbf{st}) \leftarrow \mathcal{Q}(1^N, i)$ is a randomized algorithm that is given the size of the database N and a retrieval index $i \in [N]$. It outputs a query q_j for every server $j \in [\ell]$ and a state $\mathbf{st} \in \{0, 1\}^*$. We denote $\mathbf{q} = (q_1, \dots, q_\ell)$.
2. The answer algorithm $a_j = \mathcal{A}(j, \mathbf{q}, D)$ is a deterministic algorithm that is given an index $j \in [\ell]$, a query $\mathbf{q}$, and a database $D \in \{0, 1\}^N$ for some $N \in \mathbb{N}$. It outputs a response a_j . We denote $\mathbf{a} = (a_1, \dots, a_\ell)$.
3. The reconstruction algorithm $y = \mathcal{C}(1^N, \mathbf{a}, \mathbf{st})$ is a deterministic algorithm that is given the size of the database N , the ℓ answers $\mathbf{a} = (a_j)_{j \in [\ell]}$, and the state $\mathbf{st}$. It outputs $y \in \{0, 1, \perp\}$.

The protocol is efficient if all three algorithms run in polynomial time in N . The total communication complexity is the maximum (taken over the randomness of the client and the databases D) of the total number of bits sent in the protocol, i.e., $\max_{r, D} \{\sum_{i=1}^{\ell} |q_i| + |a_i|\}$, where r is the randomness of $\mathcal{Q}$.

2.3 Defining Security for PIR via Security Properties

We next define the security properties of PIR protocols as they appear in the literature. We first define correctness, which states that in an honest execution, the client outputs the correct value.

► **Definition 8** (Correctness). Let $\ell \in \mathbb{N}$ and let $\Pi = (\mathcal{Q}, \mathcal{A}, \mathcal{C})$ be an ℓ -server PIR protocol. We say that Π is statistically correct if for any $N \in \mathbb{N}$, any database $D \in \{0, 1\}^N$, and any retrieval index $i \in [N]$,

$$\Pr_{(\mathbf{q}, \mathbf{st}) \leftarrow \mathcal{Q}(1^N, i)} \left[\mathcal{C} \left(1^N, (\mathcal{A}(j, \mathbf{q}_j, D))_{j \in [\ell]}, \mathbf{st} \right) = D_i \right] \geq 1 - \varepsilon(N),$$

for some negligible function $\varepsilon(\cdot)$, where the probability is over the randomness of $\mathcal{Q}$. If $\varepsilon = 0$ then we say that Π is perfectly correct.

We now define t -privacy, stating that no set of t servers learns anything about the retrieval index of the client.

► **Definition 9** (Privacy). Let $\ell \in \mathbb{N}$, let $t \in [\ell]$, and let $\Pi = (\mathcal{Q}, \mathcal{A}, \mathcal{C})$ be an ℓ -server PIR protocol. We say that Π is statistically t -private if for any set of t servers $\mathcal{I} \in \binom{[\ell]}{t}$, and any two sequences of indices $\{i_N\}_{N \in \mathbb{N}}$ and $\{i'_N\}_{N \in \mathbb{N}}$,

$$\{\mathbf{q}_{\mathcal{I}}\}_{N \in \mathbb{N}, D \in \{0, 1\}^N} \stackrel{\text{S}}{=} \{\mathbf{q}'_{\mathcal{I}}\}_{N \in \mathbb{N}, D \in \{0, 1\}^N},$$

where $(\mathbf{q}, \mathbf{st}) \leftarrow \mathcal{Q}(1^N, i_N)$ and $(\mathbf{q}', \mathbf{st}') \leftarrow \mathcal{Q}(1^N, i'_M)$. We define computationally/perfectly t -privacy by replacing $\stackrel{\text{S}}{=}$ with $\stackrel{\text{C}}{=}$ and $\stackrel{\text{P}}{=}$, respectively, in the above equation.

We next define security properties that should hold against malicious behavior by a subset of the servers. Following [23, 24], we first define tampering algorithms;³ this is an adversary that can modify some of the answers.

³ Eriguchi et al. [23, 24] defined tampering functions rather than algorithms.

► **Definition 10** (A tampering algorithm [23, 24]). Let $\ell \in \mathbb{N}$, $\mathcal{I} \subseteq [\ell]$, and let $\Pi = (\mathcal{Q}, \mathcal{A}, \mathcal{C})$ be an ℓ -server PIR protocol. A tampering algorithm $\mathcal{B}$ for Π is a randomized algorithm $\mathbf{a}' = (a'_j)_{j \in [\ell]} \leftarrow \mathcal{B}(\mathcal{I}, \mathbf{q}, D)$ that is given a set $\mathcal{I} \subseteq [\ell]$, a set of ℓ queries, and a database $D \in \{0, 1\}^N$ for some $N \in \mathbb{N}$. It outputs answers $(a'_j)_{j \in [\ell]}$ such that $a'_j = \mathcal{A}(j, q_j, D)$ for all $j \in [\ell] \setminus \mathcal{I}$, and a'_j depends on $\mathcal{I}$, $q_{\mathcal{I}}$, and D for all $j \in \mathcal{I}$.

We now define t -error-detecting. Roughly, it means that no set of t cheating servers can force the client to output the wrong value. Note, however, that the client is allowed to output $\perp$ (even with probability 1).

► **Definition 11** (Error-detecting PIR [23, 24]). Let $\ell \in \mathbb{N}$, $t \in [\ell]$, and let $\Pi = (\mathcal{Q}, \mathcal{A}, \mathcal{C})$ be an ℓ -server PIR protocol. We say that Π is statistical t -error-detecting if for any $N \in \mathbb{N}$, for any database $D \in \{0, 1\}^N$, for any retrieval index $i \in [N]$, any subset $\mathcal{I} \in \binom{[\ell]}{t}$ of t servers, and any tampering algorithm $\mathcal{B}$ for Π ,

$$\Pr_{(\mathbf{q}, \text{st}) \leftarrow \mathcal{Q}(1^N, i)} [\mathcal{C}(1^N, \mathcal{B}(\mathcal{I}, \mathbf{q}, D), \text{st}) \in \{D_i, \perp\}] \geq 1 - \varepsilon(N),$$

for some negligible function $\varepsilon(\cdot)$, where the probability is over the randomness of $\mathcal{Q}$ and $\mathcal{B}$. We define perfect t -error-detecting by requiring the above to hold for $\varepsilon(N) = 0$ for all $N \in \mathbb{N}$. We define computational t -error-detecting by requiring the above to hold only for PPT algorithms $\mathcal{B}$.

Finally, we define t -Byzantine robustness, which roughly means that the client outputs the correct value (i.e., D_i) even if at most t servers cheat. Note that any Byzantine robust PIR protocol is also correct, by taking $\mathcal{I} = \emptyset$.

► **Definition 12** (Byzantine robust PIR [9]). Let $\ell \in \mathbb{N}$, let $t \in [\ell]$, and let $\Pi = (\mathcal{Q}, \mathcal{A}, \mathcal{C})$ be an ℓ -server PIR protocol. We say that Π is statistical t -Byzantine robust if for any $N \in \mathbb{N}$, any database $D \in \{0, 1\}^N$, any retrieval index $i \in [N]$, any subset $\mathcal{I} \in \binom{[\ell]}{t}$ of t servers, and any tampering algorithm $\mathcal{B}$ for Π ,

$$\Pr_{(\mathbf{q}, \text{st}) \leftarrow \mathcal{Q}(1^N, i)} [\mathcal{C}(1^N, \mathcal{B}(\mathcal{I}, \mathbf{q}, D), \text{st}) = D_i] \geq 1 - \varepsilon(N),$$

for some negligible function $\varepsilon(\cdot)$, where the probability is over the randomness of $\mathcal{Q}$ and $\mathcal{B}$. We define perfect t -Byzantine robust by requiring the above to hold for $\varepsilon(N) = 0$ for all $N \in \mathbb{N}$. We define computational t -Byzantine robust by requiring the above to hold only for PPT algorithms $\mathcal{B}$.

3 Defining Security via The Real vs. Ideal World Paradigm

In this section, we present the security definition for PIR via the real vs. ideal world paradigm. We present the definition for both full security (i.e., with guaranteed output delivery) and for security-with-abort. We first define the real world execution, followed by the ideal worlds for full security and security-with-abort. Then, we present the security definitions. Our main contribution in the definition is defining the appropriate functionalities that are computed in the ideal world.

The Real World

Let $\ell \in \mathbb{N}$ and let $\Pi = (\mathcal{Q}, \mathcal{A}, \mathcal{C})$ be an ℓ -server PIR protocol. Toward defining security, we first describe an execution of Π in the presence of an adversary $\mathcal{B}$. The adversary controls a subset $\mathcal{I} \subseteq [\ell]$ of the servers. It receives the database $D \in \{0, 1\}^N$ and the queries the corrupted servers receive from the client, and instructs each server how to respond. We

consider *malicious* adversaries that can instruct the corrupted servers to respond to the client in an arbitrary way. At the end of the execution, the adversary outputs some function of its view (i.e., its randomness, the database, and the queries it received from the client). For a database size $N \in \mathbb{N}$, a database $D \in \{0, 1\}^N$, and a retrieval index $i \in [N]$, we let $\text{VIEW}_{\Pi, \mathcal{B}}^{\text{real}}(N, D, i)$ and $\text{OUT}_{\Pi, \mathcal{B}}^{\text{real}}(N, D, i)$ denote the outputs of $\mathcal{B}$ and the client respectively in an execution of Π . Finally, let

$$\text{REAL}_{\Pi, \mathcal{B}}(N, D, i) = \left(\text{VIEW}_{\Pi, \mathcal{B}}^{\text{real}}(N, D, i), \text{OUT}_{\Pi, \mathcal{B}}^{\text{real}}(N, D, i) \right).$$

The Ideal World – Full Security

We next describe an ideal computation with *guaranteed output delivery* (also referred to as full security) for PIR, where a trusted party T performs the computation on behalf of the parties, and the ideal model adversary cannot abort the computation. Note that we only consider adversaries that *do not* corrupt the client and corrupt a minority of the servers. The input of the adversary is the input of the servers it corrupts, i.e., the database D . An ideal computation with database size $N \in \mathbb{N}$, on input a database $D \in \{0, 1\}^N$ for the servers and retrieval index $i \in [N]$ for the client, in the presence of an adversary (a simulator) Sim controlling a set $\mathcal{I} \subseteq [\ell]$, proceeds as follows.

Parties send inputs to the trusted party: Each honest server S_j , i.e., $j \notin \mathcal{I}$, sends $D^j := D$ to the trusted party T . For each corrupted server S_j , i.e., $j \in \mathcal{I}$, the adversary Sim sends to T a database $D^j \in \{0, 1\}^N$ of its choice. If the server does not send any database, then T sets $D^j = 0^N$. The client sends i to T .

The trusted party performs the computation: If more than $1/2$ of the databases equal some D' , then the trusted party sends D'_i to the client. Otherwise, send D_i^1 to the client.

Output: The client outputs whatever it received from T and the adversary outputs some function of its view (i.e., its random string and the database D).

For a database size $N \in \mathbb{N}$, a database $D \in \{0, 1\}^N$, and a retrieval index $i \in [N]$, let $\text{VIEW}_{\text{Sim}}^{\text{god}}(N, D, i)$ and $\text{OUT}_{\text{Sim}}^{\text{god}}(N, D, i)$ denote the outputs of Sim and the client respectively in an execution of the above ideal world. Finally, let

$$\text{IDEAL}_{\text{Sim}}^{\text{god}}(N, D, i) = \left(\text{VIEW}_{\text{Sim}}^{\text{god}}(N, D, i), \text{OUT}_{\text{Sim}}^{\text{god}}(N, D, i) \right).$$

Note that the client always outputs D_i if there is an honest majority.

The Ideal World – Security-With-Abort

We next describe an ideal computation with *security-with-abort* for PIR. Unlike the ideal model for full security, here the adversary can abort the computation. An ideal computation with database size $N \in \mathbb{N}$, on input a database $D \in \{0, 1\}^N$ and index $i \in [N]$ for the client, in the presence of an adversary (a simulator) Sim controlling a set $\mathcal{I} \subseteq [\ell]$, proceeds as follows.

Parties send inputs to the trusted party: Each honest server S_j sends $D^j := D$ to the trusted party T . For each corrupted server S_j , the adversary Sim sends to T a database $D^j \in \{0, 1\}^N$ of its choice. If the server does not send any database, then T sets $D^j = 0^N$. The client sends i to T .

The trusted party performs the computation: If the databases that T received are not identical, then T sends $\perp$ to the client. Otherwise, it sends it D_i^1 .

Output: The client outputs whatever it received from T and the adversary outputs some function of its view (i.e., its random string and the database D).

8:12 On the Definition of Malicious PIR

For a database size $N \in \mathbb{N}$, a database $D \in \{0,1\}^N$, and a retrieval index $i \in [N]$, let $\text{VIEW}_{\text{Sim}}^{\text{swa}}(N, D, i)$ and $\text{OUT}_{\text{Sim}}^{\text{swa}}(N, D, i)$ denote the outputs of Sim and the client respectively in an execution of the above ideal world. Finally, let

$$\text{IDEAL}_{\text{Sim}}^{\text{swa}}(N, D, i) = (\text{VIEW}_{\text{Sim}}^{\text{swa}}(N, D, i), \text{OUT}_{\text{Sim}}^{\text{swa}}(N, D, i)).$$

Defining Security

Next, we define the security of a PIR protocol via the real vs. ideal world paradigm. We define full security and security-with-abort.

► **Definition 13** (Security via real vs. ideal paradigm). *Let $\ell \in \mathbb{N}$, $t \in [\ell]$, and let $\Pi = (\mathcal{Q}, \mathcal{A}, \mathcal{C})$ be an ℓ -server PIR protocol. We say that Π is statistically t -fully secure if for every adversary $\mathcal{B}$ controlling a set $\mathcal{I} \subseteq [\ell]$ of at most t servers, there exists a simulator Sim controlling the same subset $\mathcal{I}$ in the ideal world, such that*

$$\left\{ \text{IDEAL}_{\text{Sim}}^{\text{god}}(N, D, i) \right\}_{N \in \mathbb{N}, D \in \{0,1\}^N, i \in [N]} \stackrel{\text{S}}{=} \left\{ \text{REAL}_{\Pi, \mathcal{B}}(N, D, i) \right\}_{N \in \mathbb{N}, D \in \{0,1\}^N, i \in [N]}.$$

We say that Π is statistically t -secure-with-abort if the above holds with $\text{IDEAL}_{\text{Sim}}^{\text{swa}}(N, D, i)$ replacing $\text{IDEAL}_{\text{Sim}}^{\text{god}}(N, D, i)$.

Perfect t -security is defined by replacing $\stackrel{\text{S}}{=}$ with $\stackrel{\text{P}}{=}$ in the above equations. Computational t -security is defined by replacing $\stackrel{\text{S}}{=}$ with $\stackrel{\text{C}}{=}$ and limiting both the real-world adversary $\mathcal{B}$ and the ideal-world simulator Sim to be PPT algorithms.

► **Remark 14.** Standard security definitions following the real vs. ideal world paradigm consider adversaries that receive auxiliary information. The same auxiliary information is given to the real-world adversary and its simulator. This is necessary to argue that composition of secure protocols remains secure. If we were to add this to the security definition, then in order to argue that full security is equivalent to the privacy and Byzantine robustness properties (see Theorem 15 below), we would have had to add the auxiliary information to the privacy requirement (the statement and proof of equivalence remain the same). We decided not to include the auxiliary information in the privacy definition since this is not a common definition in the literature.

4 Comparing the Definitions

In this section, we compare the definitions given in the previous section. We start by showing the equivalence between full security and the property-based security definition. Specifically, we show that a PIR protocol is fully secure if and only if it is both private and Byzantine robust (recall that Byzantine robustness implies correctness). The interesting direction is showing the property-based security definition implies the real vs. ideal definition.

► **Theorem 15.** *Let $\ell, t \in \mathbb{N}$ be such that $t < \ell/2$, let $\Pi = (\mathcal{Q}, \mathcal{A}, \mathcal{C})$ be an ℓ -server PIR protocol, and let $\text{type} \in \{\text{computationally, statistically, perfectly}\}$. Then, Π is type t -fully secure if and only if it is type t -private and type t -Byzantine robust.*

Proof. We prove the results for $\text{type} = \text{statistically}$. The other cases are similar. We first prove that if Π is both statistically t -private and statistically t -Byzantine robust, then it is statistically t -fully secure. Fix a real-world adversary $\mathcal{B}$ corrupting a set $\mathcal{I}$ of at most t servers. We define its simulator Sim as follows. First, it sends to the trusted party the

database D for every corrupted party. It then computes $(\mathbf{q}^1, \text{st}^1) \leftarrow \mathcal{Q}(1^N, 1)$ (i.e., queries for the index 1), sends $\mathbf{q}_T^1$ to the adversary, and outputs whatever it outputs. Note that if $\mathcal{B}$ and $\mathcal{Q}$ are PPT algorithms, then Sim is a PPT algorithm.

We now show that the statistical difference between the real and ideal worlds is negligible. For a retrieval index $i \in [N]$ and a database $D \in \{0, 1\}^N$, let $y_{D,i}$ denote the output of the client in the real world. Note that it suffices to show that

$$\{(\mathbf{q}_T^i, y_{D,i})\}_{N \in \mathbb{N}, D \in \{0,1\}^N, i \in [N]} \stackrel{s}{=} \{(\mathbf{q}_T^1, D_i)\}_{N \in \mathbb{N}, D \in \{0,1\}^N, i \in [N]}, \quad (1)$$

where $(\mathbf{q}^i, \text{st}^i) \leftarrow \mathcal{Q}(1^N, i)$.

Fix an event $\mathcal{T}$ and let

$$\Delta = \Pr[(\mathbf{q}_T^i, y_{D,i}) \in \mathcal{T}] - \Pr[(\mathbf{q}_T^1, D_i) \in \mathcal{T}],$$

where the probabilities are over the execution of the real/ideal world. Then

$$\begin{aligned} \Delta &= \Pr[(\mathbf{q}_T^i, y_{D,i}) \in \mathcal{T}] - \Pr[(\mathbf{q}_T^1, D_i) \in \mathcal{T}] \\ &= \Pr[(\mathbf{q}_T^i, y_{D,i}) \in \mathcal{T}] - \Pr[(\mathbf{q}_T^i, D_i) \in \mathcal{T}] + \Pr[(\mathbf{q}_T^i, D_i) \in \mathcal{T}] - \Pr[(\mathbf{q}_T^1, D_i) \in \mathcal{T}]. \end{aligned}$$

Observe that by the t -privacy of Π ,

$$\Pr[(\mathbf{q}_T^i, D_i) \in \mathcal{T}] - \Pr[(\mathbf{q}_T^1, D_i) \in \mathcal{T}] = \text{neg}(N).$$

Indeed, if this was not the case, then

$$\Pr[\mathbf{q}_T^i \in \mathcal{T}_{D_i}] - \Pr[\mathbf{q}_T^1 \in \mathcal{T}_{D_i}]$$

is non-negligible, where $\mathcal{T}_{D_i} = \{\mathbf{q} : (\mathbf{q}, D_i) \in \mathcal{T}\}$. Therefore,

$$\begin{aligned} \Delta &\leq \Pr[(\mathbf{q}_T^i, y_{D,i}) \in \mathcal{T}] - \Pr[(\mathbf{q}_T^i, D_i) \in \mathcal{T}] + \text{neg}(N) \\ &= \Pr[(\mathbf{q}_T^i, D_i) \in \mathcal{T}] \cdot \Pr[y_{D,i} = D_i] + \Pr[(\mathbf{q}_T^i, 1 - D_i) \in \mathcal{T}] \cdot \Pr[y_{D,i} = 1 - D_i] \\ &\quad - \Pr[(\mathbf{q}_T^i, D_i) \in \mathcal{T}] + \text{neg}(N). \end{aligned}$$

By the t -Byzantine robustness of Π ,

$$\Pr[y_{D,i} = 1 - D_i] = \text{neg}(N).$$

Thus,

$$\begin{aligned} \Delta &\leq \Pr[(\mathbf{q}_T^i, D_i) \in \mathcal{T}] \cdot \Pr[y_{D,i} = D_i] + \Pr[(\mathbf{q}_T^i, 1 - D_i) \in \mathcal{T}] \cdot \Pr[y_{D,i} = 1 - D_i] \\ &\quad - \Pr[(\mathbf{q}_T^i, D_i) \in \mathcal{T}] + \text{neg}(N) \\ &= \Pr[(\mathbf{q}_T^i, D_i) \in \mathcal{T}] \cdot (1 - \text{neg}(N)) + \Pr[(\mathbf{q}_T^i, 1 - D_i) \in \mathcal{T}] \cdot \text{neg}(N) \\ &\quad - \Pr[(\mathbf{q}_T^i, D_i) \in \mathcal{T}] + \text{neg}(N) \\ &= \Pr[(\mathbf{q}_T^i, D_i) \in \mathcal{T}] - \Pr[(\mathbf{q}_T^i, D_i) \in \mathcal{T}] + \text{neg}(N) \\ &= \text{neg}(N). \end{aligned}$$

We now prove the second direction. Assume that Π is statistically t -fully secure. We first prove that Π is statistically t -private. Fix a set $\mathcal{I}$ of size t and consider the real-world adversary $\mathcal{B}$ controlling the servers in $\mathcal{I}$ that outputs the queries it received from the client. Then by assumption, there exists an ideal-world simulator Sim such that

$$\{\text{IDEAL}_{\text{Sim}}^{\text{god}}(N, D, i)\}_{N \in \mathbb{N}, D \in \{0,1\}^N, i \in [N]} \stackrel{s}{=} \{\text{REAL}_{\Pi, \mathcal{B}}(N, D, i)\}_{N \in \mathbb{N}, D \in \{0,1\}^N, i \in [N]}. \quad (2)$$

In particular,

$$\left\{ \text{VIEW}_{\text{Sim}}^{\text{god}}(N, D, i) \right\}_{N \in \mathbb{N}, D \in \{0,1\}^N, i \in [N]} \stackrel{s}{=} \left\{ \text{VIEW}_{\Pi, \mathcal{B}}^{\text{real}}(N, D, i) \right\}_{N \in \mathbb{N}, D \in \{0,1\}^N, i \in [N]}.$$

Now, as Sim does not receive any message from the trusted party, it follows that its output is independent of i . Thus,

$$\text{VIEW}_{\text{Sim}}^{\text{god}}(D, i) \equiv \text{VIEW}_{\text{Sim}}^{\text{god}}(D, i')$$

for all $D \in \{0,1\}^N$ and $i, i' \in [N]$. Therefore,

$$\left\{ \text{VIEW}_{\Pi, \mathcal{B}}^{\text{real}}(N, D, i) \right\}_{N \in \mathbb{N}, D \in \{0,1\}^N, i, i' \in [N]} \stackrel{c}{=} \left\{ \text{VIEW}_{\Pi, \mathcal{B}}^{\text{real}}(N, D, i') \right\}_{N \in \mathbb{N}, D \in \{0,1\}^N, i, i' \in [N]}.$$

Privacy follows from the fact that $\text{VIEW}_{\Pi, \mathcal{B}}^{\text{real}}(N, D, i) = \mathbf{q}_{\mathcal{I}}$, where $(\mathbf{q}, \mathbf{st}) \leftarrow \mathcal{Q}(1^N, i)$.

We now show that Π is statistically t -Byzantine robust. Fix a tampering algorithm $\mathcal{B}'$ and consider the adversary $\mathcal{B}$ that instructs the corrupted server to respond by computing $\mathcal{B}'$. Then by Equation (2),

$$\left| \Pr \left[\text{OUT}_{\text{Sim}}^{\text{god}}(N, D, i) = D_i \right] - \Pr \left[\text{OUT}_{\Pi, \mathcal{B}}^{\text{real}}(N, D, i) = D_i \right] \right| \leq \varepsilon(N),$$

for some negligible function $\varepsilon(\cdot)$. Since $t < \ell/2$, in the ideal world, the client always outputs D_i . Thus, $\Pr \left[\text{OUT}_{\Pi, \mathcal{B}}^{\text{real}}(N, D, i) = D_i \right] \geq 1 - \varepsilon(N)$. $\blacktriangleleft$

► **Remark 16.** Note that the above result holds for PIR protocols with more than a single round. To see this, let the simulator run the client on input 1 and simulate the entire execution of the protocol to generate the adversary's view. The same analysis shows that privacy and Byzantine robustness suffice to argue that the simulator succeeds.

For security-with-abort, we show that the real vs. ideal definition is *strictly stronger* than the property-based security. The next theorem states that security-with-abort implies the corresponding property-based security.

► **Theorem 17.** *Let $\ell, t \in \mathbb{N}$, let $\Pi = (\mathcal{Q}, \mathcal{A}, \mathcal{C})$ be an ℓ -server PIR protocol, and let $\text{type} \in \{\text{computationally}, \text{statistically}, \text{perfectly}\}$. If Π is type t -secure-with-abort, then it is type correct, type t -private, and type t -error-detecting.*

Proof. We assume that $\text{type} = \text{statistically}$ since the other cases are handled similarly. For correctness, consider an adversary that does not corrupt any server. Then the output of the client in the ideal world is D_i . Therefore, in the real world, the client outputs D_i except with negligible probability. The proof that Π is statistically t -private is identical to the proof done in Theorem 15 and is therefore omitted. We now show that Π is statistically t -error-detecting. Fix a tampering algorithm $\mathcal{B}'$ and consider the adversary $\mathcal{B}$ that instructs the corrupted server to respond by computing $\mathcal{B}'$. Then there is a simulator Sim such that

$$\left| \Pr \left[\text{OUT}_{\text{Sim}}^{\text{swa}}(N, D, i) = y \right] - \Pr \left[\text{OUT}_{\Pi, \mathcal{B}}^{\text{real}}(N, D, i) = y \right] \right| \leq \varepsilon(N),$$

for all $y \in \{0, 1, \perp\}$, where $\varepsilon(\cdot)$ is a negligible function. Since in the ideal world the client never outputs $1 - D_i$, it follows that $\Pr \left[\text{OUT}_{\Pi, \mathcal{B}}^{\text{real}}(N, D, i) \in \{D_i, \perp\} \right] \geq 1 - \varepsilon(N)$. $\blacktriangleleft$

We next show that the converse does not hold: there exists an ℓ -server PIR protocol Π that is correct, 1-private, and 1-error-detecting but is not computationally 1-secure-with-abort. Intuitively, we construct a protocol where a malicious server can force a “selective abort” of the client, that is, the adversary forces the client to abort if and only if it holds certain indices i (e.g., $i = 1$). This is impossible to simulate since the simulation is independent of i . We next formalize this intuition.

► **Example 18.** Let Π be a perfectly 1-secure-with-abort PIR protocol (e.g., each server sends D to the client; if all databases are the same, then the client outputs D_i , else it aborts). Consider the following PIR protocol Π' . The client computes the same queries as in Π , all servers respond with the same messages, and additionally, the first server sends an additional bit $b \in \{0, 1\}$. An honest server sends $b = 0$. If the client has input $i = 1$ and it received $b = 1$ then it outputs $\perp$. Otherwise, it proceeds as in Π . Clearly, Π' is correct, 1-private, and 1-error-detecting.

We next show that Π' is *not* 1-secure-with-abort. Intuitively, this is because the adversary can force a probability for abort on $i = 1$ that is different from the probability on $i \neq 1$ (although each server does not know i). Fix an adversary $\mathcal{B}$ that corrupts S_1 and sends $b = 1$, and assume towards contradiction that there exists a simulator Sim for it. Consider an execution of Π' with $i \leftarrow \{1, 2\}$ (i.e., $i = 1$ with probability $1/2$ and $i = 2$ with probability $1/2$) and $D = 1^N$. Then in the real world,

$$\Pr_{i \leftarrow \{1, 2\}} [i = 1, \text{OUT}_{\Pi', \mathcal{B}}^{\text{real}}(N, 1^N, i) = \perp] = \Pr_{i \leftarrow \{1, 2\}} [i = 2, \text{OUT}_{\Pi', \mathcal{B}}^{\text{real}}(N, 1^N, i) = 1] = \frac{1}{2}.$$

Let us analyze the ideal world. Let p denote the probability that Sim sends $D = 1^N$ to the trusted party (and thus the client outputs 1). Since in the ideal world the simulator is independent of i ,

$$\Pr_{i \leftarrow \{1, 2\}} [i = 1, \text{OUT}_{\text{Sim}}^{\text{swa}}(N, 1^N, i) = \perp] = \frac{1}{2} \cdot (1 - p)$$

and

$$\Pr_{i \leftarrow \{1, 2\}} [i = 2, \text{OUT}_{\text{Sim}}^{\text{swa}}(N, 1^N, i) = 1] = \frac{1}{2} \cdot p.$$

Clearly, it cannot be the case where both values are $1/2$ as in the real world, thus the two worlds can be easily distinguished.

One might expect that the reason the above example worked is because the client aborts with different probabilities for different inputs. We next show that adding such a requirement is still insufficient to argue for simulation-based security. Roughly, this is done by letting the client send to the first server a random index i' that will inform the server whether the client will abort on i' (if the additional bit b from the previous example is 1). Thus, in the real world, the adversary can choose for which inputs the client will abort, which cannot be simulated in the ideal world. We next formalize this.

► **Example 19.** Let Π be a perfectly 1-secure-with-abort PIR protocol (e.g., the servers send D to the client who then proceeds like the trusted party). Consider the following PIR protocol Π' . The client computes the same queries as in Π and appends the first query a random $i' \leftarrow [N]$ sampled uniformly at random. All servers respond with the same messages, and additionally, the first server sends an additional bit $b \in \{0, 1\}$. An honest server sends $b = 0$. If the client has input $i = i'$ and it received $b = 1$, then it outputs $\perp$. Otherwise, it proceeds as in Π . Clearly, Π' is correct, 1-private, 1-error-detecting, and the probability the client aborts is $1/N$ if the first server cheats (and 0 otherwise). The formal argument that Π' is not secure-with-abort is similar to the previous example and is therefore omitted.

5 A Generic Transformation From Private PIR to Secure-With-Abort PIR

In this section, we show a generic transformation of a private PIR protocol to a secure-with-abort protocol, based on Eriguchi et al. [24]. The original construction of [24] is *insecure* according to Definition 13 (see Remark 26 below). However, we prove that a modification of the construction is secure. We prove the following.

► **Theorem 20.** *Let $\ell \in \mathbb{N}$, $t \in [\ell]$, and $\text{type} \in \{\text{statistically}, \text{computationally}\}$. There exists a compiler that transforms any statistically correct and type t -private ℓ -server PIR protocol Π to a type t -secure-with-abort protocol. Moreover, if the total communication complexity of Π is c , then the new protocol has total communication complexity $\ell^2 \cdot \omega(\log N) \cdot (c + \ell \log \ell)$.*

Proof. We prove the result for $\text{type} = \text{statistically}$. The case where $\text{type} = \text{computationally}$ follows from the fact that the simulator we construct is efficient (if the adversary is) and from simple hybrid arguments. Let $\Pi = (\mathcal{Q}, \mathcal{A}, \mathcal{C})$ be a statistically correct and statistically t -private PIR protocol. The idea of the transformation is the following. First, the client generates the queries $\mathbf{q}$ as in Π . Then, with probability $1/2$, the client samples two indices $m_1, m_2 \in [\ell]$ at random such that $m_1 \neq m_2$, and sends q_j to the j^{th} server S_j for every $j \neq m_1$, and sends (m_2, q_{m_2}) to server S_{m_1} (i.e., query q_{m_2} is a duplicate query). With probability $1/2$, the client sends q_j to S_j for every $j \in [\ell]$ (without duplicating any of them). We refer to the first kind of execution, where the client duplicated a query, as a *test execution*, and we refer to the second kind of execution as a *real execution*. The j^{th} server, upon receiving q_j , responds as S_j does in Π , and upon receiving (m_2, q_{m_2}) , S_{m_1} responds as S_{m_2} does in Π .

Notice that if in a test execution, S_{m_1} is honest, then the view of the adversary in this case is identical to its view in a real execution. Thus, if the adversary cheats in the test execution, then it cheats in the real execution.

We then let the parties repeat this process in parallel sufficiently many times. The output of the client is defined as follows. If in one of the test executions, the client receives from the servers S_{m_1} and S_{m_2} different answers, then one of them is malicious and so the client aborts. Otherwise, the client computes the output of all of the real executions and outputs the majority. Intuitively, if we repeat this sufficiently many times, then the adversary cannot distinguish the real executions from most of the test executions. Therefore, if it cheats on too many real executions, then the client will catch it in one of the test executions. Otherwise, it will cheat on a minority of the real executions; hence, the correctness of the underlying PIR protocol implies that the client will compute the correct value most of the time.

For convenience, instead of deciding at random and independently which execution is a test and which is a real execution, we let the client randomly sample exactly half of the executions to be real.⁴ We next formalize this.

► **Protocol 21** ($\Pi_{\text{swa}} = (\mathcal{Q}_{\text{swa}}, \mathcal{A}_{\text{swa}}, \mathcal{C}_{\text{swa}})$).

Inputs: *The client holds the database length 1^N and a retrieval index $i \in [N]$. Each server holds a database $D \in \{0, 1\}^N$.*

Let $\lambda \in \mathbb{N}$ be an integer such that $\lambda/2$ is odd to be determined by the analysis below.

⁴ Note that if Π is perfectly correct, our protocol achieves only statistical correctness since it could be the case where the client duplicates a query in every execution of Π .

The query algorithm: The client samples a set $\mathcal{K} \leftarrow \binom{[\lambda]}{\lambda/2}$ uniformly at random and computes $(\mathbf{q}^1, \mathbf{st}^1), \dots, (\mathbf{q}^\lambda, \mathbf{st}^\lambda) \leftarrow \mathcal{Q}(1^N, i)$ independently. It then does the following for all $k \in [\lambda]$.

1. Sample a pair $m_1^k, m_2^k \in [\ell]$ of distinct elements independently and uniformly at random.
2. If $k \in \mathcal{K}$ then set $\hat{q}_j^k = q_j^k$ for all $j \in [\ell]$. Otherwise, for every $j \in [\ell] \setminus \{m_1^k\}$ set $\hat{q}_j^k = q_j^k$, and set $\hat{q}_{m_1^k}^k = (m_2^k, q_{m_2^k}^k)$.

Let $\mathbf{st} = \mathcal{K} || ((m_1^k, m_2^k, \mathbf{st}^k))_{k \in [\lambda]}$. The client sends $(\hat{q}_j^1, \dots, \hat{q}_j^\lambda)$ to the j^{th} server.

The answer algorithm: For every $k \in [\lambda]$ and $j \in [\ell]$, if server S_j receives $\hat{q}_j^k = q_j^k$ from the client, it computes $a_j^k = \mathcal{A}(j, \hat{q}_j^k, D)$. Else, it receives $\hat{q}_j^k = (m_2^k, q_{m_2^k}^k)$ and computes $a_j^k = \mathcal{A}(m_2^k, q_{m_2^k}^k, D)$. It sends $(a_j^1, \dots, a_j^\lambda)$ to the client.

The reconstruction algorithm: The client computes $y^k = \mathcal{C}(1^N, a_1^k, \dots, a_\ell^k, \mathbf{st}^k)$ for all $k \in \mathcal{K}$. Output $\perp$ if there exists $k \in [\lambda] \setminus \mathcal{K}$ such that $a_{m_1^k}^k \neq a_{m_2^k}^k$. Otherwise, the client outputs $\text{maj}_{k \in \mathcal{K}} \{y^k\}$.

We now show the security of Π_{swa} . Fix an adversary $\mathcal{B}$ corrupting a set $\mathcal{I} \subseteq [\ell]$ of at most t servers. We define its simulator Sim whose input is D as follows. Sample $(\hat{\mathbf{q}}, \mathbf{st}) \leftarrow \mathcal{Q}_{\text{swa}}(1^N, 1)$ and send $\hat{\mathbf{q}}_{\mathcal{I}}$ to $\mathcal{B}$. For every $j \in \mathcal{I}$ let a_j denote the response $\mathcal{B}$ made for server S_j , and for every $j \in [\ell] \setminus \mathcal{I}$ let $a_j = \mathcal{A}_{\text{swa}}(j, \hat{q}_j, D)$. Write $\mathbf{st} = \mathcal{K} || ((m_1^k, m_2^k, \mathbf{st}^k))_{k \in [\lambda]}$. If there exists $k \in [\lambda] \setminus \mathcal{K}$ such that $a_{m_1^k}^k \neq a_{m_2^k}^k$, then send to the trusted party a different database for the corrupted servers. Otherwise, send D as the input of every corrupted server. Finally, output whatever $\mathcal{B}$ outputs and halt. We show that

$$\{\text{IDEAL}_{\text{Sim}}^{\text{swa}}(N, D, i)\}_{N \in \mathbb{N}, D \in \{0,1\}^N, i \in [N]} \stackrel{\text{S}}{=} \{\text{REAL}_{\Pi_{\text{swa}}, \mathcal{B}}(N, D, i)\}_{N \in \mathbb{N}, D \in \{0,1\}^N, i \in [N]}. \quad (3)$$

To prove this, we first prove the following lemma, stating that $\mathcal{B}$ cannot force the client to output $1 - D_i$ with noticeable probability.

► **Lemma 22.** Let noAbort be the event that $a_{m_1^k}^k = a_{m_2^k}^k$ for every $k \in [\lambda] \setminus \mathcal{K}$ (i.e., the client did not output $\perp$), and let Fail denote the event that $\text{maj}_{k \in \mathcal{K}} \{y^k\} = 1 - D_i$ (i.e., the client output the wrong value). Then

$$\Pr[\text{noAbort} \wedge \text{Fail}] \leq 2e^{-\lambda/64} + e^{-\frac{\lambda}{8\ell(\ell-1)}} + \text{neg}(N), \quad (4)$$

where the probability is over the random coins of the client and the adversary.

Proof. We prove that the bound holds for any fixed randomness of the adversary. In the following, we assume the randomness of $\mathcal{B}$ and the queries of the client before duplication (i.e., $\mathbf{q}^1, \dots, \mathbf{q}^\lambda$) are fixed. Observe that for every $k \in [\lambda]$, the adversary cannot distinguish the case where $k \in \mathcal{K}$ and the case where $k \notin \mathcal{K}$ and $m_1^k \notin \mathcal{I}$. Therefore, if it instructs a server to cheat in one case, then it will instruct it to cheat in the other case.

To this end, we say that a server S_j *cheated in execution k* if its answer is different from $\mathcal{A}(j, q_j^k, D)$. That is, if it were to receive its original query before duplicating, then it would send a different answer. Let

$$A = \{k \in [\lambda] : \exists j \in \mathcal{I} \text{ s.t. } S_j \text{ cheated in execution } k\},$$

i.e., A denotes the set of all executions where the adversary cheated. As we fixed the randomness of the adversary and $\mathbf{q}^1, \dots, \mathbf{q}^\lambda$, the set A is properly defined. We show that if $|A|$ is too large, then the client will catch the adversary with overwhelming probability in one of the test executions, and if it is too small, then it is unlikely that the majority of the outputs computed for the real execution will be incorrect. We separate the proof into two cases.

We first analyze the probability in (4) assuming $|A| < 3\lambda/8$. We show that it is unlikely that the client outputs the wrong value. That is, we show **Fail** occurs with low probability. Observe that the expectation of $|A \cap \mathcal{K}|$ is at most $|A|/2 < 3\lambda/16$. By Hoeffding's inequality (Theorem 6), it follows that

$$\Pr_{\mathcal{K} \leftarrow \binom{[\lambda]}{\lambda/2}} \left[|A \cap \mathcal{K}| \geq \frac{\lambda}{4} \right] \leq \Pr_{\mathcal{K} \leftarrow \binom{[\lambda]}{\lambda/2}} \left[|A \cap \mathcal{K}| - \frac{3\lambda}{16} \geq \frac{\lambda}{16} \right] \leq e^{-\lambda/64}.$$

Let $k \in \mathcal{K} \setminus A$, i.e., the k^{th} execution is an honest real execution. By the statistical correctness of the original PIR protocol, the probability that the client outputs $1 - D_i$ in the k^{th} execution of the protocol is negligible. Therefore,

$$\begin{aligned} \Pr[\text{Fail}] &= \Pr[\text{maj}_{k \in \mathcal{K}} \{y^k\} = 1 - D_i] \\ &= \Pr \left[|A \cap \mathcal{K}| \geq \frac{\lambda}{4} \right] \cdot \Pr \left[\text{maj}_{k \in \mathcal{K}} \{y^k\} = 1 - D_i \mid |A \cap \mathcal{K}| \geq \frac{\lambda}{4} \right] \\ &\quad + \Pr \left[|A \cap \mathcal{K}| < \frac{\lambda}{4} \right] \cdot \Pr \left[\text{maj}_{k \in \mathcal{K}} \{y^k\} = 1 - D_i \mid |A \cap \mathcal{K}| < \frac{\lambda}{4} \right] \\ &\leq \Pr \left[|A \cap \mathcal{K}| \geq \frac{\lambda}{4} \right] + \Pr \left[\text{maj}_{k \in \mathcal{K}} \{y^k\} = 1 - D_i \mid |A \cap \mathcal{K}| < \frac{\lambda}{4} \right] \\ &\leq e^{-\lambda/64} + \Pr \left[\exists k \in \mathcal{K} \setminus A, y^k = 1 - D_i \mid |A \cap \mathcal{K}| < \frac{\lambda}{4} \right] \\ &\leq e^{-\lambda/64} + \text{neg}(N). \end{aligned}$$

We now analyze the probability in (4) assuming $|A| \geq 3\lambda/8$. We show that in this case, the client will catch the adversary cheating with high probability. Since the expectation of $|A \cap \bar{\mathcal{K}}|$ is at least $|A|/2 \geq 3\lambda/16$, by Hoeffding's inequality, it follows that

$$\Pr_{\mathcal{K} \leftarrow \binom{[\lambda]}{\lambda/2}} \left[|A \cap \bar{\mathcal{K}}| \leq \frac{\lambda}{8} \right] \leq \Pr_{\mathcal{K} \leftarrow \binom{[\lambda]}{\lambda/2}} \left[|A \cap \bar{\mathcal{K}}| - \frac{3\lambda}{16} \leq -\frac{\lambda}{16} \right] \leq e^{-\lambda/64}.$$

Next, for every $k \in A \cap \bar{\mathcal{K}}$ let **Caught_k** be the event that $m_1^k \notin \mathcal{I}$ and m_2^k cheated in execution k (that is, it is the event the adversary got caught cheating in the k^{th} execution). Fix an honest server $h \in [\ell] \setminus \mathcal{I}$, and for every $k \in A \cap \bar{\mathcal{K}}$ fix a server $c_k \in \mathcal{I}$ that cheated execution k . Observe that for every $k \in A \cap \bar{\mathcal{K}}$ the client will catch the adversary with probability at least

$$\Pr[\text{Caught}_k \mid k \in A] \geq \Pr[m_1^k = h, m_2^k = c_k \mid k \in A] \geq \frac{1}{\ell(\ell-1)}.$$

Now, observe that for every $k \in [\lambda] \setminus \mathcal{K}$, if $a_{m_1^k}^k = a_{m_2^k}^k$ then either the adversary did not cheat on the k^{th} execution or **Caught_k** did not occur. Therefore,

$$\begin{aligned} \Pr[\text{noAbort}] &= \Pr \left[\forall k \in [\lambda] \setminus \mathcal{K}, a_{m_1^k}^k = a_{m_2^k}^k \right] \\ &\leq \Pr \left[\forall k \in ([\lambda] \setminus \mathcal{K}) \cap A, \neg \text{Caught}_k \right] \\ &= \Pr \left[|A \cap \bar{\mathcal{K}}| \leq \frac{\lambda}{8} \right] \cdot \Pr \left[\forall k \in ([\lambda] \setminus \mathcal{K}) \cap A, \neg \text{Caught}_k \mid |A \cap \bar{\mathcal{K}}| \leq \frac{\lambda}{8} \right] \\ &\quad + \Pr \left[|A \cap \bar{\mathcal{K}}| > \frac{\lambda}{8} \right] \cdot \Pr \left[\forall k \in ([\lambda] \setminus \mathcal{K}) \cap A, \neg \text{Caught}_k \mid |A \cap \bar{\mathcal{K}}| > \frac{\lambda}{8} \right] \\ &\leq e^{-\lambda/64} + \Pr \left[\forall k \in A \cap \bar{\mathcal{K}}, \neg \text{Caught}_k \mid |A \cap \bar{\mathcal{K}}| > \frac{\lambda}{8} \right] \\ &\leq e^{-\lambda/64} + \left(1 - \frac{1}{\ell(\ell-1)} \right)^{\lambda/8} \\ &\leq e^{-\lambda/64} + e^{-\frac{\lambda}{8\ell(\ell-1)}}. \end{aligned}$$

We conclude that

$$\Pr[\text{noAbort} \wedge \text{Fail}] \leq 2e^{-\lambda/64} + e^{-\frac{\lambda}{8\ell(\ell-1)}} + \text{neg}(N). \quad \blacktriangleleft$$

We now use Lemma 22 to show that Equation (3) holds. Specifically, we show that conditioned that the event $\text{noAbort} \wedge \text{Fail}$ does *not* occur, the two ensembles are statistically close. First, note that by the t -privacy of Π , the queries that $\mathcal{B}$ receives in the real world are statistically close to the queries it receives from Sim in the ideal world. Therefore, its responses are statistically close. Now, if the client aborts in the real world, then there exists a test execution where it gets different responses. Thus, the same holds in the ideal world (except with negligible probability), hence the simulator changes its database. This means that in both worlds, the client outputs $\perp$. Otherwise, since we condition on $\text{noAbort} \wedge \text{Fail}$ not occurring, the client outputs D_i in both worlds. Finally, setting $\lambda = \omega(\ell^2 \cdot \log N)$ implies that $\text{noAbort} \wedge \text{Fail}$ occurs with negligible probability, which completes the proof. $\blacktriangleleft$

Using the t -private ℓ -server PIR protocol of Woodruff and Yekhanin [46] we obtain the following.

► **Corollary 23.** *For every $\ell \in \mathbb{N}$ and $t \in [\ell - 1]$ there exists a t -secure-with-abort ℓ -server PIR protocol with communication complexity*

$$\omega(\log N) \cdot \frac{\ell^4}{t} \cdot \log \ell \cdot N^{1/\lfloor (2\ell-1)/t \rfloor}.$$

Applying Theorem 20 to the protocol resulting from the compiler of Barkol et al. [5] applied to the 2-server protocol of Dvir and Gopi [20] yields the following.

► **Corollary 24.** *For every $t \in \mathbb{N}$, there exists a t -secure-with-abort 2^t -server PIR protocol with communication complexity*

$$2^{\max\{\tilde{O}(\sqrt{\log N}), t \log t\}}.$$

Finally, for 3^t -server PIR protocols, Barkol et al. [5] and Efremenko [22] gives the following.

► **Corollary 25.** *For every $t \in \mathbb{N}$, there exists a t -secure-with-abort 3^t -server PIR protocol with communication complexity*

$$3^{2t} \cdot \max\left\{t 2^{\tilde{O}(\sqrt{\log N})}, t \cdot 3^t\right\}.$$

► **Remark 26** (On the insecurity of the compiler of [24]). In the original construction of Eriguchi et al. [24], instead of computing $\text{maj}_{k \in \mathcal{K}}\{y^k\}$, the client also tests whether all of these values are equal. If not, it outputs $\perp$, and if they are equal, the client outputs this value. Although their protocol is correct, t -private, and t -error-detecting, it is not t -secure-with-abort. Intuitively, this is because in the underlying protocol Π the adversary can cause the client to output an incorrect value for different inputs i (recall that Π is not guaranteed to satisfy any robustness property). This causes the client to abort only on certain inputs; thus it has the same selective abort issue presented in Examples 18 and 19. We next give details.

Consider the protocol Π , where each server sends D to the client, and the client computes the output as follows. If $i = 1$ then output the i^{th} entry of the database sent by S_1 , and otherwise, output the i^{th} entry of the database sent by S_2 . Clearly this protocol is correct and private. However, the compiled protocol of Eriguchi et al. [24] is not 1-secure-with-abort since an adversary corrupting S_1 can force the client to output an incorrect value only on $i = 1$. Therefore, the client will abort only when $i = 1$.

References

- 1 Bar Alon, Amos Beimel, and Or Lasri. Simplified PIR and CDS protocols and improved linear secret-sharing schemes. Cryptology ePrint Archive, Paper 2024/1599, 2024. URL: <https://eprint.iacr.org/2024/1599>.
- 2 Andris Ambainis. Upper bound on the communication complexity of private information retrieval. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *Proc. of the 24th International Colloquium on Automata, Languages and Programming*, volume 1256 of *Lecture Notes in Computer Science*, pages 401–407, 1997. doi:10.1007/3-540-63165-8_196.
- 3 Hadar Amran. Constructing multi-servers private information retrieval protocols. Master’s thesis, Ben-Gurion University, 2016.
- 4 Karim A. Banawan and Sennur Ulukus. The capacity of private information retrieval from byzantine and colluding databases. *IEEE Trans. Inf. Theory*, 65(2):1206–1219, 2019. doi:10.1109/TIT.2018.2869154.
- 5 Omer Barkol, Yuval Ishai, and Enav Weinreb. On locally decodable codes, self-correctable codes, and t -private PIR. In Moses Charikar, Klaus Jansen, Omer Reingold, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 10th International Workshop, APPROX 2007, and 11th International Workshop, RANDOM 2007*, volume 4627 of *Lecture Notes in Computer Science*, pages 311–325. Springer, 2007. doi:10.1007/978-3-540-74208-1_23.
- 6 Amos Beimel and Yuval Ishai. Information-theoretic private information retrieval: A unified construction. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings*, volume 2076 of *Lecture Notes in Computer Science*, pages 912–926. Springer, 2001. doi:10.1007/3-540-48224-5_74.
- 7 Amos Beimel, Yuval Ishai, Ranjit Kumaresan, and Eyal Kushilevitz. On the cryptographic complexity of the worst functions. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *Lecture Notes in Computer Science*, pages 317–342. Springer, 2014. doi:10.1007/978-3-642-54242-8_14.
- 8 Amos Beimel, Yuval Ishai, and Eyal Kushilevitz. General constructions for information-theoretic private information retrieval. *Journal of Computer and System Sciences*, 71(2):213–247, 2005. doi:10.1016/j.jcss.2005.03.002.
- 9 Amos Beimel and Yoav Stahl. Robust information-theoretic private information retrieval. *J. Cryptol.*, 20(3):295–321, 2007. doi:10.1007/S00145-007-0424-2.
- 10 Shany Ben-David, Yael Tauman Kalai, and Omer Paneth. Verifiable private information retrieval. In Eike Kiltz and Vinod Vaikuntanathan, editors, *Theory of Cryptography – 20th International Conference, TCC 2022, Part III*, volume 13749 of *Lecture Notes in Computer Science*, pages 3–32. Springer, 2022. doi:10.1007/978-3-031-22368-6_1.
- 11 Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011*, pages 97–106. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.12.
- 12 Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT ’99*. Springer, 1999. doi:10.1007/3-540-48910-X_28.
- 13 Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000. doi:10.1007/S001459910006.
- 14 Yeow Meng Chee, Tao Feng, San Ling, Huaxiong Wang, and Liang Feng Zhang. Query-efficient locally decodable codes of subexponential length. *Computational Complexity*, 22(1):159–189, 2013. doi:10.1007/s00037-011-0017-1.

- 15 Benny Chor and Niv Gilboa. Computationally private information retrieval (extended abstract). In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pages 304–313. ACM, 1997. doi:10.1145/258533.258609.
- 16 Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science*, pages 41–50. IEEE Computer Society, 1995. doi:10.1109/SFCS.1995.492461.
- 17 Simone Colombo, Kirill Nikitin, Henry Corrigan-Gibbs, David J. Wu, and Bryan Ford. Authenticated private information retrieval. In Joseph A. Calandrino and Carmela Troncoso, editors, *32nd USENIX Security Symposium, USENIX Security 2023*, pages 3835–3851. USENIX Association, 2023. URL: <https://www.usenix.org/conference/usenixsecurity23/presentation/colombo>.
- 18 Casey Devet, Ian Goldberg, and Nadia Heninger. Optimally robust private information retrieval. In Tadayoshi Kohno, editor, *Proceedings of the 21th USENIX Security Symposium*, pages 269–283. USENIX Association, 2012. URL: <https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/devet>.
- 19 Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016 – 36th Annual International Cryptology Conference, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 93–122. Springer, 2016. doi:10.1007/978-3-662-53015-3_4.
- 20 Zeev Dvir and Sivakanth Gopi. 2-server PIR with subpolynomial communication. *J. ACM*, 63(4):39:1–39:15, 2016. doi:10.1145/2968443.
- 21 Cynthia Dwork, Michael Langberg, Moni Naor, Kobbi Nissim, and Omer Reingold. Succinct proofs for np and spooky interactions. Unpublished manuscript, <https://www.wisdom.weizmann.ac.il/~naor/PAPERS/spooky.pdf>, 2004.
- 22 Klim Efremenko. 3-query locally decodable codes of subexponential length. *SIAM J. Comput.*, 41(6):1694–1703, 2012. doi:10.1137/090772721.
- 23 Reo Eriguchi, Kaoru Kurosawa, and Koji Nuida. Multi-server PIR with full error detection and limited error correction. In Dana Dachman-Soled, editor, *3rd Conference on Information-Theoretic Cryptography, ITC 2022*, volume 230 of *LIPICs*, pages 1:1–1:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.ITC.2022.1.
- 24 Reo Eriguchi, Kaoru Kurosawa, and Koji Nuida. On the optimal communication complexity of error-correcting multi-server PIR. In Eike Kiltz and Vinod Vaikuntanathan, editors, *Theory of Cryptography – 20th International Conference, TCC 2022, Part III*, volume 13749 of *Lecture Notes in Computer Science*, pages 60–88. Springer, 2022. doi:10.1007/978-3-031-22368-6_3.
- 25 Reo Eriguchi, Kaoru Kurosawa, and Koji Nuida. Efficient and generic methods to achieve active security in private information retrieval and more advanced database search. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part V*, volume 14655 of *Lecture Notes in Computer Science*, pages 92–121. Springer, 2024. doi:10.1007/978-3-031-58740-5_4.
- 26 Craig Gentry and Zulfikar Ramzan. Single-database private information retrieval with constant communication rate. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005*, volume 3580 of *Lecture Notes in Computer Science*, pages 803–815. Springer, 2005. doi:10.1007/11523468_65.
- 27 Fatemeh Ghasemi, Swastik Kopparty, and Madhu Sudan. Improved pir schemes using matching vectors and derivatives. Technical Report 2411.11611, Cryptology ePrint Archive, 2024. doi:10.48550/arXiv.2411.11611.

- 28 Ian Goldberg. Improving the robustness of private information retrieval. In *2007 IEEE Symposium on Security and Privacy (S&P 2007)*, 20-23, pages 131–148. IEEE Computer Society, 2007. doi:10.1109/SP.2007.23.
- 29 Oded Goldreich. *Foundations of Cryptography – VOLUME 2: Basic Applications*. Cambridge University Press, 2004. doi:10.1017/CB09780511721656.
- 30 Yuval Ishai and Eyal Kushilevitz. Improved upper bounds on information-theoretic private information retrieval (extended abstract). In Jeffrey Scott Vitter, Lawrence L. Larmore, and Frank Thomson Leighton, editors, *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, pages 79–88. ACM, 1999. doi:10.1145/301250.301275.
- 31 Toshiya Itoh. Efficient private information retrieval. *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, E82-A(1):11–20, 1999.
- 32 Toshiya Itoh and Yasuhiro Suzuki. New constructions for query-efficient locally decodable codes of subexponential length. *CoRR*, abs/0810.4576, 2008. arXiv:0810.4576.
- 33 Toshiya Itoh and Yasuhiro Suzuki. Improved constructions for query-efficient locally decodable codes of subexponential length. *IEICE Transactions on Information and Systems*, E93-D(2):263–270, 2010. doi:10.1587/transinf.e93.d.263.
- 34 Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In F. Frances Yao and Eugene M. Luks, editors, *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pages 80–86. ACM, 2000. doi:10.1145/335305.335315.
- 35 Kiran S. Kedlaya and Sergey Yekhanin. Locally decodable codes from nice subsets of finite fields and prime factors of Mersenne numbers. Technical Report TR07-040, Electronic Colloquium on Computational Complexity, 2007. URL: <https://eccc.weizmann.ac.il/report/2007/040/>.
- 36 Iordanis Kerenidis and Ronald de Wolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument. *Journal of Computer and System Sciences*, 69(3):395–420, 2004. doi:10.1016/J.JCSS.2004.04.007.
- 37 Kaoru Kurosawa. How to correct errors in multi-server PIR. In Steven D. Galbraith and Shihō Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part II*, volume 11922 of *Lecture Notes in Computer Science*, pages 564–574. Springer, 2019. doi:10.1007/978-3-030-34621-8_20.
- 38 Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, 1997*, pages 364–373. IEEE Computer Society, 1997. doi:10.1109/SFCS.1997.646125.
- 39 Tianren Liu, Vinod Vaikuntanathan, and Hoeteck Wee. Conditional disclosure of secrets via non-linear reconstruction. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 758–790. Springer, 2017. doi:10.1007/978-3-319-63688-7_25.
- 40 Tianren Liu, Vinod Vaikuntanathan, and Hoeteck Wee. Towards breaking the exponential barrier for general secret sharing. In *EUROCRYPT 2018*, volume 10820 of *LNCS*, pages 567–596, 2018. doi:10.1007/978-3-319-78381-9_21.
- 41 Eran Mann. Private access to distributed information. Master’s thesis, Technion – Israel Institute of Technology, Haifa, 1998.
- 42 Silvio Micali and Phillip Rogaway. Secure computation (abstract). In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO '91, 11th Annual International Cryptology Conference*, volume 576 of *Lecture Notes in Computer Science*, pages 392–404. Springer, 1991. doi:10.1007/3-540-46766-1_32.
- 43 Hua Sun and Syed Ali Jafar. The capacity of private information retrieval. *IEEE Trans. Inf. Theory*, 63(7):4075–4088, 2017. doi:10.1109/TIT.2017.2689028.
- 44 Hua Sun and Syed Ali Jafar. The capacity of robust private information retrieval with colluding databases. *IEEE Trans. Inf. Theory*, 64(4):2361–2370, 2018. doi:10.1109/TIT.2017.2777490.

- 45 Stephanie Wehner and Ronald de Wolf. Improved lower bounds for locally decodable codes and private information retrieval. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Proc. of the 32nd International Colloquium on Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 1424–1436, 2005. doi:10.1007/11523468_115.
- 46 David P. Woodruff and Sergey Yekhanin. A geometric approach to information-theoretic private information retrieval. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005)*, pages 275–284. IEEE Computer Society, 2005. doi:10.1109/CCC.2005.2.
- 47 E.Y. Yang, Jie Xu, and K.H. Bennett. Private information retrieval in the presence of malicious failures. In *Proceedings 26th Annual International Computer Software and Applications*, pages 805–810, 2002. doi:10.1109/CMPSAC.2002.1045104.
- 48 Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 266–274, 2007. doi:10.1145/1250790.1250830.
- 49 Liang Feng Zhang and Reihaneh Safavi-Naini. Verifiable multi-server private information retrieval. In Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay, editors, *Applied Cryptography and Network Security – 12th International Conference, ACNS 2014, Lausanne, Switzerland*, volume 8479 of *Lecture Notes in Computer Science*, pages 62–79. Springer, 2014. doi:10.1007/978-3-319-07536-5_5.

Revocable Encryption, Programs, and More: The Case of Multi-Copy Security

Prabhanjan Ananth 

University of California, Santa Barbara, CA, USA

Saachi Mutreja

Columbia University, New York, NY, USA

Alexander Poremba 

Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

Fundamental principles of quantum mechanics have inspired many new research directions, particularly in quantum cryptography. One such principle is *quantum no-cloning* which has led to the emerging field of revocable cryptography. Roughly speaking, in a revocable cryptographic primitive, a cryptographic object (such as a ciphertext or program) is represented as a quantum state in such a way that surrendering it effectively translates into losing the capability to use this cryptographic object. All of the revocable cryptographic systems studied so far have a major drawback: the recipient only receives one copy of the quantum state. Worse yet, the schemes become completely insecure if the recipient receives many identical copies of the *same* quantum state – a property that is clearly much more desirable in practice. While multi-copy security has been extensively studied for a number of other quantum cryptographic primitives, it has so far received only little treatment in context of unclonable primitives. Our work, for the first time, shows the feasibility of revocable primitives, such as revocable encryption and revocable programs, which satisfy multi-copy security in oracle models. This suggests that the stronger notion of multi-copy security is within reach in unclonable cryptography more generally, and therefore could lead to a new research direction in the field.

2012 ACM Subject Classification Theory of computation → Cryptographic primitives

Keywords and phrases quantum cryptography, unclonable primitives

Digital Object Identifier 10.4230/LIPIcs.ITC.2025.9

Related Version *Full Version:* <https://eprint.iacr.org/2024/1687>

Funding *Prabhanjan Ananth:* Supported by the National Science Foundation under the grants FET-2329938 and the Career award (2341004).

Saachi Mutreja: Supported by AFOSR award FA9550-21-1-0040, NSF CAREER award CCF-2144219, and the Sloan Foundation.

Alexander Poremba: Supported by the U.S. Department of Energy, Office of Science, National Quantum Information Science Research Centers, Co-design Center for Quantum Advantage (C2QA) under contract number DE-SC0012704.

Acknowledgements The authors would like to thank Henry Yuen and Tal Malkin for many useful discussions.

1 Introduction

Designing mechanisms to provably revoke cryptographic capabilities is an age-old problem [46, 45]. In the public-key infrastructure, certificate authorities have the ability to invalidate public-key certificates [49], especially when the certificates have been compromised. Key rotation policies [29] guarantee that outdated decryption keys become ineffective for future use. The existing approaches to tackle with this problem have their limitations owing to the



© Prabhanjan Ananth, Saachi Mutreja, and Alexander Poremba;
licensed under Creative Commons License CC-BY 4.0
6th Conference on Information-Theoretic Cryptography (ITC 2025).
Editor: Niv Gilboa; Article No. 9; pp. 9:1–9:23



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

fact that cryptographic secrets are represented as binary strings and hence, it is infeasible to provably ensure that the malicious attackers have erased information from their devices. In the context of key rotation, a compromised key can still be used to decrypt old ciphertexts. Another issue with using classical information to represent cryptographic keys is that it is difficult to detect compromise: a hacker could steal the classical key from a device without leaving a trace.

Recently, a line of works [47, 11, 5, 13, 28, 41, 8] have leveraged quantum information and proposed new approaches for provable revocation of cryptographic objects, such as ciphertexts, programs and keys. These works studied revocation in the context of many Crypto 101 primitives, including pseudorandom functions, private-key and public-key encryption and digital signatures. In a revocable cryptographic primitive, an object (such as a program or a decryption key, etc.) is associated with a quantum state in such a way that *only with access to the state* the functionality of the original cryptographic object is retained. The common template for defining revocable security is in the form of a cryptographic game: The adversary receives *one copy* of the quantum state that it can use for a limited period of time after which it is supposed to return back the state to the owner. The security guarantee stipulates that after the state is returned, the adversary effectively loses the capability to use the cryptographic object. At this point, it should be clear to the reader the necessity that such cryptographic objects are represented as quantum states: indeed, if they were classical, the adversary could always maintain a secret copy, while pretending to have erased everything from its device. On the other hand, the no-cloning theorem [50, 32] of quantum mechanics suggests that the above security experiment could very well be achieved.

Multi-Copy Security

Let us now zoom in on the part of the security experiment, where the adversary receives *only one copy* of the quantum state. In all of the prior works in the literature so far [5, 13, 28, 41, 8], this limitation persists. One could consider a more general definition, where the adversary receives k *identical* copies of the quantum state and is later asked to return back all of the copies of the state. The security guarantee is similar to before: after returning all of the k copies, the adversary should effectively lose all access to the underlying cryptographic object. We term this general security experiment to be *multi-copy security*.

There are a couple of reasons to study multi-copy security for revocable primitives.

- **HISTORICAL CONTEXT:** Multi-copy security is not new and has been extensively studied in quantum cryptography, especially in the context of foundational primitives such as pseudorandom states [35, 23, 14, 7, 21, 22] and one-way state generators [43, 42]. Indeed, multi-copy security has been crucial in the design of many cryptographic constructions. The works of [7, 12, 37] used tomography, which inherently requires multiple copies in order to dequantize the communication in some of the quantum cryptographic primitives. Specifically for revocable primitives, a conceptual reason to study multi-copy security is to understand whether having more copies necessarily makes it easier for the adversary to clone quantum states. Investigating multi-copy security for revocable primitives is a starting step towards understanding multi-copy security for more advanced primitives such as public-key quantum money [3, 52]. The question of whether multi-copy security is possible in unclonable cryptography was also recently raised in [40].
- **NESTED LEASING:** Having access to many more copies of the quantum state would also give more power to the user; for example, using a permutation test [36] (a generalized version of SWAP test) where one is given $|\phi\rangle$ and polynomially many copies of $|\psi\rangle$, one can approximately test the overlap between $|\phi\rangle$ and $|\psi\rangle$. This ability allows for nested

leasing of cryptographic objects, such as programs or keys. Suppose a user **A** is leased a large number of, say k , copies of a quantum program $|\psi\rangle$. User **A** could further lease a number, say $k' \ll k$, of copies of $|\psi\rangle$ to user **B**. At a later point in time, when user **B** is asked to return back its copies to user **A**. User **A** can then use its $k - k'$ copies to approximately test whether the returned copies are correct. If the test succeeds, user **A** then is in a position to return back all of its k copies to the true owner¹. Such a nested leasing approach could be especially useful in organizations with hierarchical structure.

The notion of multi-copy security we consider in this work is closely related to *collusion-resistant security*, considered in the works of [39, 27]. The crucial difference is that in the prior works, the adversary receives *i.i.d* copies of the quantum key whereas in our case, the adversary receives *identical* copies of the quantum key. In the nested leasing application discussed above, it was crucial that the user received many *identical* copies of the same state.

Multi-copy security using commonly studied unclonable states: Challenges

The first step towards addressing multi-copy security is to identify quantum states that are unclonable. We discuss the commonly studied unclonable quantum states below:

- **BB84 STATES**: these states are of the form $H^\theta |x\rangle$, where $\theta \in \{0, 1\}^n$ and $x \in \{0, 1\}^n$. These states have been influential in the design of private-key quantum money [48] and in the design of encryption schemes with unclonable ciphertexts [26, 24]. Given many copies of $H^\theta |x\rangle$, one can learn x and θ and hence, recover a complete description of $H^\theta |x\rangle$.
- **SUBSPACE AND COSET STATES**: these states are of the form $(\sqrt{|A|})^{-1} \cdot \sum_{\mathbf{x} \in A} |\mathbf{x}\rangle$, where $A \subseteq \mathbb{F}_2^n$ is a sparse subspace of $\mathbb{F}_2^n$, for some $n \in \mathbb{N}$. These states have been crucial in the design of public-key quantum money [3, 52], among other primitives. Again given many copies, one can learn the basis of the subspace and hence a complete description of the subspace state. Another related class of unclonable states are coset states which are superpositions over a coset (rather than a subspace) and moreover, each term in the superposition has a phase that depends on a dual coset. Coset states have been influential in constructions of quantum copy-protection [30]. Similar to subspace states, coset states are also learnable.
- **SIS-BASED STATES**: these states are of the form $\sum_{\mathbf{x} \in \mathbb{Z}_q^m, \mathbf{A}\mathbf{x}=\mathbf{y}} \alpha_{\mathbf{x}} |\mathbf{x}\rangle$, where $q, m \in \mathbb{N}$, $|\alpha_{\mathbf{x}}|^2$ is a discrete Gaussian distribution such that most of the weight is on low norm vectors $\mathbf{x}$ and finally, $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{y} \in \mathbb{Z}_q^n$. They were useful in designing traditional and advanced encryption systems with unclonable quantum keys [44, 13, 41, 8]. Given many copies of this state, one can recover a short basis of the kernel of $\mathbf{A}$ which can then be used to recover the above state.

In other words, all the above types of states are learnable and hence, they cannot be the basis of any unclonable cryptographic scheme satisfying multi-copy security. This suggests that we need to look for new unclonable quantum states that are unlearnable even given many copies. In the past, discovering new unclonable quantum states has led to pushing the frontier of unclonable quantum cryptographic primitives and we believe our endeavour could reap similar results.

¹ A drawback of this approach is that there is some room for user **B** to cheat with noticeable probability in this approach without user **A** noticing, which means that the owner would not always be to pinpoint whether user **A** or user **B** cheated. Still, this approach offers a non-trivial solution to this challenging problem.

2 Our Results

We now give an overview of our results.

Our Approach: Quantum Pseudorandomness Meets Unclonable Cryptography

We use *subset states* to tackle multi-copy security. Subset states have been recently studied in the context of quantum pseudorandomness [33, 34, 2]. A subset state is associated with an unstructured and random subset $S \subset \{0, 1\}^n$ of the form $|S\rangle = (\sqrt{|S|})^{-1} \sum_{x \in S} |x\rangle$. Several recent works [33, 34] showed that random subset states (of non-trivial size) are approximate state k -designs for any polynomial k as a function of $n \in \mathbb{N}$, which make random subset states² a natural candidate for multi-copy security – particularly since Haar states are unlearnable given polynomially many identical copies. At first sight, it would seem as though that the fact that random subset states are close to Haar states should discourage us from using them for unclonable cryptography, especially since Haar states have virtually no structure. Indeed, the structure of BB84 states, subspace states and others has been crucially exploited in various applications. Our work shows that subset states, in some sense, have the minimal amount of structure to enable a number of interesting cryptographic primitives in the context of multi-copy revocable cryptography. To the best of our knowledge, these applications also mark the first use case of subset states in the context of cryptography. Our main technical contribution is of *information-theoretic* nature: we prove a query lower bound for *forging* subset elements; concretely, we show that any quantum algorithm that receives k copies of a random subset state $|S\rangle$ cannot produce $k + 1$ many subset elements in S unless it makes a large amount of queries to a membership oracle for S . We believe that this result could be of independent interest.

Multi-Copy Revocable Encryption. We first study revocable encryption [47, 5, 13] with multi-copy security. A revocable encryption scheme is a regular encryption scheme but where the ciphertexts are associated with quantum states. Additionally, a revocable encryption scheme comes with the following security notion called *multi-copy revocable security*: informally, it states that any adversary that successfully returns k valid copies of a quantum ciphertext which it was given by a challenger, where k is an arbitrary polynomial, necessarily loses the ability to decrypt the ciphertexts in the future – even if the secret key is revealed. In more detail, the security game is formulated as follows:

- The adversary selects a pair of messages (m_0, m_1) and sends them to the challenger.
- The challenger randomly selects one of the two messages, say m_b , and encrypts it k times using a secret key sk , and sends the ciphertext copies $|\psi_b\rangle^{\otimes k}$ to the adversary $\mathcal{A}$.
- At a later point in time, $\mathcal{A}$ returns back all the copies of $|\psi_b\rangle$, which are then verified by the challenger.
- After successfully returning back the states, $\mathcal{A}$ receives the secret key sk in the clear. Finally, $\mathcal{A}$ outputs a guess b' .

The scheme is said to be secure if the probability that $b' = b$ is close to $1/2$.

Prior works [47, 5, 13, 28, 8] only studied variants of the above security game in the setting where the adversary receives only one copy of the quantum ciphertext. In fact, their schemes are easily seen to not satisfy multi-copy security. We show the following.

² Strictly speaking, we use *pseudorandom* subset states which can be generated efficiently via pseudorandom permutations.

► **Theorem 1.** *If post-quantum one-way functions exist, then there exists an encryption scheme with (an oracular notion of) multi-copy revocable security.*

Some remarks are in order. Firstly, our security proof does not fully achieve the standard notion of revocable security guarantee we stated above; rather, we consider a slightly different variant of the experiment, where in the second part of the game (instead of revealing the secret key in the clear) we allow the adversary to query an oracle that is powerful enough to enable decryption during the first phase of the game. While this constitutes a weaker notion of security, it nevertheless results in a meaningful notion of revocable security: once the adversary has successfully returned all of the copies of the ciphertext, it can no longer decrypt the ciphertext in the future – even if it gets access to an oracle that would have previously allowed it to do so. Second, our construction of multi-copy revocable encryption makes use of quantum-secure pseudorandom permutations (QPRPs), which can be constructed from post-quantum one-way functions [51]. In the security experiment which underlies our construction, the aforementioned oracle for decryption (i.e., that which is handed to the adversary after revocation has taken place) is in the form of an ideal oracle for the permutation itself. Once again, the rationale behind our notion of oracular security is that an attacker who receives a QPRP key in the clear would most certainly use it to evaluate the QPRP, and hence it is reasonable to consider a model in which the attacker receives an oracle for the permutation instead.³ A key advantage of oracular security is that we can directly invoke the security of the QPRP and use a perfectly random permutation instead.⁴ We remark that this model loosely resembles the random permutation model behind the international hash function standard SHA-3 [19, 20], except that the adversary only receives oracle access to the permutation during the second part of the revocable security experiment. While our construction only achieves an oracular notion of revocable security, it is nevertheless the very first construction of revocable encryption which satisfies multi-copy security *in any model*.

Multi-Copy Revocable Programs

Our previous discussion on revocable encryption illustrates that encryption and decryption functionalities can be protected even if many copies of the quantum ciphertext are made available to the recipient. We generalize this result further and study whether arbitrary functionalities can be protected. We define and study revocable programs with multi-copy security. In this notion, there is a functionality preserving compiler that takes a program and converts it into a quantum state. The security guarantee is defined similar to revocable encryption:

- The challenger compiles a program P , sampled from a distribution $\mathcal{D}$ on a set of programs $\mathcal{P}$, into a state $|\psi_P\rangle$. It then sends k copies of the state $|\psi_P\rangle$ to the adversary $\mathcal{A}$.
- At a later point in time, $\mathcal{A}$ returns back all of the copies of $|\psi_P\rangle$.
- After returning back the state, $\mathcal{A}$ is given x , where x is sampled from the input distribution of P . It then outputs a guess y .

The scheme is said to be secure if the probability that $y = P(x)$ is roughly close to the trivial success probability. Here, the trivial success probability is defined as the optimal probability of guessing $P(x)$ given just x (and the knowledge of $\mathcal{D}$ and the input distribution).

³ Note, however, that this does not capture all possible attacks; for example, the adversary could use its knowledge of the QPRP key to break the scheme in other meaningful ways.

⁴ This switch is generally not possible in the standard notion of revocable security in which the QPRP key is required to be revealed in the clear. Here, QPRP security does not apply.

Prior works propose revocable programs for specific functionalities in the plain model [30] or for general functionalities in oracle models [4]. However, these works guarantee security only if the adversary receives one copy of the state. And as before, these constructions provably do not satisfy multi-copy security. We show the following.

► **Theorem 2.** *There exist revocable programs which satisfy (an oracular notion of) multi-copy security in a classical oracle model.*

Unlike Theorem 1, the above theorem relies upon structured and ideal classical oracles.

Finally, for the special case of point functions, we show that we can again only rely upon pseudorandom permutations together with the (standard) quantum random oracle model. We show the following.

► **Theorem 3.** *There exist revocable multi-bit point functions which satisfy (an oracular notion of) multi-copy security in quantum random oracle model.*

Our results and techniques opens the door for building more advanced unclonable primitives that preserve their security even if the adversary receives many copies of the unclonable quantum state.

Are Results in Oracle Models Interesting?

It is natural for a reader to be skeptical of our results given that they are based in the oracle models. However, we would like to emphasize that achieving results in the oracle models still requires non-trivial amount of effort. As history suggests, constructions in the oracle models have eventually been adopted to constructions in the plain model. A classic example is the construction of public-key quantum money, which was first proposed in the oracle models by Aaronson and Christiano [3] and later, being instantiated in the plain model by Zhandry [52]. In a similar vein, our techniques could be useful for future works on achieving multi-copy security in the plain model.

Applications to Sponge Hashing

As a complementary contribution, we show that the techniques we developed in this paper are more broadly applicable and extend to other cryptographic settings as well. Here, we single out the so-called *sponge construction* used in SHA-3 [19, 20].

We study a simple query problem: Suppose that an adversary receives as input a hash table for a set of random input keys, where each hash is computed using a *salted* (one-round) sponge hash function. How many quantum queries are necessary to find a new *valid* element in the range of the hash function? Our contribution is a space-time trade-off which precisely characterizes the hardness of finding hash table elements in the presence of oracles that depend non-trivially on the sponge hash function.

2.1 Related work

We now discuss related notions which are relevant to this work.

Copy-Protection

This notion was first introduced by Aaronson [1]. Informally speaking, a copy-protection scheme is a compiler that transforms programs into quantum states in such a way that using the resulting states, one can run the original program. Yet, the security guarantee stipulates that any adversary given one copy of the state cannot produce a bipartite state

wherein both parts compute the original program. Copy-protection schemes have since been constructed for various classes of programs and under various different models, for example as in [1, 4, 31, 9, 10, 39, 27]. We remark, however, that all of the aforementioned works are completely insecure if multiple identical copies of the program are made available. The notion of multi-copy security we consider in this work is closely related to *collusion-resistant security*, considered in the works of [39, 27]. The crucial difference is that in the prior works, the adversary receives *i.i.d* copies of the quantum key whereas in our case, the adversary receives *identical* copies of the quantum key.

Secure Software Leasing

Another primitive relevant to revocable cryptography is secure software leasing [11]. The notion of secure software leasing states that any program can be compiled into a functionally equivalent program, represented as a quantum state, in such a way that once the compiled program is returned, the (honest) evaluation algorithm on the residual state cannot compute the original functionality. Secure leasing has been constructed for various functionalities [11, 31, 25, 38]. Similar to copy-protection, none of the aforementioned works consider multi-copy security.

Encryption Schemes with Revocable Ciphertexts

Unruh [47] proposed a (private-key) quantum timed-release encryption scheme that is *revocable*, i.e. it allows a user to *return* the ciphertext of a quantum timed-release encryption scheme, thereby losing all access to the data. Broadbent and Islam [24] introduced the notion of *certified deletion*, which is incomparable with the related notion of unclonable encryption. This has led to the development of other certified deletion protocols, for example as in Ref. [44, 15, 17, 16, 18]. However, the notion of multi-copy security, such as in our work, has not been studied.

3 Preliminaries

Let $\lambda \in \mathbb{N}$ denote the security parameter throughout this work. We assume that the reader is familiar with the fundamental cryptographic concepts.

For $N \in \mathbb{N}$, we use $[N] = \{1, 2, \dots, N\}$ to denote the set of integers up to N . The symmetric group on $[N]$ is denoted by S_N . In slight abuse of notation, we oftentimes identify elements $x \in [N]$ with bit strings $x \in \{0, 1\}^n$ via their binary representation whenever $N = 2^n$ and $n \in \mathbb{N}$. Similarly, we identify permutations $\pi \in S_N$ with permutations $\pi : \{0, 1\}^n \rightarrow \{0, 1\}^n$ over bit strings of length n . For a bit string $x \in \{0, 1\}^n$, we frequently use the notation $(x||*)$, where $*$ serves as a placeholder to denote the set $\{(x||y) : y \in \{0, 1\}^m\}$, where $m \in \mathbb{N}$ is another integer which is typically clear in context.

We write $\text{negl}(\cdot)$ to denote any *negligible* function, which is a function f such that, for every constant $c \in \mathbb{N}$, there exists an integer N such that for all $n > N$, $f(n) < n^{-c}$.

► **Lemma 4** (One-Way-to-Hiding Lemma, [6]). *Let $\mathcal{X}, \mathcal{Y}$ be arbitrary sets and let $\mathcal{S} \subseteq \mathcal{X}$ be a (possibly random) subset. Let $G, H : \mathcal{X} \rightarrow \mathcal{Y}$ be arbitrary (possibly random) functions such that $H(x) = G(x)$, for all $x \notin \mathcal{S}$. Let z be a classical bit string or a (possibly mixed) quantum state (Note that $G, H, \mathcal{S}, z$ may have arbitrary joint distribution). Let $\mathcal{A}$ be an oracle-aided quantum algorithm that makes at most q quantum queries. Let $\mathcal{B}$ be an algorithm that on input z chooses a random query index $i \leftarrow [q]$, runs $\mathcal{A}^H(z)$, measures $\mathcal{A}$'s i -th query and outputs the measurement outcome. Then, we have*

$$|\Pr[\mathcal{A}^G(z) = 1] - \Pr[\mathcal{A}^H(z) = 1]| \leq 2q\sqrt{\Pr[\mathcal{B}^H(z) \in \mathcal{S}]}.$$

Moreover, for any fixed choice of G, H, S and z (when z is a classical string or a pure state), we get

$$\| |\psi_q^H\rangle - |\psi_q^G\rangle \| \leq 2q \sqrt{\frac{1}{q} \sum_{i=0}^{q-1} \|\Pi_S |\psi_i^H\rangle\|^2},$$

where $|\psi_i^H\rangle$ denotes the intermediate state of $\mathcal{A}$ just before the $(i+1)$ -st query, where the initial state at $i=0$ corresponds to z , and Π_S is a projector onto S .

Pseudorandom Permutations

A quantum-secure pseudorandom permutation is a bijective function family which can be constructed from quantum-secure one-way functions [51].

► **Definition 5 (QPRP).** Let $\lambda \in \mathbb{N}$ denote the security parameter. Let $P : \{0,1\}^\lambda \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a function, where $n(\lambda) = \text{poly}(\lambda)$ is an integer, such that each function $P_k(x) = P(k, x)$ in the corresponding family $\{P_k\}_{k \in \{0,1\}^\lambda}$ is bijective. We say P is a (strong) quantum-secure pseudorandom permutation (or QPRP) if, for every QPT $\mathcal{A}$ with access to both the function and its inverse, it holds that

$$\left| \Pr_{k \sim \{0,1\}^\lambda} [\mathcal{A}^{P_k, P_k^{-1}}(1^\lambda) = 1] - \Pr_{\varphi \sim \mathcal{P}_n} [\mathcal{A}^{\varphi, \varphi^{-1}}(1^\lambda) = 1] \right| \leq \text{negl}(\lambda),$$

where $\mathcal{P}_n$ denotes the set of permutations over n -bit strings.

Subset States

We consider the following notations.

- We denote the set of distinct k -tuples over a set S by $\text{dist}(S, k)$.
- Suppose S is a set. We denote $|S\rangle = \frac{1}{\sqrt{|S|}} \sum_{x \in S} |x\rangle$.
- Suppose $X = \{x_1, \dots, x_t\} \subseteq \{0,1\}^n$. We denote $|\sigma_X\rangle = \frac{1}{t!} \sum_{\sigma \in S_t} |x_{\sigma(1)}, \dots, x_{\sigma(t)}\rangle$, where S_t denotes the symmetric group on $[t]$.

We use the following lemma which follows from Propositions 3.3 and 3.4 in [34].

► **Lemma 6 ([34]).** Let $n, k \in \mathbb{N}$. Let $T \subseteq \{0,1\}^n$ be a subset of size $|T| = t$. Then, it holds that

$$\text{TD} \left(\mathbb{E}_{\substack{S \subseteq T \\ |S|=s}} [|S\rangle \langle S|^{\otimes k}], \mathbb{E}_{\substack{X \subseteq T \\ |X|=k}} [|\sigma_X\rangle \langle \sigma_X|] \right) \leq O \left(\frac{k}{\sqrt{s}} + \frac{sk}{t} \right).$$

4 $k \mapsto k+1$ Unforgeability of Subset States

We now prove the following theorem. Roughly speaking, our theorem says that any quantum algorithm which receives k copies of a random subset state $|S\rangle$ (and a membership oracle for S) cannot find $k+1$ distinct elements in S with high probability unless it makes a large number of queries.

► **Theorem 7** ($k \mapsto k + 1$ Unforgeability of Subset States). *Let $n \in \mathbb{N}$ and $k \in \mathbb{N}$. Then, for any q -query quantum oracle algorithm $\mathcal{A}$, and any $1 \leq k < s < t \leq 2^n$, it holds that*

$$\Pr_{\substack{S \subseteq \{0,1\}^n \\ |S|=s}} \left[(x_1, \dots, x_{k+1}) \in \text{dist}(S, k+1) : (x_1, \dots, x_{k+1}) \leftarrow \mathcal{A}^{\mathcal{O}_S}(|S\rangle^{\otimes k}) \right] \\ \leq O \left(q \cdot \sqrt{\frac{t-s}{2^n}} + q \cdot \sqrt{\frac{t-k}{2^n}} + \frac{k}{\sqrt{s}} + \frac{sk}{t} \right) + \text{negl}(n).$$

In particular, we can let $k = \text{poly}(n)$, $q = \text{poly}(n)$ and $s(n) = n^{\omega(1)}$ be superpolynomial. Then, for any $t(n) = n^{\omega(1)}$ with $s(n)/t(n) = 1/n^{\omega(1)}$ and $t(n)/2^n = 1/n^{\omega(1)}$, the probability is at most $\text{negl}(n)$.

Proof. Using Lemma 8, we can prove Theorem 7 as follows: Let's assume for contradiction that there exists a q query quantum oracle algorithm $\mathcal{A}$, and a $1 \leq k < s < t \leq 2^n$, such that

$$\Pr_{\substack{S \subseteq \{0,1\}^n \\ |S|=s}} \left[(x_1, \dots, x_{k+1}) \in \text{dist}(S, k+1) : (x_1, \dots, x_{k+1}) \leftarrow \mathcal{A}^{\mathcal{O}_S}(|S\rangle^{\otimes k}) \right] \\ = O \left(q \cdot \sqrt{\frac{t-s}{2^n}} + q \cdot \sqrt{\frac{t-k}{2^n}} + \frac{k}{\sqrt{s}} + \frac{sk}{t} \right) + \delta(n).$$

where $\delta(\cdot)$ is some non negligible function. Then, from Lemma 8, this implies that,

$$\Pr_{X \subseteq \{0,1\}^n, |X|=k} \left[(x_1, \dots, x_{k+1}) \in \text{dist}(X, k+1) : (x_1, \dots, x_{k+1}) \leftarrow \mathcal{A}^{\mathcal{O}_X}(|\sigma_X\rangle) \right] \\ \geq \delta(n)$$

However, the probability that $\mathcal{A}$ can succeed in this experiment is 0. This is because, X only contains k elements, and therefore, $\mathcal{A}$ can never produce $k + 1$ distinct elements from X . This must imply that $\delta(n)$ is negligible. ◀

Technical Lemma

We need to show the following lemma which made use of in Theorem 7.

► **Lemma 8.** *Let $n, k, t \in \mathbb{N}$ be integers such that $1 \leq k < s < t \leq 2^n$. Then, for any q -query quantum oracle algorithm $\mathcal{D}$ which outputs a single bit, it holds that*

$$\left| \Pr_{\substack{S \subseteq \{0,1\}^n \\ |S|=s}} \left[\mathcal{D}^{\mathcal{O}_S}(|S\rangle^{\otimes k}) = 1 \right] - \Pr_{\substack{X \subseteq \{0,1\}^n \\ |X|=k}} \left[\mathcal{D}^{\mathcal{O}_X}(|\sigma_X\rangle) = 1 \right] \right| \\ \leq O \left(q \cdot \sqrt{\frac{t-s}{2^n}} + q \cdot \sqrt{\frac{t-k}{2^n}} + \frac{k}{\sqrt{s}} + \frac{sk}{t} \right).$$

Proof. Consider the following hybrid distributions.

H₁. Output $\mathcal{D}^{\mathcal{O}_S}(|S\rangle^{\otimes k})$, where $S \subseteq \{0,1\}^n$ is a random subset of size $|S| = s$.

H₂. Output $\mathcal{D}^{\mathcal{O}_S}(|S\rangle^{\otimes k})$, where the subset S is sampled as follows: first, sample a random subset $T \subseteq \{0,1\}^n$ of size $|T| = t$, and then let $S \subseteq T$ be a random subset of size $|S| = s$.

Let $p(\mathbf{H}_i)$ be the probability that $\mathbf{H}_i$ outputs 1, for some i . We now show the following.

▷ **Claim 9.** $p(\mathbf{H}_2) = p(\mathbf{H}_1)$.

Proof. The distribution of sampling $S \subseteq \{0,1\}^n$ of size $|S| = s$ is identical to the distribution of first sampling a superset $T \subseteq \{0,1\}^n$ of size $|T| = t$, and letting $S \subseteq T$ be a random subset of size $|S| = s$. ◀

9:10 Revocable Encryption, Programs, and More: The Case of Multi-Copy Security

H₃. Output $\mathcal{D}^{\mathcal{O}_T}(|S\rangle^{\otimes k})$, where the subset S is sampled as follows: first, sample a random subset $T \subseteq \{0,1\}^n$ of size $|T| = t$, and then let $S \subseteq T$ be a random subset of size $|S| = s$.

▷ Claim 10.

$$|p(\mathbf{H}_3) - p(\mathbf{H}_2)| \leq O\left(q \cdot \sqrt{\frac{t-s}{2^n}}\right).$$

Proof. We can model the quantum oracle algorithm $\mathcal{D}^{\mathcal{O}_S}$ on input $|S\rangle^{\otimes t}$ as a sequence of oracle queries and unitary computations followed by a measurement. Thus, the final output state just before the measurement can be written as

$$|\Psi_q^S\rangle = U_q \mathcal{O}_S U_{q-1} \dots U_1 \mathcal{O}_S U_0 |\psi_0\rangle |S\rangle^{\otimes k},$$

where $U_0, U_1, \dots, U_q$ are unitaries (possibly acting on additional workspace registers, which we omit above), and where $|\psi_0\rangle$ is some fixed initial state which is independent of S .

In the next step of the proof, we will use the “subset flooding” technique to drown S in a random superset. Let $T \subseteq \{0,1\}^n$ be a random superset of S of size $t > s$. We now consider the state

$$|\Psi_q^T\rangle = U_q \mathcal{O}_T U_{q-1} \dots U_1 \mathcal{O}_T U_0 |\psi_0\rangle |S\rangle^{\otimes k}.$$

We now claim that the states $|\Psi_q^S\rangle$ and $|\Psi_q^T\rangle$ are sufficiently close. From the definition of $\mathcal{O}_T$ and $\mathcal{O}_S$, we have that $\mathcal{O}_T(x) \neq \mathcal{O}_S(x)$ iff $x \in T \setminus S \subset \{0,1\}^n$. By the O2H Lemma (Lemma 4),

$$\begin{aligned} \mathbb{E}_{\substack{T \subseteq \{0,1\}^n, |T|=t \\ S \subseteq T, |S|=s}} \left[\left\| |\Psi_q^S\rangle - |\Psi_q^T\rangle \right\|^2 \right] &\leq 2q \mathbb{E}_{\substack{T \subseteq \{0,1\}^n, |T|=t \\ S \subseteq T, |S|=s}} \sqrt{\frac{1}{q} \sum_{i=0}^{q-1} \left\| \Pi_{T \setminus S} |\Psi_i^S\rangle \right\|^2} \\ &\leq 2q \sqrt{\frac{1}{q} \sum_{i=0}^{q-1} \mathbb{E}_{\substack{T \subseteq \{0,1\}^n, |T|=t \\ S \subseteq T, |S|=s}} \left\| \Pi_{T \setminus S} |\Psi_i^S\rangle \right\|^2} \quad (\text{Jensen's inequality}) \\ &= O\left(q \cdot \sqrt{\frac{t-s}{2^n}}\right). \end{aligned}$$

Therefore, the probability (over the choice of S and T) that $\mathcal{D}^{\mathcal{O}_S}(|S\rangle^{\otimes k})$ succeeds is at most the probability that $\mathcal{D}^{\mathcal{O}_T}(|S\rangle^{\otimes k})$ succeeds – up to an additive loss of $O(q \cdot \sqrt{\frac{t-s}{2^n}})$. ◁

H₄. Output $\mathcal{D}^{\mathcal{O}_T}(|\sigma_X\rangle)$, where the subset X is sampled as follows: first, sample a random subset $T \subseteq \{0,1\}^n$ of size $|T| = t$, and then let $X \subseteq T$ be a random subset of size $|X| = k$.

▷ Claim 11.

$$|p(\mathbf{H}_4) - p(\mathbf{H}_3)| \leq O\left(\frac{k}{\sqrt{s}} + \frac{sk}{t}\right).$$

Proof. Here, we make use of Lemma 6 which says that, for any superset $T \subseteq \{0,1\}^n$ of size $|T| = t$,

$$\text{TD} \left(\mathbb{E}_{\substack{S \subseteq T \\ |S|=s}} [|S\rangle \langle S|^{\otimes k}], \mathbb{E}_{\substack{X \subseteq T \\ |X|=k}} [|\sigma_X\rangle \langle \sigma_X|] \right) \leq O\left(\frac{k}{\sqrt{s}} + \frac{sk}{t}\right). \quad \triangleleft$$

$\mathbf{H}_5$. Output $\mathcal{D}^{\mathcal{O}_X}(|\sigma_X\rangle)$, where the subset X is sampled as follows: first, sample a random subset $T \subseteq \{0, 1\}^n$ of size $|T| = t$, and then let $X \subseteq T$ be a random subset of size $|X| = k$.

▷ Claim 12.

$$|p(\mathbf{H}_5) - p(\mathbf{H}_4)| \leq O\left(q \cdot \sqrt{\frac{t-k}{2^n}}\right).$$

Proof. Suppose that $X \subseteq T$ is a random subset of size $|X| = k$, and let $|\sigma_X\rangle = \frac{1}{k!} \sum_{\sigma \in S_k} |x_{\sigma(1)}, \dots, x_{\sigma(k)}\rangle$, where S_k denotes the symmetric group on $[k]$. Consider the state

$$|\Phi_q^T\rangle = U_q \mathcal{O}_T U_{q-1} \dots U_1 \mathcal{O}_T U_0 |\psi_0\rangle |\sigma_X\rangle.$$

prepared by $\mathcal{D}^{\mathcal{O}_T}(|\sigma_X\rangle)$ just before the measurement. Similarly, we let

$$|\Phi_q^X\rangle = U_q \mathcal{O}_X U_{q-1} \dots U_1 \mathcal{O}_X U_0 |\psi_0\rangle |\sigma_X\rangle.$$

be the state prepared by $\mathcal{A}^{\mathcal{O}_X}(|\sigma_X\rangle)$. Using Lemma 4 as before, we get that

$$\begin{aligned} \mathbb{E}_{\substack{T \subseteq \{0,1\}^n, |T|=t \\ X \subseteq T, |X|=k}} \left\| |\Phi_q^T\rangle - |\Phi_q^X\rangle \right\| &\leq 2q \mathbb{E}_{\substack{T \subseteq \{0,1\}^n, |T|=t \\ X \subseteq T, |X|=k}} \sqrt{\frac{1}{q} \sum_{i=0}^{q-1} \left\| \Pi_{T \setminus X} |\Phi_i^X\rangle \right\|^2} \\ &\leq 2q \sqrt{\frac{1}{q} \sum_{i=0}^{q-1} \mathbb{E}_{\substack{T \subseteq \{0,1\}^n, |T|=t \\ X \subseteq T, |X|=k}} \left\| \Pi_{T \setminus X} |\Phi_i^X\rangle \right\|^2} \quad (\text{Jensen's inequality}) \\ &\leq O\left(q \cdot \sqrt{\frac{t-k}{2^n}}\right). \end{aligned}$$

Therefore, the probability (over the choice of X and T) that $\mathcal{D}^{\mathcal{O}_T}(|\sigma_X\rangle)$ succeeds is at most the probability that $\mathcal{D}^{\mathcal{O}_X}(|\sigma_X\rangle)$ succeeds (up to an additive error of $\frac{q}{\sqrt{t-k}}$). ◁

Therefore, by applying the triangle inequality, we get that

$$|p(\mathbf{H}_1) - p(\mathbf{H}_5)| \leq O\left(q \cdot \sqrt{\frac{t-s}{2^n}} + q \cdot \sqrt{\frac{t-k}{2^n}} + \frac{k}{\sqrt{s}} + \frac{sk}{t}\right).$$

This proves the claim. ◀

5 Multi-Copy Revocable Encryption: Definition

In this section we formally define and construct multi-copy secure revocable encryption schemes. These are regular encryption scheme but where the ciphertexts are associated with quantum states. Moreover, the security property guarantees that any adversary that successfully returns k valid copies of a quantum ciphertext (which it received from a trusted party), where k is an arbitrary polynomial, necessarily loses the ability to decrypt the ciphertexts in the future – even if the secret key is revealed.

Our definition of revocable encryption is as follows:

► **Definition 13** (Revocable Encryption). *Let $\lambda \in \mathbb{N}$ denote the security parameter. A revocable encryption scheme $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Revoke})$ with plaintext space $\mathcal{M}$ consists of the following QPT algorithms:*

- $\text{KeyGen}(1^\lambda)$: on input the security parameter 1^λ , output a secret key sk .
- $\text{Enc}(\text{sk}, m)$: on input the secret key sk and a message $m \in \mathcal{M}$, output a (pure) ciphertext state $|\psi\rangle$ and a (private) verification key vk .
- $\text{Dec}(\text{sk}, \rho)$: on input the secret key sk and a quantum state ρ , output a message m' .
- $\text{Revoke}(\text{sk}, \text{vk}, \sigma)$: on input the secret key sk , a verification key vk and a state σ , output $\top$ or $\perp$.

In addition, we require that Σ satisfies the following two properties:

Correctness of decryption: for all plaintexts $m \in \mathcal{M}$, it holds that

$$\Pr \left[m \leftarrow \text{Dec}(\text{sk}, |\psi\rangle) : \begin{array}{l} \text{sk} \leftarrow \text{KeyGen}(1^\lambda) \\ (|\psi\rangle, \text{vk}) \leftarrow \text{Enc}(\text{sk}, m) \end{array} \right] \geq 1 - \text{negl}(\lambda).$$

Correctness of revocation: for all plaintexts $m \in \mathcal{M}$, it holds that

$$\Pr \left[\top \leftarrow \text{Revoke}(\text{sk}, \text{vk}, |\psi\rangle) : \begin{array}{l} \text{sk} \leftarrow \text{KeyGen}(1^\lambda) \\ (|\psi\rangle, \text{vk}) \leftarrow \text{Enc}(\text{sk}, m) \end{array} \right] \geq 1 - \text{negl}(\lambda).$$

There are two properties we require the above scheme to satisfy.

Firstly, we require the above scheme to be correct. That is, we require the following to hold for all $m \in \{0, 1\}^\ell$,

$$\Pr \left[m \leftarrow \text{Dec}(\text{sk}, |\psi\rangle) : \begin{array}{l} \text{sk} \leftarrow \text{KeyGen}(1^\lambda) \\ |\psi\rangle \leftarrow \text{Enc}(\text{sk}, m) \end{array} \right] \geq 1 - \epsilon(\lambda),$$

for some negligible function $\epsilon(\cdot)$.

Multi-Copy Revocable Security

We use the following notion of security.

► **Definition 14** (Multi-Copy Revocable Security). *Let $\lambda \in \mathbb{N}$ denote the security parameter and let $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Revoke})$ be a revocable encryption scheme with plaintext space $\mathcal{M}$. Consider the following experiment between a QPT adversary $\mathcal{A}$ and a challenger.*

$\text{RevokeExpt}_{\lambda, \Sigma, \mathcal{A}}(b)$:

1. $\mathcal{A}$ submits two messages $m_0, m_1 \in \mathcal{M}$ and a polynomial $k = k(\lambda)$ to the challenger.
2. The challenger samples a key $\text{sk} \leftarrow \text{KeyGen}(1^\lambda)$ and produces $|\psi_b\rangle \leftarrow \text{Enc}(\text{sk}, m_b)$. Afterwards, the challenger sends the quantum state $|\psi_b\rangle^{\otimes k}$ to $\mathcal{A}$.
3. $\mathcal{A}$ returns a quantum state ρ .
4. The challenger performs the measurement $\left\{ |\psi_b\rangle\langle\psi_b|^{\otimes k}, \mathbb{I} - |\psi_b\rangle\langle\psi_b|^{\otimes k} \right\}$ on the returned state ρ . If the measurement succeeds, the game continues; otherwise, the challenger aborts.
5. The challenger sends the secret key sk to $\mathcal{A}$.
6. $\mathcal{A}$ outputs a bit b' .

We say that the revocable encryption scheme $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Revoke})$ has multi-copy revocable security, if the following holds for all $\mu \in \{0, 1, \perp\}$:

$$\left| \Pr[\mu \leftarrow \text{RevokeExpt}_{\lambda, \Sigma, \mathcal{A}}(0)] - \Pr[\mu \leftarrow \text{RevokeExpt}_{\lambda, \Sigma, \mathcal{A}}(1)] \right| \leq \text{negl}(\lambda),$$

► **Remark 15.** In this work, we will work with a weaker variant of this security game, one where the post revocation adversary is given access to an oracle that depends on the key sk instead. The reason for this will become clear from context.

► **Remark 16 (Search variant).** Occasionally, we also consider the search variant of multi-copy revocable encryption. Here, the experiment is similar, except that in Step 1, the adversary only submits k to the challenger. Then, in Step 2, the challenger chooses a message m uniformly at random from the plaintext space, encrypts it k times and sends all of the copies to the adversary. Finally, the adversary is said to win the game if it guesses m correctly.

6 Construction of Multi-Copy Secure Revocable Encryption

In this section, we instantiate our revocable encryption scheme using quantum-secure pseudorandom permutations (QPRPs), and we prove security in an oracular model: this means that, rather than revealing the QPRP key in the final part of the revocable security experiment, we instead allow the adversary to query an oracle for a the permutation instead.

► **Construction 17.** Let $\lambda \in \mathbb{N}$ be the security parameter. Let $n, m \in \mathbb{N}$ be polynomial in λ . Let $\Phi = \{\Phi_\lambda\}_{\lambda \in \mathbb{N}}$ be an ensemble of permutations $\Phi_\lambda = \{\varphi_\kappa : \{0, 1\}^{n+m} \rightarrow \{0, 1\}^{n+m}\}_{\kappa \in \mathcal{K}_\lambda}$, for some set $\mathcal{K}_\lambda$. Consider the scheme $\Sigma^\Phi = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Revoke})$ which consists of the following QPT algorithms:

- **KeyGen**(1^λ): sample a uniformly random key $\kappa \in \mathcal{K}_\lambda$ and let $\text{sk} = \kappa$.
- **Enc**(sk, μ): on input the secret key $\text{sk} = \kappa$ and message $\mu \in \{0, 1\}^m$, sample $y \sim \{0, 1\}^n$ and prepare the subset state given by

$$|S_y\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0, 1\}^n} |\varphi_\kappa(x||y)\rangle.$$

Output the ciphertext state $(|S_y\rangle^{\otimes k}, y \oplus \mu)$ and (private) verification key $\text{vk} = y$.

- **Dec**(sk, ct): on input the decryption key κ and ciphertext state $(|S_y\rangle, z) \leftarrow \text{ct}$, do the following:
 - Coherently compute φ_κ^{-1} on $|S_y\rangle$ and store the answer in a separate output register.
 - Measure the output register to get $x' || y' \in \{0, 1\}^{n+m}$.
 - Output $y' \oplus z$.
- **Revoke**($\text{sk}, \text{vk}, \rho$): on input sk , a state ρ and verification key vk , it parses $\kappa \leftarrow \text{sk}$, $y \leftarrow \text{vk}$ and applies the measurement $\{|S_y\rangle\langle S_y|, \mathbb{I} - |S_y\rangle\langle S_y|\}$ to ρ ; it outputs $\top$ if it succeeds, and $\perp$ otherwise.

Proof of Multi-Copy Revocable Security

► **Theorem 18.** Construction 17, when instantiated with a QPRP $\Phi = \{\Phi_\lambda\}_{\lambda \in \mathbb{N}}$, satisfies (an oracular notion of) multi-copy revocable security.

Proof. Because we are working in the oracular model of revocable security, we can invoke QPRP security and assume that Σ in Construction 17 is instantiated with a perfectly random permutation φ rather than a QPRP permutation φ_κ .

Suppose that our construction does not achieve multi-copy revocable security. Then, there exists an adversary $\mathcal{A}$ such that

$$|\Pr[\mu \leftarrow \text{RevokeExpt}_{\lambda, \Sigma, \mathcal{A}}(0)] - \Pr[\mu \leftarrow \text{RevokeExpt}_{\lambda, \Sigma, \mathcal{A}}(1)]| = \epsilon(\lambda),$$

for some non-negligible function $\epsilon(\cdot)$. For convenience, we model $\mathcal{A}$ as a pair of quantum algorithms $(\mathcal{A}_0, \mathcal{A}_1)$, where $\mathcal{A}_0$ corresponds to the pre-revocation adversary, and $\mathcal{A}_1$ corresponds to the post-revocation adversary. We consider the following sequence of hybrid distributions.

$\mathbf{H}_1^b$. This corresponds to $\text{RevokeExpt}^{\mathcal{A}}(1^\lambda, b)$.

1. $\mathcal{A}_0$ submits two m -bit messages (μ_0, μ_1) and a polynomial $k = k(\lambda)$ to the challenger.
2. The challenger samples $y \sim \{0, 1\}^m$ and produces a quantum state

$$|S_y\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0, 1\}^n} |\varphi(x||y)\rangle.$$

The challenger then sends $|S_y\rangle^{\otimes k}$ and $y \oplus \mu_b$ to $\mathcal{A}_0$.

3. $\mathcal{A}_0$ prepares a bipartite state on registers R and AUX , and sends R to the challenger and AUX to $\mathcal{A}_1$.
4. The challenger performs the projective measurement $\{|S_y\rangle\langle S_y|^{\otimes k}, \mathbb{I} - |S_y\rangle\langle S_y|^{\otimes k}\}$ on R . If the measurement succeeds, the challenger outputs $\perp$. Otherwise, the challenger continues.
5. The challenger grants $\mathcal{A}_1$ quantum oracle access to φ^{-1} .
6. $\mathcal{A}_1$ outputs a bit b' .

$\mathbf{H}_2^b$. This is the same experiment as in $\mathbf{H}_1^b$, except that we change how $|S_y\rangle^{\otimes k}$ is generated before revocation:

- The challenger samples a random subset $S \subseteq \{0, 1\}^{n+m}$ of size $|S| = 2^n$.
- The challenger samples a random $y \sim \{0, 1\}^m$.
- The challenger sends $|S\rangle^{\otimes k}$ and $y \oplus \mu_b$ to $\mathcal{A}_0$.

Later, the challenger samples a random permutation $\pi : \{0, 1\}^{n+m} \rightarrow \{0, 1\}^{n+m}$ subject to the constraint that $\pi(s) = *||y$, for all $s \in S$. In other words, $\pi(S) = T_y$, where

$$T_y := \{x \in \{0, 1\}^{n+m} : x = (*||y)\}$$

and $|S| = |T_y| = 2^n$. After revocation, $\mathcal{A}_1$ receives oracle access to π .

▷ **Claim 19.** $\mathbf{H}_1^b$ and $\mathbf{H}_2^b$ are identically distributed.

Proof. This follows immediately. We just changed the order in which we sample things. ◁

$\mathbf{H}_3^b$. This is the same experiment as in $\mathbf{H}_2^b$, except that we change the second part of the experiment: after the challenger sends $|S\rangle^{\otimes k}$ and $y \oplus \mu_b$ to $\mathcal{A}_0$, he does the following:

- The challenger samples a random function $g : S \rightarrow T_y$.⁵
- The challenger samples a random permutation $\omega : (\{0, 1\}^{n+m} \setminus S) \rightarrow (\{0, 1\}^{n+m} \setminus T_y)$.

After revocation, $\mathcal{A}_1$ receives oracle access to $f : \{0, 1\}^{n+m} \rightarrow \{0, 1\}^{n+m}$, where

$$f(x) = \begin{cases} g(x), & \text{if } x \in S \\ \omega(x), & \text{if } x \notin S. \end{cases}$$

⁵ Note that we have $g(S) \subseteq T_y$ in general.

▷ Claim 20. $\mathbf{H}_2^b$ and $\mathbf{H}_3^b$ are $O(q^3/2^n)$ -close whenever $\mathcal{A}$ makes q queries.

Proof. Here, we apply Zhandry's result which says that random functions are indistinguishable from random permutations. Consider the following algorithm $\mathcal{B}$ which receives oracle access to O , which is either a random function $F : \{0, 1\}^n \rightarrow \{0, 1\}^n$ or a random permutation $P : \{0, 1\}^n \rightarrow \{0, 1\}^n$:

1. $\mathcal{B}$ samples a random subset $S \subseteq \{0, 1\}^{n+m}$ of size $|S| = 2^n$ and a random $y \sim \{0, 1\}^m$.
2. $\mathcal{B}$ sends $|S_y\rangle^{\otimes k}$ and $y \oplus \mu_b$ to $\mathcal{A}_0$.
3. When $\mathcal{A}_0$ replies with a bipartite state on registers R and AUX , $\mathcal{B}$ performs the projective measurement $\left\{ |S\rangle\langle S|^{\otimes k}, \mathbb{I} - |S\rangle\langle S|^{\otimes k} \right\}$ on R . If it succeeds, $\mathcal{B}$ outputs $\perp$. Otherwise, $\mathcal{B}$ continues.
4. $\mathcal{B}$ runs the post-revocation adversary $\mathcal{A}_1$ on input AUX . Whenever $\mathcal{A}_1$ makes a query, $\mathcal{B}$ answers using the function

$$f(x) = \begin{cases} (\tau \circ O \circ \sigma)(x), & \text{if } x \in S \\ \omega(x), & \text{if } x \notin S \end{cases}$$

where we let

- σ be some canonical mapping from S (with $|S| = 2^n$) onto $\{0, 1\}^n$, i.e., σ is a function which assigns each element of $S \subseteq \{0, 1\}^{n+m}$ a unique bit string in $\{0, 1\}^n$.
- τ be the function which maps each $x \in \{0, 1\}^n$ to $(x||y) \in \{0, 1\}^{n+m}$.

Note that whenever O is a random permutation, the view of $\mathcal{A}$ is precisely $\mathbf{H}_3^b$; whereas, if O is a random function, the view of $\mathcal{A}$ is precisely $\mathbf{H}_4^b$. Therefore, the claim follows from Zhandry's result on indistinguishability of random permutations from random functions (see full version for formal statement). $\triangleleft$

$\mathbf{H}_4^b$. This is the same experiment as in $\mathbf{H}_3^b$, except that we change the second part of the experiment once again: after the challenger sends $|S_y\rangle^{\otimes k}$ and $y \oplus \mu_b$ to $\mathcal{A}_0$, he does the following:

- the challenger samples a random function $f : \{0, 1\}^{n+m} \rightarrow \{0, 1\}^{n+m}$ subject to the constraint that $f(s) = *||y$, for all $s \in S$.

After revocation, $\mathcal{A}_1$ receives oracle access to f .

▷ Claim 21. $\mathbf{H}_3^b$ and $\mathbf{H}_4^b$ are $O(q^3/(2^{n+m} - 2^n))$ -close whenever $\mathcal{A}$ makes q queries.

Proof. The proof again follows since distinguishing $\mathbf{H}_3^b$ and $\mathbf{H}_4^b$ amounts to distinguishing a random function from a random permutation mapping a random permutation $\omega : (\{0, 1\}^{n+m} \setminus S) \rightarrow (\{0, 1\}^{n+m} \setminus T_y)$. Since the domain and co-domain are of equal size $2^{n+m} - 2^n$, the advantage is at most $O(q^3/(2^{n+m} - 2^n))$. $\triangleleft$

$\mathbf{H}_5^b$. This is the same experiment as in $\mathbf{H}_4^b$, but now we change what $\mathcal{A}_0$ receives in the pre-revocation phase:

- The challenger samples a random subset $S \subseteq \{0, 1\}^{n+m}$ of size $|S| = 2^n$.
- The challenger samples a random $y \sim \{0, 1\}^m$.
- The challenger sends $(|S_y\rangle^{\otimes k}, y)$ to $\mathcal{A}_0$.

After revocation, the challenger samples a random function $f : \{0, 1\}^{n+m} \rightarrow \{0, 1\}^{n+m}$ subject to the constraint that $f(s) = *||y \oplus \mu_b$, for all $s \in S$. $\mathcal{A}$ receives oracle access to f .

▷ Claim 22. $\mathbf{H}_4^b$ and $\mathbf{H}_5^b$ are identical.

Proof. This follows from the fact that we have just re-labeled the variables. $\triangleleft$

$\mathbf{H}_6^b$. This is the same experiment as in $\mathbf{H}_5^b$, except that we once again change what $\mathcal{A}_0$ receives in the pre-revocation phase:

- The challenger samples a random subset $S \subseteq \{0, 1\}^{n+m}$ of size $|S| = 2^n$.
- The challenger samples a random $y \sim \{0, 1\}^m$.
- The challenger samples a random $u \sim \{0, 1\}^m$.
- The challenger sends $(|S_y\rangle^{\otimes k}, y)$ to $\mathcal{A}_0$.

After revocation, the challenger samples a random function $f : \{0, 1\}^{n+m} \rightarrow \{0, 1\}^{n+m}$ subject to the constraint that $f(s) = *||u$, for all $s \in S$. $\mathcal{A}$ receives oracle access to f .

Note that hybrids $\mathbf{H}_6^0$ and $\mathbf{H}_6^1$ are identically distributed. By assumption, we also know that $\mathcal{A}$ distinguishes $\mathbf{H}_1^0$ and $\mathbf{H}_1^1$ with non-negligible advantage. Therefore, our previous hybrid argument has shown that $\mathcal{A}$ must also distinguish $\mathbf{H}_5^0$ and $\mathbf{H}_6^0$ (respectively, hybrids $\mathbf{H}_5^1$ and $\mathbf{H}_6^1$) with advantage at least $\epsilon'(\lambda)$, for some non negligible function $\epsilon'(\lambda)$. To complete the proof, we show the following claim which yields the desired contradiction to the $k \mapsto k + 1$ unforgeability property of subset states from Theorem 7.

▷ **Claim 23.** *Forge* in Algorithm 1 is a $\text{poly}(\lambda)$ -query algorithm (which internally runs $\mathcal{A}$) such that

$$\Pr_{\substack{S \subseteq \{0,1\}^{n+m} \\ |S|=2^n}} \left[(x_1, \dots, x_{k+1}) \in \text{dist}(S, k+1) : (x_1, \dots, x_{k+1}) \leftarrow \text{Forge}^{\mathcal{O}_S}(|S\rangle^{\otimes k}) \right] \geq 1/\text{poly}(\lambda).$$

Proof. Since $\mathbf{H}_5^b$ and $\mathbf{H}_6^b$ can be distinguished by the adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ with advantage $\epsilon'(\lambda)$, for some non-negligible function $\epsilon'(\lambda)$, this implies that the post-revocation adversary $\mathcal{A}_1$ is an algorithm for which the following property holds: given as input uniformly random string y and an auxiliary register (conditioned on revocation succeeding), $\mathcal{A}_1$ can distinguish whether it is given an oracle for a function $H : \{0, 1\}^{n+m} \rightarrow \{0, 1\}^{n+m}$ which is random subject to the constraint that $H(s) = *||y \oplus \mu_b$ for all $s \in S$, or whether it is given an oracle for a function $G : \{0, 1\}^{n+m} \rightarrow \{0, 1\}^{n+m}$ which is a random function subject to the constraint that $G(s) = *||u$ for all $s \in S$. Crucially, the two functions differ precisely on inputs belonging to S , and are otherwise identical.

Consider the quantum extractor $\text{Ext}^G(y, \text{AUX})$ which is defined as follows:

1. Sample $i \leftarrow [q]$, where q denotes the total number of queries made by $\mathcal{A}$.
2. Run $\mathcal{A}_1^G(y, \text{AUX})$ just before the $(i - 1)$ -st query to G .
3. Measure $\mathcal{A}_1$'s i -th query in the computational basis, and output the measurement outcome.

From the O2H Lemma (see full version for the formal statement), we get that

$$\left| \Pr \left[\mathcal{A}_1^H(y, \text{AUX}_{|\top}) = 1 : \begin{array}{l} S \subseteq \{0,1\}^{n+m}, |S|=2^n \\ y \sim \{0,1\}^m \\ (\mathbf{R}, \text{AUX}) \leftarrow \mathcal{A}_0(|S\rangle^{\otimes k}, y) \\ H \text{ s.t. } H(s) = *||y \oplus \mu_b, \forall s \in S \end{array} \right] - \Pr \left[\mathcal{A}_1^G(y, \text{AUX}_{|\top}) = 1 : \begin{array}{l} S \subseteq \{0,1\}^{n+m}, |S|=2^n \\ y \sim \{0,1\}^m, u \sim \{0,1\}^m \\ (\mathbf{R}, \text{AUX}) \leftarrow \mathcal{A}_0(|S\rangle^{\otimes k}, y) \\ G \text{ s.t. } G(s) = *||u, \forall s \in S \end{array} \right] \right| \\ \leq 2q \sqrt{\Pr \left[\text{Ext}^G(y, \text{AUX}_{|\top}) \in S : \begin{array}{l} S \subseteq \{0,1\}^{n+m}, |S|=2^n \\ y \sim \{0,1\}^m, u \sim \{0,1\}^m \\ (\mathbf{R}, \text{AUX}) \leftarrow \mathcal{A}_0(|S\rangle^{\otimes k}, y) \\ G \text{ s.t. } G(s) = *||u, \forall s \in S \end{array} \right]},$$

where $\text{AUX}_{|\top}$ corresponds to the register AUX conditioned on the event that the projective measurement

$$\left\{ |S\rangle\langle S|^{\otimes k}, \mathbb{I} - |S\rangle\langle S|^{\otimes k} \right\}$$

succeeds on register $\mathbf{R}$. Because the distinguishing advantage of the adversary $\mathcal{A}_1$ is non-negligible (conditioned on the event that revocation succeeds on register $\mathbf{R}$) and $q = \text{poly}(\lambda)$, we get

$$\Pr \left[\text{Ext}^G(y, \text{AUX}_{|\top}) \in S : \begin{array}{l} S \subset \{0,1\}^{n+m}, |S|=2^n \\ y \sim \{0,1\}^m, u \sim \{0,1\}^m \\ (\text{R}, \text{AUX}) \leftarrow \mathcal{A}_0(|S|^{\otimes k}, y) \\ G \text{ s.t. } G(s) = *||u, \forall s \in S \end{array} \right] \geq 1/\text{poly}(\lambda).$$

To complete the proof, we now show that $\text{Forge}^{\mathcal{O}_S}(|S|^{\otimes k})$ in Algorithm 1 is a successful algorithm against the $k \rightarrow k+1$ unforgeability of subset states.

■ **Algorithm 1** $\text{Forge}^{\mathcal{O}_S}(|S|^{\otimes k})$.

Input: $|S|^{\otimes k}$ and a membership oracle $\mathcal{O}_S$, where $S \subset \{0,1\}^{n+m}$ is a subset.

Output: $x_1, \dots, x_{k+1} \in \{0,1\}^{n+m}$.

- 1 Sample uniformly random strings $y, u \sim \{0,1\}^m$;
- 2 Run $(\text{R}, \text{AUX}) \leftarrow \mathcal{A}_0(|S|^{\otimes k}, y)$;
- 3 Measure R in the computational basis to obtain $x_1, \dots, x_k$;
- 4 Run the quantum extractor $\text{Ext}^G(y, \text{AUX})$ to obtain an element x_{k+1} , where the oracle G can be simulated via $\mathcal{O}_S$ as follows: on input $x \in \{0,1\}^{n+m}$, we let

$$G(x) = \begin{cases} g_1(x)||u, & \text{if } \mathcal{O}_S(x) = 1 \\ g_2(x), & \text{otherwise} \end{cases}$$

where $g_1 : \{0,1\}^{n+m} \rightarrow \{0,1\}^n$ and $g_2 : \{0,1\}^{n+m} \rightarrow \{0,1\}^{n+m}$ are uniformly random functions.

- 5 Output $(x_1, \dots, x_{k+1})$.
-

Let $\text{Revoke}(S, k, \text{R})$ denote the projective measurement $\{|S\rangle\langle S|^{\otimes k}, \mathbb{I} - |S\rangle\langle S|^{\otimes k}\}$ of register R . Using the Simultaneous Distinct Extraction Lemma (Lemma 24), we get that

$$\begin{aligned} & \Pr_{\substack{S \subset \{0,1\}^{n+m} \\ |S|=2^n}} \left[(x_1, \dots, x_{k+1}) \in \text{dist}(S, k+1) : (x_1, \dots, x_{k+1}) \leftarrow \text{Forge}^{\mathcal{O}_S}(|S|^{\otimes k}) \right] \\ & \geq \left(1 - O\left(\frac{k^2}{2^n}\right) \right) \cdot \Pr \left[\text{Revoke}(S, k, \text{R}) = \top : \begin{array}{l} S \subset \{0,1\}^{n+m}, |S|=2^n \\ y \sim \{0,1\}^m \\ (\text{R}, \text{AUX}) \leftarrow \mathcal{A}_0(|S|^{\otimes k}, y) \end{array} \right] \\ & \quad \cdot \Pr \left[\text{Ext}^G(y, \text{AUX}_{|\top}) \in S : \begin{array}{l} S \subset \{0,1\}^{n+m}, |S|=2^n \\ y \sim \{0,1\}^m, u \sim \{0,1\}^m \\ (\text{R}, \text{AUX}) \leftarrow \mathcal{A}_0(|S|^{\otimes k}, y) \\ G \text{ s.t. } G(s) = *||u, \forall s \in S \end{array} \right] \end{aligned}$$

which is at least inverse polynomial in λ . This proves the claim. $\triangleleft$

Thus, we obtain the desired contradiction to the $k \mapsto k+1$ unforgeability property of subset states from Theorem 7. This completes the proof of multi-copy revocable encryption security. $\blacktriangleleft$

6.1 Simultaneous Distinct Extraction Lemma

The following lemma allows us to analyze the probability of simultaneously extracting $k+1$ distinct subset elements in some subset $S \subseteq \{0,1\}^n$ in terms of the success probability of revocation (i.e., the projection onto $|S\rangle\langle S|^{\otimes k}$) and the success probability of extracting another subset element from the adversary's state.

► **Lemma 24** (Simultaneous Distinct Extraction). *Let $n, k \in \mathbb{N}$ and let $\rho \in \mathcal{D}(\mathcal{H}_X \otimes \mathcal{H}_Y)$ be an any density matrix, where $\mathcal{H}_X$ is an $n \cdot k$ -qubit Hilbert space and where $\mathcal{H}_Y$ is an arbitrary Hilbert space. Let $|S\rangle$ denote a subset state, for some subset $S \subseteq \{0, 1\}^n$, and let $\mathcal{E} : \mathcal{L}(\mathcal{H}_Y) \rightarrow \mathcal{L}(\mathcal{H}_{X'})$ be any CPTP map of the form*

$$\mathcal{E}_{Y \rightarrow X'}(\sigma) = \text{Tr}_E \left[V_{Y \rightarrow X'E} \sigma V_{Y \rightarrow X'E}^\dagger \right], \quad \forall \sigma \in \mathcal{D}(\mathcal{H}_Y),$$

for some isometry $V_{Y \rightarrow X'E}$ and n -qubit Hilbert space $\mathcal{H}_{X'}$. Consider the POVM element

$$\Lambda = \sum_{\substack{s_1, \dots, s_{k+1} \in S \\ (s_1, \dots, s_{k+1}) \in \text{dist}(S, k+1)}} |s_1, \dots, s_k\rangle\langle s_1, \dots, s_k|_X \otimes V_{Y \rightarrow X'E}^\dagger (|s_{k+1}\rangle\langle s_{k+1}|_{X'} \otimes I_E) V_{Y \rightarrow X'E}.$$

Let $\rho_X = \text{Tr}_Y[\rho_{XY}]$ denote the reduced state. Then, it holds that

$$\text{Tr}[\Lambda \rho] \geq \left(1 - O\left(\frac{k^2}{|S|}\right)\right) \cdot \text{Tr}[|S\rangle\langle S|^{\otimes k} \rho_X] \cdot \text{Tr}[|S\rangle\langle S| \mathcal{E}_{Y \rightarrow X'}(\sigma)],$$

where $\sigma = \text{Tr}\left[(|S\rangle\langle S|^{\otimes k} \otimes I)\rho\right]^{-1} \cdot \text{Tr}_X[(|S\rangle\langle S|^{\otimes k} \otimes I)\rho]$ is a reduced state in system Y .

7 Multi-Copy Revocable Programs: Definition

In this section, we study whether arbitrary functionalities can be revoked. We define and study revocable programs with multi-copy security. In this notion, there is a functionality preserving compiler that takes a program and converts it into a quantum state, which can later be certifiably revoked.

We now give a formal definition of revocable programs.

► **Definition 25** (Revocable Program). *Let $\mathcal{P} = \bigcup_{\lambda \in \mathbb{N}} \mathcal{P}_\lambda$ be a class of efficiently computable program families $\mathcal{P}_\lambda = \{P : \mathcal{X}_\lambda \rightarrow \mathcal{Y}_\lambda\}$ with domain $\mathcal{X}_\lambda$ and range $\mathcal{Y}_\lambda$. A revocable program compiler for the class $\mathcal{P}$ is a tuple $\Sigma = (\text{Compile}, \text{Eval}, \text{Revoke})$ consisting of the following QPT algorithms:*

- **Compile**($1^\lambda, P$): on input the security parameter 1^λ and a program $P \in \mathcal{P}_\lambda$ with $P : \mathcal{X}_\lambda \rightarrow \mathcal{Y}_\lambda$, output a quantum state $|\Psi_P\rangle$ and a (private) verification key vk .
- **Eval**($|\Psi_P\rangle, x$): on input a quantum state $|\Psi_P\rangle$ and input $x \in \mathcal{X}_\lambda$, output $P(x)$.
- **Revoke**(vk, σ): on input the verification key vk and a state σ , output $\top$ or $\perp$.

In addition, we require that Σ satisfies the following two properties for all $\lambda \in \mathbb{N}$:

Correctness of evaluation: for all programs $P \in \mathcal{P}_\lambda$ and inputs $x \in \mathcal{X}_\lambda$, it holds that

$$\Pr[P(x) \leftarrow \text{Eval}(|\Psi_P\rangle, x) : (|\Psi_P\rangle, \text{vk}) \leftarrow \text{Compile}(1^\lambda, P)] \geq 1 - \text{negl}(\lambda).$$

Correctness of revocation: for all programs $P \in \mathcal{P}_\lambda$, it holds that

$$\Pr[\top \leftarrow \text{Revoke}(\text{vk}, (|\Psi_P\rangle)) : (|\Psi_P\rangle, \text{vk}) \leftarrow \text{Compile}(1^\lambda, P)] \geq 1 - \text{negl}(\lambda).$$

Multi-Copy Revocable Security

We use the following notion of security.

► **Definition 26** (Multi-Copy Revocable Security for Programs). Let $\mathcal{P} = \bigcup_{\lambda \in \mathbb{N}} \mathcal{P}_\lambda$ be a class of efficiently computable program families $\mathcal{P}_\lambda = \{P : \mathcal{X}_\lambda \rightarrow \mathcal{Y}_\lambda\}$ and let $\mathcal{D}_{\mathcal{P}} = \bigcup_{\lambda \in \mathbb{N}} \mathcal{D}_{\mathcal{P}_\lambda}$ be an ensemble of program distributions. Let $\mathcal{D}_{\mathcal{X}} = \bigcup_{\lambda \in \mathbb{N}} \mathcal{D}_{\mathcal{X}_\lambda}$ be an ensemble of challenge distribution families with $\mathcal{D}_{\mathcal{X}_\lambda} = \{\mathcal{D}_{\mathcal{X}_\lambda}(P)\}_{P \in \mathcal{P}_\lambda}$. Consider the following experiment between a QPT adversary $\mathcal{A}$ and a challenger.

$\text{RevokeExpt}_{\lambda, \Sigma, \mathcal{A}}^{\mathcal{D}_{\mathcal{P}}, \mathcal{D}_{\mathcal{X}}}$:

1. $\mathcal{A}$ submits a polynomial $k = k(\lambda)$ to the challenger.
2. The challenger samples a program $P \sim \mathcal{D}_{\mathcal{P}_\lambda}$ with domain $\mathcal{X}_\lambda$ and range $\mathcal{Y}_\lambda$, and runs $\text{Compile}(1^\lambda, P)$ to generate a pair $(|\Psi_P\rangle, \text{vk})$. Afterwards, the challenger sends the quantum state $|\Psi_P\rangle^{\otimes k}$ to $\mathcal{A}$.
3. $\mathcal{A}$ returns a quantum state ρ .
4. The challenger performs the measurement $\left\{ |\Psi_P\rangle\langle\Psi_P|^{\otimes k}, \mathbb{I} - |\Psi_P\rangle\langle\Psi_P|^{\otimes k} \right\}$ on the returned state ρ . If the measurement succeeds, the game continues; otherwise, the challenger aborts.
5. The challenger samples a challenge input $x \sim \mathcal{D}_{\mathcal{X}_\lambda}(P)$ sends x to $\mathcal{A}$.
6. The adversary outputs $y \in \mathcal{Y}_\lambda$.
7. The challenger outputs 1 if and only if $P(x) = y$.

We say that a revocable program compiler $\Sigma = (\text{Compile}, \text{Eval}, \text{Revoke})$ has multi-copy revocable security for the ensembles $\mathcal{D}_{\mathcal{P}}$ and $\mathcal{D}_{\mathcal{X}}$, if the following holds for any QPT adversary $\mathcal{A}$:

$$\Pr \left[1 \leftarrow \text{RevokeExpt}_{\lambda, \Sigma, \mathcal{A}}^{\mathcal{D}_{\mathcal{P}}, \mathcal{D}_{\mathcal{X}}} \right] \leq p_{\text{triv}}^{\mathcal{D}_{\mathcal{P}}, \mathcal{D}_{\mathcal{X}}}(\lambda) + \text{negl}(\lambda),$$

where $p_{\text{triv}}^{\mathcal{D}_{\mathcal{P}}, \mathcal{D}_{\mathcal{X}}}(\lambda) = \sup_{\mathcal{A}} \{\Pr[P(x) \leftarrow \mathcal{A}(x) : x \leftarrow \mathcal{D}_{\mathcal{X}_\lambda}(P)]\}$ is the trivial guessing probability.

8 Construction of Multi-Copy Secure Revocable Programs in a Classical Oracle Model

In this section, we give a construction of multi-copy secure revocable programs; specifically, we work with a classical oracle model. Our construction is as follows:

► **Construction 27.** Let $\lambda \in \mathbb{N}$ be the security parameter. Let $n, m \in \mathbb{N}$ be polynomial in λ . Let $\Phi = \{\Phi_\lambda\}_{\lambda \in \mathbb{N}}$ be an ensemble of permutations $\Phi_\lambda = \{\varphi_\kappa : \{0, 1\}^{n+m} \rightarrow \{0, 1\}^{n+m}\}_{\kappa \in \mathcal{K}_\lambda}$, for some set $\mathcal{K}_\lambda$.

- **Setup**(1^λ): sample a uniformly random key $\kappa \in \mathcal{K}_\lambda$ and let $\text{vk} = \kappa$.
- **Compile**($1^\lambda, P$): on input 1^λ and a program $P \in \mathcal{P}_\lambda$ with $P : \mathcal{X}_\lambda \rightarrow \mathcal{Y}_\lambda$, do the following:
 - Sample $y \sim \{0, 1\}^m$ and prepare the subset state given by

$$|S_y\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0, 1\}^n} |\varphi_\kappa(x||y)\rangle.$$

- Let $|\Psi_P\rangle = |S_y\rangle$ and, for brevity, let $S = \{\varphi_\kappa(x||y) : x \in \{0, 1\}^n\}$ be the corresponding subset.
- Let $O = O_{P, S}$ denote an a (public) classical oracle, which is defined as follows:

$$O_{P, S}(x, s) = \begin{cases} P(x), & \text{if } s \in S \\ 0, & \text{otherwise} \end{cases}$$

- $\text{Eval}^O(|\Psi_P\rangle, x)$: on input $|\Psi_P\rangle$, $x \in \mathcal{X}_\lambda$, do the following:
 - Coherently evaluate $O_{P,S}$ on input $|x\rangle \otimes |\Psi_P\rangle$, and compute its output into an ancillary register.
 - Measure the ancillary register and then output the measurement outcome.
- $\text{Revoke}(\text{vk}, \rho)$: on input vk , a state ρ and verification key vk , it parses $y \leftarrow \text{vk}$ and applies the measurement $\{|S_y\rangle\langle S_y|, \mathbb{I} - |S_y\rangle\langle S_y|\}$ to ρ ; it outputs $\top$ if it succeeds, and $\perp$ otherwise.

The above scheme is easily seen to satisfy correctness.

Proof of Multi-Copy Revocable Security

Before we analyze the security of Construction 27, let us first remark that we can instantiate the scheme using a QPRP family $\Phi_\lambda = \{\varphi_\kappa : \{0, 1\}^{n+m} \rightarrow \{0, 1\}^{n+m}\}_{\kappa \in \mathcal{K}_\lambda}$, for some key space $\mathcal{K}_\lambda$. In the security proof, however, we will work with the random permutation model instead. This means that we will consider random permutations throughout the security game.

► **Theorem 28.** *Construction 27 satisfies multi-copy revocable security for any pair of distributions $\mathcal{D}_\mathcal{P}, \mathcal{D}_\mathcal{X}$ in a classical oracle model, where the recipient receives an (ideal classical) oracle for the purpose of evaluation.*

Proof. Please see full version. ◀

9 Construction of Multi-Copy Secure Revocable Point Functions in the QROM

Please see the full version for this section.

10 Applications to Sponge Hashing

Please see the full version for this section.

References

- 1 Scott Aaronson. Quantum copy-protection and quantum money. In *2009 24th Annual IEEE Conference on Computational Complexity*, pages 229–242. IEEE, 2009. doi:10.1109/CCC.2009.42.
- 2 Scott Aaronson, Adam Bouland, Bill Fefferman, Soumik Ghosh, Umesh Vazirani, Chenyi Zhang, and Zixin Zhou. Quantum Pseudoentanglement. In Venkatesan Guruswami, editor, *15th Innovations in Theoretical Computer Science Conference (ITCS 2024)*, volume 287 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:21, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITCS.2024.2.
- 3 Scott Aaronson and Paul Christiano. Quantum money from hidden subspaces. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 41–60, 2012. doi:10.1145/2213977.2213983.
- 4 Scott Aaronson, Jiahui Liu, Qipeng Liu, Mark Zhandry, and Ruizhe Zhang. New approaches for quantum copy-protection. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part I 41*, pages 526–555. Springer, 2021. doi:10.1007/978-3-030-84242-0_19.

- 5 Shweta Agrawal, Fuyuki Kitagawa, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Public key encryption with secure key leasing. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 581–610. Springer, 2023. doi:10.1007/978-3-031-30545-0_20.
- 6 Andris Ambainis, Mike Hamburg, and Dominique Unruh. Quantum security proofs using semi-classical oracles. Cryptology ePrint Archive, Paper 2018/904, 2018. URL: <https://eprint.iacr.org/2018/904>.
- 7 Prabhanjan Ananth, Aditya Gulati, Luowen Qian, and Henry Yuen. Pseudorandom (function-like) quantum state generators: New definitions and applications. In *Theory of Cryptography Conference*, pages 237–265. Springer, 2022. doi:10.1007/978-3-031-22318-1_9.
- 8 Prabhanjan Ananth, Zihan Hu, and Zikuan Huang. Quantum key-revocable dual-regev encryption, revisited. *Cryptology ePrint Archive*, 2024.
- 9 Prabhanjan Ananth and Fatih Kaleoglu. A note on copy-protection from random oracles. *arXiv preprint*, 2022. arXiv:2208.12884.
- 10 Prabhanjan Ananth, Fatih Kaleoglu, and Qipeng Liu. Cloning games: A general framework for unclonable primitives. *arXiv preprint*, 2023. arXiv:2302.01874.
- 11 Prabhanjan Ananth and Rolando L La Placa. Secure software leasing. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 501–530. Springer, 2021. doi:10.1007/978-3-030-77886-6_17.
- 12 Prabhanjan Ananth, Yao-Ting Lin, and Henry Yuen. Pseudorandom strings from pseudorandom quantum states. In *ITCS*, 2024.
- 13 Prabhanjan Ananth, Alexander Poremba, and Vinod Vaikuntanathan. Revocable cryptography from learning with errors. In *Theory of Cryptography Conference*, pages 93–122. Springer, 2023. doi:10.1007/978-3-031-48624-1_4.
- 14 Prabhanjan Ananth, Luowen Qian, and Henry Yuen. Cryptography from pseudorandom quantum states. In *Annual International Cryptology Conference*, pages 208–236. Springer, 2022. doi:10.1007/978-3-031-15802-5_8.
- 15 James Bartusek and Dakshita Khurana. Cryptography with certified deletion. In *Advances in Cryptology – CRYPTO 2023: 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20–24, 2023, Proceedings, Part V*, pages 192–223, Berlin, Heidelberg, 2023. Springer-Verlag. doi:10.1007/978-3-031-38554-4_7.
- 16 James Bartusek, Dakshita Khurana, Giulio Malavolta, Alexander Poremba, and Michael Walter. Weakening assumptions for publicly-verifiable deletion. Cryptology ePrint Archive, Paper 2023/559, 2023. URL: <https://eprint.iacr.org/2023/559>.
- 17 James Bartusek, Dakshita Khurana, and Alexander Poremba. Publicly-verifiable deletion via target-collapsing functions. In *Advances in Cryptology – CRYPTO 2023: 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20–24, 2023, Proceedings, Part V*, pages 99–128, Berlin, Heidelberg, 2023. Springer-Verlag. doi:10.1007/978-3-031-38554-4_4.
- 18 Amit Behera, Zvika Brakerski, Or Sattath, and Omri Shmueli. Pseudorandomness with proof of destruction and applications. In *Theory of Cryptography Conference*, pages 125–154. Springer, 2023. doi:10.1007/978-3-031-48624-1_5.
- 19 G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Cryptographic sponge functions. Submission to NIST (Round 3), 2011. URL: <http://sponge.noekeon.org/CSF-0.1.pdf>.
- 20 G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. The keccak sha-3 submission. Submission to NIST (Round 3), 2011. URL: <http://keccak.noekeon.org/Keccak-submission-3.pdf>.
- 21 John Bostanci, Yuval Efron, Tony Metger, Alexander Poremba, Luowen Qian, and Henry Yuen. Unitary complexity and the uhlmann transformation problem, 2023. doi:10.48550/arXiv.2306.13073.

- 22 John Bostanci, Jonas Haferkamp, Dominik Hangleiter, and Alexander Poremba. Efficient quantum pseudorandomness from hamiltonian phase states, 2024. doi:10.48550/arXiv.2410.08073.
- 23 Zvika Brakerski and Omri Shmueli. (pseudo) random quantum states with binary phase. In *Theory of Cryptography Conference*, pages 229–250. Springer, 2019. doi:10.1007/978-3-030-36030-6_10.
- 24 Anne Broadbent and Rabib Islam. Quantum encryption with certified deletion. In *Theory of Cryptography: 18th International Conference, TCC 2020, Durham, NC, USA, November 16–19, 2020, Proceedings, Part III* 18, pages 92–122. Springer, 2020. doi:10.1007/978-3-030-64381-2_4.
- 25 Anne Broadbent, Stacey Jeffery, Sébastien Lord, Supartha Podder, and Aarthi Sundaram. Secure software leasing without assumptions. In *Theory of Cryptography Conference*, pages 90–120. Springer, 2021. doi:10.1007/978-3-030-90459-3_4.
- 26 Anne Broadbent and Sébastien Lord. Uncloneable quantum encryption via oracles. *arXiv preprint*, 2019. arXiv:1903.00130.
- 27 Alper Çakan and Vipul Goyal. Unclonable cryptography with unbounded collusions. *Cryptology ePrint Archive*, 2023.
- 28 Orestis Chardouvelis, Vipul Goyal, Aayush Jain, and Jiahui Liu. Quantum key leasing for pke and fhe with a classical lessor. *arXiv preprint*, 2023. arXiv:2310.14328.
- 29 Google Cloud. Key rotation, 2024. URL: <https://cloud.google.com/kms/docs/key-rotation>.
- 30 Andrea Coladangelo, Jiahui Liu, Qipeng Liu, and Mark Zhandry. Hidden cosets and applications to unclonable cryptography. In *Advances in Cryptology – CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part I*, pages 556–584, Berlin, Heidelberg, 2021. Springer-Verlag. doi:10.1007/978-3-030-84242-0_20.
- 31 Andrea Coladangelo, Christian Majenz, and Alexander Poremba. Quantum copy-protection of compute-and-compare programs in the quantum random oracle model. *Quantum*, 8:1330, May 2024. doi:10.22331/q-2024-05-02-1330.
- 32 DGBJ Dieks. Communication by epr devices. *Physics Letters A*, 92(6):271–272, 1982.
- 33 Tudor Giurgica-Tiron and Adam Boulund. Pseudorandomness from subset states. *arXiv preprint*, 2023. doi:10.48550/arXiv.2312.09206.
- 34 Fernando Granha Jeronimo, Nir Magrafta, and Pei Wu. Subset states and pseudorandom states. *arXiv preprint*, 2023. doi:10.48550/arXiv.2312.15285.
- 35 Zhengfeng Ji, Yi-Kai Liu, and Fang Song. Pseudorandom quantum states. In *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part III* 38, pages 126–152. Springer, 2018. doi:10.1007/978-3-319-96878-0_5.
- 36 Masaru Kada, Harumichi Nishimura, and Tomoyuki Yamakami. The efficiency of quantum identity testing of multiple states. *Journal of Physics A: Mathematical and Theoretical*, 41(39):395309, 2008.
- 37 Dakshita Khurana and Kabir Tomer. Commitments from quantum one-wayness. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 968–978, 2024. doi:10.1145/3618260.3649654.
- 38 Fuyuki Kitagawa, Ryo Nishimaki, and Takashi Yamakawa. Secure software leasing from standard assumptions. In *Theory of Cryptography Conference*, pages 31–61. Springer, 2021. doi:10.1007/978-3-030-90459-3_2.
- 39 Jiahui Liu, Qipeng Liu, Luowen Qian, and Mark Zhandry. Collusion resistant copy-protection for watermarkable functionalities. In *Theory of Cryptography Conference*, pages 294–323. Springer, 2022. doi:10.1007/978-3-031-22318-1_11.
- 40 Tony Metger, Alexander Poremba, Makrand Sinha, and Henry Yuen. Simple constructions of linear-depth t-designs and pseudorandom unitaries, 2024. doi:10.48550/arXiv.2404.12647.

- 41 Tomoyuki Morimae, Alexander Poremba, and Takashi Yamakawa. Revocable quantum digital signatures. *arXiv preprint*, 2023. [arXiv:2312.13561](#).
- 42 Tomoyuki Morimae and Takashi Yamakawa. One-wayness in quantum cryptography. *arXiv preprint*, 2022. [arXiv:2210.03394](#).
- 43 Tomoyuki Morimae and Takashi Yamakawa. Quantum commitments and signatures without one-way functions. In *Annual International Cryptology Conference*, pages 269–295. Springer, 2022. doi:10.1007/978-3-031-15802-5_10.
- 44 Alexander Poremba. Quantum proofs of deletion for learning with errors. *arXiv preprint*, 2022. [arXiv:2203.01610](#).
- 45 Ronald L Rivest. Can we eliminate certificate revocation lists? In *Financial Cryptography: Second International Conference, FC’98 Anguilla, British West Indies February 23–25, 1998 Proceedings 2*, pages 178–183. Springer, 1998. doi:10.1007/BFB0055482.
- 46 Stuart G Stubblebine. Recent-secure authentication: Enforcing revocation in distributed systems. In *Proceedings 1995 IEEE Symposium on Security and Privacy*, pages 224–235. IEEE, 1995. doi:10.1109/SECPRI.1995.398935.
- 47 Dominique Unruh. Revocable quantum timed-release encryption. Cryptology ePrint Archive, Paper 2013/606, 2013. doi:10.1145/2817206.
- 48 Stephen Wiesner. Conjugate coding. *SIGACT News*, 15(1):78–88, January 1983. doi:10.1145/1008908.1008920.
- 49 Wikipedia. Certificate revocation, 2024. URL: https://en.wikipedia.org/wiki/Certificate_revocation.
- 50 William K Wootters and Wojciech H Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, 1982.
- 51 Mark Zhandry. A note on quantum-secure prps, 2016. [arXiv:1611.05564](#).
- 52 Mark Zhandry. Quantum lightning never strikes the same state twice. or: quantum money from cryptographic assumptions. *Journal of Cryptology*, 34:1–56, 2021.

Multi-Source Randomness Extraction and Generation in the Random-Oracle Model

Sandro Coretti  

IOG, Paris, France

Pooya Farshim  

IOG, Zürich, Switzerland

Durham University, UK

Patrick Harasser  

Technische Universität Darmstadt, Germany

Karl Southern  

Durham University, UK

Abstract

We study the multi-source randomness extraction and generation properties of the monolithic random oracle (RO), whereby one is tasked with extracting or generating uniform random bits from multiple unpredictable sources. We formalize this problem according to the query complexities of the involved parties – sources, distinguishers, and predictors, where the latter are used to define unpredictability.

We show both positive and negative results. On the negative side, we rule out definitions where the predictor is not at least as powerful as the source or the distinguisher. On the positive side, we show that the RO is a multi-source extractor when the query complexity of the distinguisher is bounded. Our main positive result in this setting is with respect to arbitrary unpredictable sources, which we establish via a combination of a compression argument (Dodis, Guo, and Katz, EUROCRYPT’17) and the decomposition of high min-entropy sources into flat sources.

Our work opens up a rich set of problems, ranging from statistical multi-source extraction with respect to unbounded distinguishers to novel decomposition techniques (Unruh, CRYPTO’07; Coretti et al., EUROCRYPT’18) and multi-source extraction for non-monolithic constructions.

2012 ACM Subject Classification Security and privacy → Cryptography

Keywords and phrases Multi-source randomness extraction, Multi-source randomness generation, Compression argument, Convex decomposition

Digital Object Identifier 10.4230/LIPIcs.ITC.2025.10

Related Version *Full Version:* <https://eprint.iacr.org/2025/1258> [12]

Funding *Pooya Farshim:* Supported in part by EPSRC grant EP/V034065/1.

Patrick Harasser: Funded by the Deutsche Forschungsgemeinschaft (DFG) – SFB 1119 – 236615297.

Acknowledgements We thank Balthazar Bauer for many fruitful discussions and for participating in the early stages of this work. We also thank the anonymous reviewers who helped improve the presentation of our results.

1 Introduction

1.1 Overview

Randomness Extraction. High-quality, close-to-uniform randomness is indispensable in cryptography for many applications (encryption, authentication, zero-knowledge proofs, and multi-party computation, to name but a few). In fact, there are a number of cryptographic tasks that simply cannot be achieved without perfect randomness [18, 21, 28]. However, while “nature” provides many sources that contain some amounts of entropy (e.g., those



© Sandro Coretti, Pooya Farshim, Patrick Harasser, and Karl Southern;
licensed under Creative Commons License CC-BY 4.0

6th Conference on Information-Theoretic Cryptography (ITC 2025).

Editor: Niv Gilboa; Article No. 10; pp. 10:1–10:23



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

based on keystrokes, mouse movements, the timing of interrupts, or passwords), sources that output perfectly uniform random bits are practically non-existent. A fundamental task is therefore so-called *randomness extraction*, which involves obtaining uniform randomness from high-entropy sources. One of the key principles in randomness extraction is that no assumption is made about the sources, except that they have high min-entropy; doing otherwise would entail the need for a very deep understanding of the sources, which may be impossible in practice.

An important hurdle to randomness extraction is that deterministic extractors do not exist [9], i.e., there is no function f whose output is close to uniform on *all* high-entropy sources. A simple counterexample is the so-called “extractor-fixing” attack, where one considers the sources $\mathcal{S}_0$ and $\mathcal{S}_1$ that sample an input uniformly among those on which the output of f starts with a zero and a one, respectively. At least one of these sources, say, $\mathcal{S}_0$, has close to maximal min-entropy, but the output $f(\mathcal{S}_0)$ is certainly not uniformly random (it always starts with 0).

One way to circumvent this issue involves seeded extractors [29], where f receives an additional uniformly random value s – the seed – as input, and the requirement is that for all *seed-independent* high-entropy sources $\mathcal{S}$, $(f(s, \mathcal{S}), s)$ be close to uniform. Such seeded extractors may be useful for certain applications (e.g., privacy amplification), but in the context of randomness generation, their drawback is that they need a uniform seed in the first place. Moreover, even if one was somehow able to produce a “once-and-for-all” random seed, it is unrealistic to assume that sources would not depend on this (publicly known) value.

For these reasons, one often uses cryptographic hash functions (CHFs) instead of seeded extractors in practice. However, the only CHF-based extractor constructions known in the standard model (achieving in fact only the weaker notion of randomness condensing) require non-standard assumptions [19]. Therefore, an approach commonly taken is to analyze constructions in an idealized model, where the extractor or parts of it are modeled as an idealized primitive. Instances of this approach include:

1. Modeling the entire extractor as a monolithic random oracle;
2. Merkle–Damgård-based constructions where the compression function is treated as a random function;
3. The Davies–Meyer compression function with the block cipher modeled as an ideal cipher;
4. Sponge-based constructions with the round function treated as a random permutation.

The analysis in an idealized model can serve as heuristic justification for the use of these constructions in practice. Another benefit of the CHF-based approach in idealized models is that the final constructions are often very efficient.

Extraction in Idealized Models. When studying extraction in idealized models, there are two possibilities. One is to consider only oracle-independent sources (as done, e.g., in [15, 7, 32, 35]). This approach provides limited realism, since the oracle abstracts a CHF, and it is unreasonable to assume that sources are completely independent of it. For example, the timing of an interrupt may well depend on computations related to the hash function in use. Furthermore, if one does not allow oracle-dependent sources, results in the idealized model do not match those in the standard model: For example, the extractor-fixing attack can in general not be performed without oracle access, as the idealized primitive essentially acts as a (very large) seed.

The other option is to consider oracle-dependent sources. However, here the extractor-fixing attack reappears, and at a first glance, the only solution seems to be using a seed (as done, e.g., in [24, 20]). Fortunately, there are other alternatives: Work by Coretti et al. [11]

weakens the notion of oracle-dependence by requiring that the output of a source have entropy even given the queries the source made to the idealized primitive and the corresponding answers. This essentially amounts to modeling that “natural” sources occurring in practice add extra entropy on top of the CHF evaluations occurring around them.

Independent Sources of Entropy. Another way of avoiding a seed is to consider split entropy sources, i.e., sources that produce independent values satisfying certain min-entropy conditions. The study of such “multi-source” extractors began with the work of Chor and Goldreich [9], who introduced the problem of extracting randomness from two or more independent weak sources. Since then, many works have improved the entropy requirements, output length, and error parameters of extractors for both the two-source and the multi-source settings [2, 14, 27, 8]. These developments have led to increasingly efficient constructions, with broad applications in derandomization, cryptography, and complexity theory.

When studying the practicality of such extractors, it is pertinent to first examine whether sources of entropy that can be assumed to be independent actually exist. The answer appears to be affirmative when considering sources used in practice: For instance, Intel Secure Key, a hardware-based random number generator integrated into most modern Intel processors, uses thermal noise as entropy source [26]. Another source of entropy is sensor data measuring different physical properties, such as temperature and light intensity. Yet other sources use user-driven events such as keystrokes, mouse movements, and touchscreen interactions. All these measurements are influenced by distinct physical processes and environmental factors, and can therefore reasonably be considered independent.

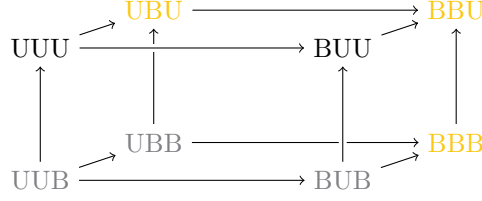
In spite of the theoretical advancements mentioned above, for practically relevant constructions of extractors one still resorts to CHFs and, to avoid non-standard assumptions, to idealized models. As a result, in this work we initiate the study of CHF-based multi-source extractors by providing suitable ideal-model definitions and by analyzing, as a first step, the security of the random oracle itself as a multi-source extractor wrt. several definitional variants. Investigating the security of non-monolithic constructions (e.g., Merkle–Damgård or Sponges) is left as an exciting open problem.

1.2 Our Contributions

Multi-Source Extractors in Idealized Models. A first contribution of this work is to put forth a definitional framework that captures extraction in idealized models, allowing the source to depend on the idealized primitive. When defining oracle-dependent extraction in an idealized model, the following definitional choices are possible:

1. The source can either fully depend on the ideal primitive (IP) or be computationally restricted by limiting the number of queries it makes to the IP;
2. To measure the entropy (or unpredictability) of the source, one can require unpredictability conditioned on the entire IP or only with respect to a predictor with bounded-query access to the IP;
3. For the distinguisher trying to tell the extractor output from a uniform string, one can either consider an unbounded party that gets the entire oracle as input or one that is only allowed a limited number of queries to the IP.

These choices result in $2^3 = 8$ possible notions, captured by a simple “source-distinguisher-predictor” (*SDP*) notation, where each entity can be “bounded” (B) or “unbounded” (U) depending on the type of access the corresponding party has to the IP (see Figure 1). We establish a complete set of generic implications between these notions.



■ **Figure 1** Multi-source extractor notions in the random-oracle model. Notions are labeled via an “SDP” string (for *Source*, *Distinguisher*, and *Predictor*), with $S, D, P \in \{B, U\}$ depending on whether the corresponding party has bounded-query (B) or unbounded (U) access to the random oracle. An arrow from X to Y means that any construction for X is also a construction for Y . Corners typeset in gray are shown not to be achieved by the monolithic construction via attacks presented in Section 4, specifically Proposition 17 (UBB) and Proposition 18 (BUB). Corners typeset in yellow are achieved in Sections 5 and 6, specifically Theorem 19 (BBB) and Theorem 23 (UBU).

Our definitions are captured by two games: In the first game, the source interacts with the IP to produce values x and z , where x is the input to the extractor and z is some leakage about x . A predictor, also with access to the IP, is then given z and tries to predict x . The second game is the extraction game, in which $\ell \geq 1$ sources interact with the oracle to produce otherwise independently sampled (x_i, z_i) , and a distinguisher attempts to tell apart the extractor output on the x_i from a uniform string, given the leakage strings z_i .

Note that this definitional framework also captures as a special case the entropy notion from [11], in which there is a single bounded source, and the predictor as well as the distinguisher are bounded. This is done by considering canonical bounded sources, which output the query/answer list in the leakage.

Results. We then go on to investigate whether the monolithic random oracle is a good multi-source extractor. Specifically, we make the following findings:

1. We first establish a positive result for one of the easiest cases, BBB (see Theorem 19). This result also implies the case BBU, as making the predictor unbounded reduces the class of sources.
2. We then turn to the main focus of this work, involving unbounded sources. This is a crucial setting, as it minimizes the assumptions on the source.
 - a. We first observe that the random oracle does not work as an extractor when the predictor is bounded (i.e., the UBB and UUB configurations). This is due to the fact that sources with unbounded oracle access can return a fixed point in the RO. While such sources are unpredictable for a bounded-query predictor, a bounded distinguisher can easily check if the given input is fixed under the RO (see Proposition 17).
 - b. The main result of our paper is to show that the monolithic random oracle is an effective extractor in the UBU setting (see Theorem 23), i.e., with unbounded source and predictor, but with a bounded distinguisher. This is an important case as, in practice, one can reasonably assume distinguishers to be bounded.
3. Finally, we show an attack against BUB, where sources can leak some information about their output to the distinguisher, which can use its unbounded oracle access to check if the challenge it is given has a preimage compatible with the leakage (see Proposition 18).

Finding efficient constructions when the distinguisher is unbounded (i.e., for UUU and BUU) is left for future work.

Comparison to Coretti et al. [11]. Comparing our main result to the analogous one in [11], we observe that the two statements are formally incomparable. Indeed, our notion assumes at least two *independent, unbounded* sources, while their setting assumes a *single, bounded* source in the BBU setting and that the source leaks its random oracle queries.

Techniques. In our main result, the source can depend arbitrarily on the random oracle. This dependency allows the source to create arbitrary correlations in the distribution of the oracle, conditioned on the source’s output and potential leakage. One way of dealing with such correlations is to use the compression technique of Gennaro and Trevisan [25].

Our proof strategy follows that of Dodis et al. [16], which in turn is based on De et al. [13]. We begin by using a compression argument to establish that a random oracle, when applied to inputs $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_\ell)$ produced by independent sources $\mathcal{S}_1(H), \dots, \mathcal{S}_\ell(H)$, satisfies the property of unrecoverability: It is computationally infeasible to recover $\mathbf{x}$ from $H(\mathbf{x})$, even given the leakage of the (unpredictable) sources. The proof then transitions to reducing distinguishing $H(\mathbf{x})$ from a random output to a next-bit guessing problem. The difficulty of next-bit guessing is in turn established via reduction to the unrecoverability property and via an incompressibility argument that uses a next-bit guesser to compress oracles.

Throughout these steps, it is crucial to account for the fact that the inputs $\mathbf{x}_i$ are not uniformly random and independent of the oracle, as is the case in [16]. Instead, they can fully depend on the oracle and are only guaranteed to have entropy conditioned on the oracle. To handle this obstacle, we decompose high-entropy sources into convex combinations of flat sources [34] at the necessary steps in the proof.

1.3 Related Work

There are numerous other models in which one can study randomness extraction. Some examples include:

- Dodis et al. [22], who investigate extracting randomness from extractor-dependent sources that are capable of making black-box queries to the seeded extractor;
- The study of Santha–Vazirani (SV) sources, which consist of sequences of random bits where an adversary has partial control over the conditional distribution of each bit given the previous bits – extraction from such sources is known to be impossible [31], but more recent work has shown that extraction is feasible for non-binary SV sources [3];
- Aggarwal et al. [1], who consider a weaker type of sources called SHELA sources, which do not allow for uniform extraction but for a weaker notion of “somewhere extraction” that is sufficient for some applications.

The main difference between our work and the majority of existing results is that they focus on providing explicit constructions in the standard model. For instance, the work of Dodis and Oliveira [17] uses the probabilistic method to show the existence of so-called super-strong extractors (and then provides explicit constructions thereof).¹ Note the qualitative difference between proving the existence of a not necessarily efficient extractor in the standard model via the probabilistic method versus our treatment in the random-oracle model, where the focus is on validating cryptographic constructions of extractors in an idealized model, with the monolithic extractor being the first object of study. We further comment on the differences between these results and our work in the relevant sections of the paper.

¹ In our terminology, super-strong extractors are two-source extractors that work even if one of the two inputs is made public.

1.4 Future Directions

Our work shows that compression is sufficiently powerful to establish multi-source extraction and generation bounds for the monolithic random oracle. However, compression is not straightforward to apply when the idealized object is not fully random, as is the case for many non-monolithic constructions such as Merkle–Damgård or Sponge.² This raises the question of whether other techniques can be extended to work in the multi-source setting.

The decomposition techniques of Coretti et al. (CDGS) [10] and Unruh [33] are alternative techniques that can lead to simpler proofs, and can therefore be potentially applied to non-monolithic constructions in the multi-source setting. In CDGS, for a parameter $P \in \mathbb{N}$, one decomposes the random oracle conditioned on some RO-dependent leakage z into a convex combination of sources, each of which is fixed on at most P points and dense otherwise (plus some small error distribution). CDGS use such a decomposition to show that the RO model with auxiliary information (AI-ROM) is indistinguishable from a P -bit-fixed (BF) RO model, where some inputs to the RO are fixed to values computed during decomposition on leakage z , and the other values are uniformly random. The statistical difference between the two settings is $\mathcal{O}(ST/P)$, where S is an upper bound on the size of the leakage z , T is the query complexity of the distinguisher, and P is the number of fixed points. In applications, P cannot be chosen too large, as then the probability of certain bad events (e.g., a challenge point lying in the set of fixed points) would be too large.

Unruh decomposition is conceptually simpler in that it decomposes the original random oracle and the set of fixed points is compatible with the given random oracle. The distinguishing advantage, however, is now $\mathcal{O}(\sqrt{ST/P})$.

A natural question is whether decomposition techniques like CDGS or Unruh can be used in the multi-source setting. This turns out not to be the case, at least when these techniques are applied directly in *parallel* to fix points for each source separately, and then fixing the points in the union of these sets.³ Indeed, consider sources $\mathcal{S}_1$ and $\mathcal{S}_2$ that leak $z_1 := \sum_{1 \neq x \in [N]} H(x)$ and $z_2 := \sum_{x \in [N]} H(x)$. Clearly, given (z_1, z_2) , the value $H(1)$ is revealed. That is, an AI-ROM adversary can determine $H(1)$ given leakage (z_1, z_2) . On the other hand, z_1 and z_2 *individually* leak very little information about the hash of any particular point: $H(1)$ is random and independent of the hashes of all other points. This means that running the CDGS decomposition in parallel on z_1 and z_2 will not fix $H(1)$. Consequently, a BF-ROM adversary will not be able to predict $H(1)$.⁴

Despite these attacks, we conjecture that parallel decomposition *is* possible under appropriate restrictions. Such parallel decomposition techniques would be helpful in studying the multi-source extraction properties of non-monolithic constructions. We leave the development of these techniques to future work.

1.5 Structure of the Paper

We start by recalling the preliminaries, including the compression lemmas, in Section 2. In Section 3, we formally define multi-source extractors with respect to different source classes, unpredictability notions, and indistinguishability metrics. In Section 4, we rule out

² Note that such a result does not follow from the indifferentiability of these constructions, as the multi-source extractor game, due to the unpredictability requirement on the sources, is not single-stage [30].

³ Another approach would be to combine the sources and apply decomposition to the resulting source. Multi-source extraction, however, does not hold with respect to a single source, and accordingly it is unclear how one would upper-bound the probability of sampled points lying in the set of fixed points.

⁴ We note that this is an attack on directly applying decomposition as a proof technique. Moreover, z_1 and z_2 can be made unpredictable by simply appending random bits.

the notions where the predictor is not as powerful as either the source or the distinguisher. Section 5 contains our security result for distinguishers and sources with a bounded number of queries. In Section 6, we present our main contribution on the multi-source randomness extraction with respect to arbitrary unpredictable sources. This leaves two notions, where both the distinguisher and predictor are unbounded, open for future work.

2 Preliminaries

Basic Notation. We denote by $\mathbb{N}$ and $\mathbb{R}$ the sets of natural and of real numbers, and for $n \in \mathbb{N}$ we define $[n] := \{1, \dots, n\}$. The cardinality of a set S is denoted $|S|$, and we let S^* be the set of finite-length tuples over S . As usual, the empty tuple is denoted ε . If $\mathcal{X}$ is a random variable, then sampling from $\mathcal{X}$ is denoted $x \leftarrow \mathcal{X}$; when $\mathcal{X}$ is uniformly distributed over a finite set X , we write $x \leftarrow X$ instead. The support of a random variable $\mathcal{X}$ is denoted $\text{Supp}(\mathcal{X})$. For $\ell, M, N \in \mathbb{N}$, we write $\text{Fun}(N^\ell, M)$ for the set of functions from $[N]^\ell$ to $[M]$. We use boldface notation for vectors, as in $\mathbf{x} = (x_1, \dots, x_\ell)$, and we write $|\mathbf{x}|$ for the length of the vector $\mathbf{x}$. If not stated otherwise, indexes in vectors and strings start from 1. All logarithms are to base 2.

ROM [23, 5]. Let $\ell, M, N \in \mathbb{N}$. We define the random-oracle model $\mathcal{RO}[\ell, N, M]$ as the uniform distribution on $\text{Fun}(N^\ell, M)$. When operating in this model, all algorithms (both honest and adversarial) can query a function $H \in \text{Fun}(N^\ell, M)$ as an oracle. In security games, one first samples H uniformly from $\text{Fun}(N^\ell, M)$. Afterwards, the game is run with all parties having oracle access to H . For the remainder of this paper, we will always work in this setting.

Fundamental Lemma of Game Playing [6]. We use the code-based game-playing framework of Bellare and Rogaway. Let G_1 and G_2 be two games whose code is identical except for the consequent inside one if-branch, and let **Bad** be the event that the if-statement is triggered. Then $|\Pr[G_1] - \Pr[G_2]| \leq \Pr[\text{Bad}]$.

Monolithic Construction. Let $\ell, M, N \in \mathbb{N}$. The monolithic construction is the algorithm $\text{Mono} := \text{Mono}[\ell, N, M]$ which takes a vector $\mathbf{x} \in [N]^\ell$ as input, has oracle access to $H \in \text{Fun}(N^\ell, M)$, and returns $y := H(\mathbf{x}) \in [M]$.

Source Codes. Let S be a set. A (deterministic) code for S is a pair (Enc, Dec) , where $\text{Enc}: S \rightarrow \{0, 1\}^l$ and $\text{Dec}: \{0, 1\}^l \rightarrow S$ are functions, and $l \in \mathbb{N}$. A randomized encoding for S is a pair (Enc, Dec) , where $\text{Enc}: S \times R \rightarrow \{0, 1\}^l$ and $\text{Dec}: \{0, 1\}^l \times R \rightarrow S$ are functions, R is a set, and $l \in \mathbb{N}$.

► **Lemma 1** (Deterministic compression [13, 16]). *Let S be a set, and let (Enc, Dec) be a code for S such that, for every $m \in S$, $\text{Dec}(\text{Enc}(m)) = m$. Then*

$$\mathbb{E}_{m \leftarrow S} [|\text{Enc}(m)|] \geq \log |S|.$$

► **Lemma 2** (Randomized compression [13, 16]). *Let S be a set, $\delta \in \mathbb{R}_{>0}$, and let (Enc, Dec) be a randomized encoding for S with recovery probability δ , meaning that for every $m \in S$,*

$$\Pr_{r \leftarrow R} [\text{Dec}(\text{Enc}(m, r), r) = m] \geq \delta.$$

Then $l \geq \log |S| - \log 1/\delta$.

Flat Sources. Let X be a random variable over a finite set S and $k \in \mathbb{R}_{>0}$.

1. We call X a k -source if, for every $x \in S$, $\Pr[X = x] \leq 2^{-k}$.
2. We say that X is flat if it is uniformly distributed over a subset $T \subseteq S$.
3. Assume that $2^k \in \mathbb{N}$. We call X a flat k -source if it is uniformly distributed over a subset $T \subseteq S$ with $|T| = 2^k$.

► **Lemma 3** (Flat decomposition of sources [34]). *Let $k \in \mathbb{R}_{>0}$ with $2^k \in \mathbb{N}$. Then every k -source X is a convex combination of flat k -sources, i.e., $X = \sum p_i X_i$ with $0 \leq p_i \leq 1$ for all i , $\sum p_i = 1$, and all the X_i are flat k -sources.*

Numeric Inequalities. We conclude by recalling three inequalities for real numbers and their powers that we will use in this work.

► **Lemma 4** (Numeric inequalities). *Let $e \in \mathbb{R}$ denote Euler's number. Then:*

1. *For all $x \in \mathbb{R}$ with $x \geq 1$, it holds that $(1 - 1/x)^x \leq e^{-1}$.*
2. *For all $x \in \mathbb{R}$ with $x \geq -1$, and all $r \in \mathbb{N}$, it holds that $(1 + x)^r \geq 1 + rx$.*
3. *For all $k, n \in \mathbb{N}$ with $0 < k \leq n$, it holds that $(n/k)^k \leq \binom{n}{k} \leq (en/k)^k$.*

3 Multi-Source Extractors

In this section, we formally introduce multi-source extractors (MSEs) in the random-oracle model. We first formalize unpredictability (Pred) of sources, the property that forms the underlying assumption in all results that we prove. We then go on to defining unrecoverability (UR) and MSE for Mono. Our notions are parameterized by the query complexity of the various parties in the Pred, UR, and MSE games, and by the advantages they achieve in these games. We present our definitions for the monolithic construction Mono, but generalizations to other constructions and idealized models are straightforward. Finally, we prove implications between these notions and between different parameter sets for the same notion.

► **Definition 5** (Sources, Adversaries, Distinguishers, and Predictors). *Let $\ell, M, N \in \mathbb{N}$.*

1. *A source is an algorithm $\mathcal{S}$ that takes no input and returns a pair $(x, z) \in [N] \times \{0, 1\}^*$.*
2. *An adversary is an algorithm $\mathcal{A}$ which, on input $y \in [M]$ and a vector $\mathbf{z} \in (\{0, 1\}^*)^\ell$ of strings, returns $\mathbf{x}' \in [N]^\ell$.*
3. *A distinguisher is an algorithm $\mathcal{D}$ which, on input $y \in [M]$ and a vector $\mathbf{z} \in (\{0, 1\}^*)^\ell$, returns a bit.*
4. *A predictor is an algorithm $\mathcal{P}$ which, on input $z \in \{0, 1\}^*$, returns $x' \in [N]$.*

All these algorithms are probabilistic and unbounded, and additionally receive oracle access to a function $H \in \text{Fun}(N^\ell, M)$.

► **Definition 6** (Unpredictability). *Let $\ell, M, N \in \mathbb{N}$, $\mathcal{S}$ be a source, and $\mathcal{P}$ a predictor. We define the advantage of $\mathcal{P}$ in the prediction game for $\mathcal{S}$ as $\text{Adv}_{\mathcal{S}, \mathcal{P}}^{\text{pred}} := \Pr[\text{Pred}_{\mathcal{S}}^{\mathcal{P}}]$, where the game $\text{Pred}_{\mathcal{S}}^{\mathcal{P}}$ is given in Figure 2. For $q \in \mathbb{N}$ and $\delta \in \mathbb{R}_{\geq 0}$, we say that $\mathcal{S}$ is (q, δ) -unpredictable if for every predictor $\mathcal{P}$ making at most q oracle calls, we have $\text{Adv}_{\mathcal{S}, \mathcal{P}}^{\text{pred}} \leq \delta$.*

► **Remark 7.** The notion of unpredictability that we consider represents the first main point of difference with several prior works. For example, the class of sources that we consider is *larger* than the one in the work of Dodis and Oliveira [17], who essentially require high min-entropy for every oracle. We on the other hand require unpredictability to hold only on average over the choice of the RO. This approach has been used in other prior works, e.g., [11, 4]. In particular, the class of sources that we consider contains the one of Coretti et al. [11] as a subclass, by setting the leakage z to be the query-answer pairs of the source.

Game $\text{Pred}_{\mathcal{S}}^{\mathcal{P}}$: $H \leftarrow \text{Fun}(N^\ell, M)$ $(x, z) \leftarrow \mathcal{S}^H$ $x' \leftarrow \mathcal{P}^H(z)$ return $(x = x')$	Game $\text{UR}^{\mathcal{S}, \mathcal{A}}$: $H \leftarrow \text{Fun}(N^\ell, M)$ for $i = 1$ to ℓ do $(x_i, z_i) \leftarrow \mathcal{S}_i^H$ $y \leftarrow H(x)$; $x' \leftarrow \mathcal{A}^H(y, z)$ return $(x = x')$	Game $\text{MSE}^{\mathcal{S}, \mathcal{D}}$: $H \leftarrow \text{Fun}(N^\ell, M)$; $b \leftarrow \{0, 1\}$ for $i = 1$ to ℓ do $(x_i, z_i) \leftarrow \mathcal{S}_i^H$ $y_0 \leftarrow [M]$; $y_1 \leftarrow H(x)$ $b' \leftarrow \mathcal{D}^H(y_b, z)$; return $(b = b')$
--	--	--

■ **Figure 2** The prediction game for a source $\mathcal{S}$, and the unrecoverability and multi-source-extractor games for the monolithic construction Mono.

► **Remark 8.** The class of sources that we consider can leak information to the distinguisher or predictor via z . One benefit of this definitional choice is that it allows modeling of auxiliary input via *preprocessing*. Indeed, we can assume without loss of generality that such auxiliary information is computed deterministically, and thus every source can recompute it and include it as part of its leakage. Unpredictability is preserved under this transformation as long as predictors are unbounded and can also recompute the auxiliary information.

► **Definition 9 (Unrecoverability).** Let $\ell, M, N \in \mathbb{N}$, $\mathcal{S} = (\mathcal{S}_1, \dots, \mathcal{S}_\ell)$ be a tuple of sources, and $\mathcal{A}$ an adversary. We define the advantage of $(\mathcal{S}, \mathcal{A})$ in the unrecoverability game for Mono as $\text{Adv}_{\mathcal{S}, \mathcal{A}}^{\text{ur}} := \Pr[\text{UR}^{\mathcal{S}, \mathcal{A}}]$, where the game $\text{UR}^{\mathcal{S}, \mathcal{A}}$ is given in Figure 2. For $q_{\mathcal{S}}, q_{\mathcal{A}}, q_{\mathcal{P}} \in \mathbb{N}$ and $\delta, \epsilon \in \mathbb{R}_{\geq 0}$, we say that Mono is $(q_{\mathcal{P}}, \delta, q_{\mathcal{S}}, q_{\mathcal{A}}, \epsilon)$ -unrecoverable if for every $(\mathcal{S}, \mathcal{A})$ as above such that, for every $i \in [\ell]$, $\mathcal{S}_i$ is $(q_{\mathcal{P}}, \delta)$ -unpredictable and makes at most $q_{\mathcal{S}}$ oracle calls, and such that $\mathcal{A}$ makes at most $q_{\mathcal{A}}$ oracle calls, we have $\text{Adv}_{\mathcal{S}, \mathcal{A}}^{\text{ur}} \leq \epsilon$.

► **Definition 10 (Multi-Source Extraction).** Let $\ell, M, N \in \mathbb{N}$, $\mathcal{S} = (\mathcal{S}_1, \dots, \mathcal{S}_\ell)$ be a tuple of sources, and $\mathcal{D}$ a distinguisher. We define the advantage of $(\mathcal{S}, \mathcal{D})$ in the multi-source-extraction game for Mono as $\text{Adv}_{\mathcal{S}, \mathcal{D}}^{\text{mse}} := 2 \cdot \Pr[\text{MSE}^{\mathcal{S}, \mathcal{D}}] - 1$, where the game $\text{MSE}^{\mathcal{S}, \mathcal{D}}$ is given in Figure 2. For $q_{\mathcal{S}}, q_{\mathcal{D}}, q_{\mathcal{P}} \in \mathbb{N}$ and $\delta, \epsilon \in \mathbb{R}_{\geq 0}$, we say that Mono is a $(q_{\mathcal{P}}, \delta, q_{\mathcal{S}}, q_{\mathcal{D}}, \epsilon)$ -multi-source-extractor (MSE for short) if for every $(\mathcal{S}, \mathcal{D})$ as above such that, for every $i \in [\ell]$, $\mathcal{S}_i$ is $(q_{\mathcal{P}}, \delta)$ -unpredictable and makes at most $q_{\mathcal{S}}$ oracle calls, and such that $\mathcal{D}$ makes at most $q_{\mathcal{D}}$ oracle calls, we have $\text{Adv}_{\mathcal{S}, \mathcal{D}}^{\text{mse}} \leq \epsilon$.

► **Remark 11.** Notice that different restrictions on the query complexity q of the parties above allow to formulate qualitatively different results. When q is “small,” one can use well-known techniques (e.g., lazy sampling) to prove security of a given construction. Usually, bounds obtained in this way become meaningless in the regime where q is “large,” e.g., large enough to query H on all possible inputs (which is equivalent to giving H in full as input), so different techniques (e.g., along the lines of [16]) are necessary in this setting.

► **Remark 12.** In this work, we opt for a concrete treatment of security, but an asymptotic formulation is also possible. In that setting, all idealized models, parties, and security notions depend on a security parameter λ . For security to hold, advantages must be negligible in λ . Parties are probabilistic and can be either polynomial-time in λ or unbounded, and in the latter case either have polynomially bounded oracle access to H_λ , or get H_λ in full as input.

► **Remark 13.** Let $\ell, M, N \in \mathbb{N}$, $\mathcal{S} = (\mathcal{S}_1, \dots, \mathcal{S}_\ell)$ a tuple of sources, and $\mathcal{A}$ an adversary in the UR game for Mono. Without loss of generality, we can assume $\mathcal{A}$ to be deterministic. Indeed, if R denotes the random variable representing the randomness of (the possibly randomized adversary) $\mathcal{A}$, we have

$$\begin{aligned} \text{Adv}_{\mathcal{S}, \mathcal{A}}^{\text{ur}} &= \Pr[\text{UR}^{\mathcal{S}, \mathcal{A}}] = \sum_r \Pr[\text{UR}^{\mathcal{S}, \mathcal{A}} \mid R = r] \Pr[R = r] \\ &= \sum_r \Pr[\text{UR}^{\mathcal{S}, \mathcal{A}[r]}] \Pr[R = r] = \sum_r \text{Adv}_{\mathcal{S}, \mathcal{A}[r]}^{\text{ur}} \Pr[R = r], \end{aligned}$$

where $\mathcal{A}[r]$ is the deterministic adversary $\mathcal{A}$ with hard-coded randomness r . This means that there must exist r such that $\text{Adv}_{\mathcal{S},\mathcal{A}}^{\text{ur}} \leq \text{Adv}_{\mathcal{S},\mathcal{A}[r]}^{\text{ur}}$, showing that for every (possibly randomized) adversary $\mathcal{A}$, there exists a deterministic adversary $\mathcal{B}$ such that $\text{Adv}_{\mathcal{S},\mathcal{A}}^{\text{ur}} \leq \text{Adv}_{\mathcal{S},\mathcal{B}}^{\text{ur}}$. It is therefore sufficient to upper-bound the advantage of $(\mathcal{S}, \mathcal{A})$ in the UR game for Mono when $\mathcal{A}$ is deterministic. A similar remark holds for distinguishers $\mathcal{D}$ in the MSE game for Mono.

Generic Implications. We begin by studying the relative strength of the security notions $\text{SEC} \in \{\text{UR}, \text{MSE}\}$. Clearly, if Mono is SEC-secure for a given set of parameters, it remains secure if the sources and the adversary (resp., the distinguisher) are allowed a smaller number of queries, and the predictor can make more queries. Moreover, if Mono is MSE-secure for a given set of parameters, it is also UR-secure in certain parameter ranges. In the following, fix $\ell, M, N \in \mathbb{N}$.

► **Proposition 14.** *Let $q_{\mathcal{S}}, q'_{\mathcal{S}}, q_{\mathcal{A}}, q'_{\mathcal{A}}, q_{\mathcal{D}}, q'_{\mathcal{D}}, q_{\mathcal{P}}, q'_{\mathcal{P}} \in \mathbb{N}$ and $\delta, \delta', \epsilon, \epsilon' \in \mathbb{R}_{\geq 0}$, such that $q'_{\mathcal{S}} \leq q_{\mathcal{S}}, q'_{\mathcal{A}} \leq q_{\mathcal{A}}, q'_{\mathcal{D}} \leq q_{\mathcal{D}}, q'_{\mathcal{P}} \geq q_{\mathcal{P}}, \delta' \leq \delta$, and $\epsilon' \geq \epsilon$. (1) If Mono is $(q_{\mathcal{P}}, \delta, q_{\mathcal{S}}, q_{\mathcal{A}}, \epsilon)$ -unrecoverable, then it is also $(q'_{\mathcal{P}}, \delta', q'_{\mathcal{S}}, q'_{\mathcal{A}}, \epsilon')$ -unrecoverable. (2) If Mono is a $(q_{\mathcal{P}}, \delta, q_{\mathcal{S}}, q_{\mathcal{D}}, \epsilon)$ -MSE, then it is also a $(q'_{\mathcal{P}}, \delta', q'_{\mathcal{S}}, q'_{\mathcal{D}}, \epsilon')$ -MSE.*

Proof. We only prove (1), as the proof of (2) is identical (except that $\mathcal{A}$ is replaced with $\mathcal{D}$ in the following). Let $\mathcal{S} = (\mathcal{S}_1, \dots, \mathcal{S}_{\ell})$ be any tuple of sources such that, for every $i \in [\ell]$, $\mathcal{S}_i$ is $(q'_{\mathcal{P}}, \delta')$ -unpredictable and makes at most $q'_{\mathcal{S}}$ oracle calls, and let $\mathcal{A}$ be any adversary making at most $q'_{\mathcal{A}}$ oracle calls. Then clearly, for each $i \in [\ell]$, $\mathcal{S}_i$ is also $(q_{\mathcal{P}}, \delta)$ -unpredictable and makes at most $q_{\mathcal{S}}$ oracle calls, and $\mathcal{A}$ makes at most $q_{\mathcal{A}}$ oracle calls. By assumption, this means that $\text{Adv}_{\mathcal{S},\mathcal{A}}^{\text{ur}} \leq \epsilon \leq \epsilon'$. Since this inequality holds for every pair $(\mathcal{S}, \mathcal{A})$ as above, by definition Mono is $(q'_{\mathcal{P}}, \delta', q'_{\mathcal{S}}, q'_{\mathcal{A}}, \epsilon')$ -unrecoverable. ◀

► **Proposition 15.** *Let $q_{\mathcal{S}}, q_{\mathcal{D}}, q_{\mathcal{P}} \in \mathbb{N}$ with $q_{\mathcal{D}} \geq 1$, and $\delta, \epsilon \in \mathbb{R}_{\geq 0}$. If Mono is a $(q_{\mathcal{P}}, \delta, q_{\mathcal{S}}, q_{\mathcal{D}}, \epsilon)$ -MSE, then it is $(q_{\mathcal{P}}, \delta, q_{\mathcal{S}}, q_{\mathcal{D}} - 1, \epsilon + N^{\ell}/M)$ -unrecoverable.*

Proof. Let $\mathcal{S} = (\mathcal{S}_1, \dots, \mathcal{S}_{\ell})$ be a tuple of sources such that, for every $i \in [\ell]$, $\mathcal{S}_i$ is $(q_{\mathcal{P}}, \delta)$ -unpredictable and makes at most $q_{\mathcal{S}}$ oracle calls. Let $\mathcal{A}$ be an adversary making at most $q_{\mathcal{D}} - 1$ oracle calls, and consider the distinguisher $\mathcal{D}$ which runs $\mathcal{A}$ on its input (y, z) to get $\mathbf{x}'$, and then returns the bit $(H(\mathbf{x}') = y)$. Then clearly $\mathcal{D}$ makes at most $q_{\mathcal{D}}$ oracle calls. Furthermore notice that, when $b = 1$ in $\text{MSE}^{\mathcal{S}, \mathcal{D}}$, distinguisher $\mathcal{D}$ will return 1 if $\mathcal{A}$ succeeds in finding a preimage, so that $\Pr[\text{MSE}^{\mathcal{S}, \mathcal{D}} \mid b = 1] \geq \text{Adv}_{\mathcal{S},\mathcal{A}}^{\text{ur}}$. On the other hand, when $b = 0$, we have:

$$\begin{aligned} \Pr[\text{MSE}^{\mathcal{S}, \mathcal{D}} \mid b = 0] &\geq \Pr[\text{MSE}^{\mathcal{S}, \mathcal{D}} \mid (b = 0) \wedge (y \notin \text{Rng}(H))] \Pr[y \notin \text{Rng}(H)] \\ &= \Pr[y \notin \text{Rng}(H)], \end{aligned}$$

where the last equality holds because when $y \notin \text{Rng}(H)$, $\mathcal{D}$ will always return 0. Now notice that $\Pr[y \notin \text{Rng}(H)] = (1 - 1/M)^{N^{\ell}} \geq 1 - N^{\ell}/M$, which means

$$\epsilon \geq \text{Adv}_{\mathcal{S},\mathcal{A}}^{\text{mse}} \geq \text{Adv}_{\mathcal{S},\mathcal{A}}^{\text{ur}} + 1 - \frac{N^{\ell}}{M} - 1 = \text{Adv}_{\mathcal{S},\mathcal{A}}^{\text{ur}} - \frac{N^{\ell}}{M}. \quad \blacktriangleleft$$

► **Remark 16.** If bounds on the support sizes $|\text{Supp}(\mathcal{S}_i)|$ are known, we can give a tighter lower bound of $\Pr[y \notin \text{Rng}(H)] \geq 1 - |\text{Supp}(\mathcal{S}_1)| \cdots |\text{Supp}(\mathcal{S}_{\ell})|/M$, showing that Mono is $(q_{\mathcal{P}}, \delta, q_{\mathcal{S}}, q_{\mathcal{D}} - 1, \epsilon + |\text{Supp}(\mathcal{S}_1)| \cdots |\text{Supp}(\mathcal{S}_{\ell})|/M)$ -unrecoverable. Also, notice that this result holds for *any* construction, not just for the monolithic random oracle Mono.

Src. $\mathcal{S}_i^H$: $E_H \leftarrow \{x \in [M] \subseteq [N] \mid H(x, x) = x\}$ if $(E_H \neq \emptyset)$ then $x \leftarrow \min E_H$ else $x \leftarrow [M]$ $z \leftarrow \varepsilon$; return (x, z)	Dist. $\mathcal{D}^H(y, z)$: return $(H(y, y) = y)$	Game $\mathbf{G}_1$: $H \leftarrow E$ $x' \leftarrow \mathcal{P}^H(\varepsilon)$ return $(H(x', x') = x')$
--	--	---

■ **Figure 3** Definition of sources $\mathcal{S}_i$, $i \in [2]$, of the distinguisher $\mathcal{D}$, and of the game $\mathbf{G}_1$ from the proof of Proposition 17.

4 Attacks

In this section, we present attacks against the configurations UBB (Proposition 17) and BUB (Proposition 18) in Figure 1. The former result also rules out UUB, which implies UBB.

► **Proposition 17.** *Let $M, N, q_{\mathcal{P}} \in \mathbb{N}$ with $M \leq N$, and set $\ell := 2$. Then there exist a pair of sources $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$, each of which is $(q_{\mathcal{P}}, 2(q_{\mathcal{P}} + 2)/M)$ -unpredictable and makes $q_{\mathcal{S}} = M$ oracle calls, and a distinguisher $\mathcal{D}$ making $q_{\mathcal{D}} = 1$ oracle calls, such that*

$$\text{Adv}_{\mathcal{S}, \mathcal{D}}^{\text{mse}} \geq 1 - \frac{1}{e} - \frac{1}{M}. \quad (1)$$

Proof. For every $i \in [2]$, let $\mathcal{S}_i$ be the source defined in Figure 3, which returns the smallest $x \in [M] \subseteq [N]$ such that $H(x, x) = x$ if such an x exists, and a random value $x \in [M]$ otherwise. Notice that this definition of $\mathcal{S}_i$ requires it to be allowed up to M oracle queries. Furthermore, let $\mathcal{D}$ be the distinguisher defined in Figure 3, which checks if its input $y \in [M]$ satisfies $H(y, y) = y$.

To show that $\mathcal{S}_i$ is $(q_{\mathcal{P}}, 2(q_{\mathcal{P}} + 2)/M)$ -unpredictable, let $\mathcal{P}$ be any predictor in the prediction game for $\mathcal{S}_i$ making at most $q_{\mathcal{P}}$ oracle calls, and let

$$E := \{H \in \text{Fun}(N^2, M) \mid (\exists x \in [M] \subseteq [N])(H(x, x) = x)\}$$

be the set of functions having a collision as in the definition of $\mathcal{S}_i$. A direct counting argument gives $|E^c| = (M - 1)^M \cdot M^{N^2 - M}$, from which we infer that $|E| = M^{N^2} - (M - 1)^M \cdot M^{N^2 - M}$. The law of total probability now gives

$$\begin{aligned} \Pr[\text{Pred}_{\mathcal{S}_i}^{\mathcal{P}}] &\leq \Pr[\text{Pred}_{\mathcal{S}_i}^{\mathcal{P}} \mid H \in E] + \Pr[\text{Pred}_{\mathcal{S}_i}^{\mathcal{P}} \mid H \notin E] \\ &= \Pr[\text{Pred}_{\mathcal{S}_i}^{\mathcal{P}} \mid H \in E] + \frac{1}{M} \leq \Pr[\mathbf{G}_1] + \frac{1}{M}, \end{aligned}$$

where $\mathbf{G}_1$ is the game defined in Figure 3. Here, the last inequality holds because the uniform distribution on $\text{Fun}(N^2, M)$, conditioned on the outcome being in E , is the uniform distribution on E , and because the winning condition in $\mathbf{G}_1$ is less restrictive than the one in $\text{Pred}_{\mathcal{S}_i}^{\mathcal{P}}$, given that $H \in E$. Indeed, in $\text{Pred}_{\mathcal{S}_i}^{\mathcal{P}}$, predictor $\mathcal{P}$ must guess the exact output x of $\mathcal{S}_i$, which satisfies $H(x, x) = x$, whereas in $\mathbf{G}_1$ it wins if it returns any such value $x' \in [N]$.

To bound $\Pr[\mathbf{G}_1]$, we transition to yet another game $\mathbf{G}_2$ that proceeds as $\mathbf{G}_1$, but we let $\mathcal{P}$ win if any of its oracle queries of the form $(z, z) \in [M]^2$ (there are at most $q_{\mathcal{P}}$ of them), or its final output x' , satisfy the winning condition of $\mathbf{G}_1$. Then again $\Pr[\mathbf{G}_1] \leq \Pr[\mathbf{G}_2]$, because $\mathcal{P}$ will always win $\mathbf{G}_2$ if it wins $\mathbf{G}_1$. Now notice that, by definition, predictor $\mathcal{P}$ will lose game $\mathbf{G}_2$ if for all queries of the form $(z, z) \in [M]^2$ it makes, it holds $H(z, z) \neq z$, and the same is true for its output x' . The probability of this happening is at least

$$\frac{(M - 1)^{q_{\mathcal{P}} + 1} \cdot M^{N^2 - (q_{\mathcal{P}} + 1)} - |E^c|}{|E|} = \frac{(1 - 1/M)^{q_{\mathcal{P}} + 1} - (1 - 1/M)^M}{1 - (1 - 1/M)^M},$$

<u>Src. $\mathcal{S}_i^H$:</u> $x \leftarrow [N]; z \leftarrow H(x, x)$ return (x, z) <u>Dist. $\mathcal{D}^H(y, z)$:</u> for $i = 1$ to 2 do $X_i \leftarrow \{x \in [N] \mid H(x, x) = z_i\}$ return $(\exists x_1 \in X_1)(\exists x_2 \in X_2)$ $(H(x_1, x_2) = y)$	<u>Game $\mathbf{G}_1$:</u> $T \leftarrow \emptyset; z \leftarrow [M]; x' \leftarrow \mathcal{P}^H(z); x \leftarrow [N]$ if $((x, x) \in \text{Dom}(T))$ then return 1 $T[(x, x)] \leftarrow z; \textbf{return } (x = x')$ <u>Game $\mathbf{G}_2$:</u> $T \leftarrow \emptyset; z \leftarrow [M]; x' \leftarrow \mathcal{P}^H(z); x \leftarrow [N]$ return $(x = x')$ <u>Proc. $H(u, v)$:</u> if $((u, v) \notin \text{Dom}(T))$ then $T[(u, v)] \leftarrow [M]$ return $T[(u, v)]$
---	---

■ **Figure 4** Definition of sources $\mathcal{S}_i$, $i \in [2]$, of the distinguisher $\mathcal{D}$, and of the games $\mathbf{G}_1$ and $\mathbf{G}_2$ from the proof of Proposition 18.

which means

$$\Pr[\mathbf{G}_2] \leq 1 - \frac{(1 - 1/M)^{q_{\mathcal{P}}+1} - (1 - 1/M)^M}{1 - (1 - 1/M)^M} \leq \frac{q_{\mathcal{P}} + 1}{M} \cdot \frac{e}{e - 1} \leq \frac{2(q_{\mathcal{P}} + 1)}{M}.$$

Here, the second step uses Inequality 2 from Lemma 4. Collecting all terms above, we obtain the claimed unpredictability bound for sources $\mathcal{S}_i$.

We are now left with bounding the MSE-advantage of $(\mathcal{S}, \mathcal{D})$. To that end, observe that the probability of $(\mathcal{S}, \mathcal{D})$ winning the MSE game when $b = 0$ is $1 - 1/M$, because for any value $y \in [M] \subseteq [N]$, H returns y on any input with probability $1/M$. On the other hand,

$$\begin{aligned} \Pr[\text{MSE}^{\mathcal{S}, \mathcal{D}} \mid b = 1] &\geq \Pr[\text{MSE}^{\mathcal{S}, \mathcal{D}} \mid (H \in E) \wedge (b = 1)] \Pr[H \in E \mid b = 1] \\ &= \Pr[H \in E] = \frac{M^{N^2} - (M - 1)^M \cdot M^{N^2 - M}}{M^{N^2}} = 1 - \left(1 - \frac{1}{M}\right)^M \geq 1 - \frac{1}{e}, \end{aligned}$$

where the last inequality follows from Inequality 1 from Lemma 4. Collecting the two terms above, we obtain (1). ◀

► **Proposition 18.** *Let $M, N, q_{\mathcal{P}} \in \mathbb{N}$, and set $\ell := 2$. Then there exist a pair of sources $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$, each of which is $(q_{\mathcal{P}}, (q_{\mathcal{P}} + 1)/N)$ -unpredictable and makes $q_{\mathcal{S}} = 1$ oracle calls, and a distinguisher $\mathcal{D}$ making $q_{\mathcal{D}} = N^2$ oracle calls, such that*

$$\text{Adv}_{\mathcal{S}, \mathcal{D}}^{\text{mse}} \geq 1 - \frac{\bar{M}^2}{M} - M \cdot \left(\frac{eN}{MM}\right)^{\bar{M}}, \quad (2)$$

where $\bar{M} := \lceil \sqrt[4]{M} \rceil$.

Proof. For every $i \in [2]$, let $\mathcal{S}_i$ be the source defined in Figure 4, which returns a random $x \in [N]$ and $z = H(x, x)$ as leakage. Furthermore, let $\mathcal{D}$ be the distinguisher defined in Figure 4, which finds, for each $i \in [2]$, all inputs of the form (x_i, x_i) satisfying $H(x_i, x_i) = z_i$. Notice that this definition of $\mathcal{D}$ requires it to be allowed up to N^2 oracle queries.

To show that $\mathcal{S}_i$ is $(q_{\mathcal{P}}, (q_{\mathcal{P}} + 1)/N)$ -unpredictable, let $\mathcal{P}$ be any predictor in the prediction game for $\mathcal{S}_i$. Then we have

$$\Pr[\text{Pred}_{\mathcal{S}_i}^{\mathcal{P}}] \leq \Pr[\mathbf{G}_1] \leq \Pr[\mathbf{G}_2] + \frac{q_{\mathcal{P}}}{N} = \frac{1}{N} + \frac{q_{\mathcal{P}}}{N} = \frac{q_{\mathcal{P}} + 1}{N},$$

where games G_1 and G_2 are defined in Figure 4. Here, the first inequality holds because $\mathcal{P}$ will always succeed in G_1 if it wins the prediction game for $\mathcal{S}_i$, and the second inequality follows from the fundamental lemma of game playing. Indeed, if we let **Bad** be the event in game G_1 that $(x, x) \in \text{Dom}(T)$, then G_1 and G_2 are identical until **Bad**, and $\Pr[\text{Bad}] \leq q_{\mathcal{P}}/N$ because x is random in $[N]$ and $\text{Dom}(T)$ contains at most $q_{\mathcal{P}}$ entries. Finally, the first equality in the chain above holds because x is uniformly random in $[N]$, and will thus be equal to x' with probability $1/N$.

We are now left with bounding the MSE-advantage of $(\mathcal{S}, \mathcal{D})$. To that end, observe that the probability of $(\mathcal{S}, \mathcal{D})$ winning the MSE game for **Mono** when $b = 1$ is 1, because when y is the real hash value, then a preimage of y in $X_1 \times X_2$ exists by construction. On the other hand, for the case $b = 0$, define

$$E := \left\{ H \in \text{Fun}(N^2, M) \mid \begin{array}{l} (\forall k \in \mathbb{N}) ((\exists x_1, \dots, x_k \in [N]) \\ (x_i \neq x_j \wedge H(x_i, x_i) = H(x_j, x_j)) \implies (k < \bar{M})) \end{array} \right\},$$

where $\bar{M} := \lceil \sqrt[4]{M} \rceil$. In other words, E is the set of all functions $H \in \text{Fun}(N^2, M)$ such that there are less than $\bar{M}$ pairwise distinct elements $(x_i, x_i) \in [N] \times [N]$, all mapping to the same value under H . Denote by G the game $\text{MSE}^{\mathcal{S}, \mathcal{D}}$ with bit $b = 0$ fixed and inverted winning condition. Then $\Pr[\text{MSE}^{\mathcal{S}, \mathcal{D}} \mid b = 0] = 1 - \Pr[G]$, and by the law of total probability,

$$\Pr[G] \leq \Pr[G \mid H \in E] + \Pr[H \notin E] \leq \frac{\bar{M}^2}{M} + \binom{N}{\bar{M}} \cdot M^{1-2\bar{M}}.$$

Here, the bound on the first term follows from the fact that, since $H \in E$, both X_1 and X_2 contain at most $\bar{M} - 1$ elements, and therefore $|H(X_1 \times X_2)| < \bar{M}^2$. Since $y \in [M]$ is random, it will hit an element in this set with probability at most $\bar{M}^2/M$. For the second term, observe that if $H \notin E$, then there exist at least $\bar{M}$ elements $x_i \in [N]$ such that $H(x_i, x_i)$ collide, an event happening with probability

$$\frac{\binom{N}{\bar{M}} \cdot M \cdot M^{N^2 - \bar{M}}}{M^{N^2}} = \binom{N}{\bar{M}} \cdot M^{1-\bar{M}} \leq \left(\frac{eN}{\bar{M}} \right)^{\bar{M}} \cdot M^{1-\bar{M}} = M \cdot \left(\frac{eN}{M\bar{M}} \right)^{\bar{M}}.$$

Here, the inequality follows from Inequality 3 in Lemma 4. Collecting the terms above, we obtain (2). $\blacktriangleleft$

5 The Bounded Case

Having covered attacks against the MSE-notion in the previous section, we now turn to positive results. In this section, we prove that the monolithic construction **Mono** is a multi-source extractor in the setting where all parties have bounded-query access to the random oracle (i.e., the BBB corner in Figure 1). Observe that this result also implies security in the BBU setting by Proposition 14. Our main result, where sources and predictors can query the entire oracle, is presented in Section 6.

► **Theorem 19.** *Let $\ell, M, N, q_{\mathcal{S}}, q_{\mathcal{D}}, q_{\mathcal{P}} \in \mathbb{N}$ with $\ell \geq 2$ and $q_{\mathcal{P}} \geq (\ell - 1)q_{\mathcal{S}} + q_{\mathcal{D}}$, and $k \in \mathbb{R}_{\geq 0}$. Then **Mono** is a $(q_{\mathcal{P}}, 2^{-k}, q_{\mathcal{S}}, q_{\mathcal{D}}, \epsilon)$ -MSE, where ϵ is given in the right-hand side of (3). More precisely, for every tuple $\mathcal{S} = (\mathcal{S}_1, \dots, \mathcal{S}_{\ell})$ of sources, each of which is $(q_{\mathcal{P}}, 2^{-k})$ -unpredictable and makes at most $q_{\mathcal{S}}$ oracle calls, and every distinguisher $\mathcal{D}$ making at most $q_{\mathcal{D}}$ oracle calls, we have*

$$\text{Adv}_{\mathcal{S}, \mathcal{D}}^{\text{mse}} \leq \frac{\ell q_{\mathcal{S}} + q_{\mathcal{D}}}{2^k}. \quad (3)$$

Game G_1: $H \leftarrow \text{Fun}(N^\ell, M); Q \leftarrow \emptyset$ for $i = 1$ to ℓ do $(x_i, z_i) \leftarrow \mathcal{S}_i^O$ $y \leftarrow H(x);$ return $\mathcal{D}^O(y, z)$ Proc. $O(q)$: $Q \leftarrow Q \cup \{q\};$ return $H(q)$	Game G_2: $H \leftarrow \text{Fun}(N^\ell, M); Q \leftarrow \emptyset$ for $i = 1$ to ℓ do $(x_i, z_i) \leftarrow \mathcal{S}_i^O$ $y \leftarrow H(x); b' \leftarrow \mathcal{D}^O(y, z)$ if $(x \in Q)$ then return 1 return b'	Game G_3: $H \leftarrow \text{Fun}(N^\ell, M); Q \leftarrow \emptyset$ for $i = 1$ to ℓ do $(x_i, z_i) \leftarrow \mathcal{S}_i^O$ $y \leftarrow [M]; b' \leftarrow \mathcal{D}^O(y, z)$ if $(x \in Q)$ then return 1 return b'
Pred. $\mathcal{P}_j^H(z_j)$: $Q \leftarrow \emptyset; z_{\bar{j}} \leftarrow z_{\bar{j}}; (x_j, z_j) \leftarrow \mathcal{S}_j^O$ for $i \in [\ell] \setminus \{j, \bar{j}\}$ do $(x_i, z_i) \leftarrow \mathcal{S}_i^H$ $y \leftarrow [M]; b' \leftarrow \mathcal{D}^H(y, z)$ $q \leftarrow Q; \text{return } q_{\bar{j}}$	Pred. $\mathcal{Q}^H(z_1)$: $Q \leftarrow \emptyset; z_1 \leftarrow z_1$ for $i = 2$ to ℓ do $(x_i, z_i) \leftarrow \mathcal{S}_i^H$ $y \leftarrow [M]; b' \leftarrow \mathcal{D}^O(y, z)$ $q \leftarrow Q; \text{return } q_1$	

■ **Figure 5** Code of the intermediate games, and definition of the predictors $\mathcal{P}_j$ and $\mathcal{Q}$, from the proof of Theorem 19. Oracle O is the same throughout the figure.

Proof. Consider a tuple of sources $\mathcal{S} = (\mathcal{S}_1, \dots, \mathcal{S}_\ell)$ such that, for every $i \in [\ell]$, $\mathcal{S}_i$ is $(q_{\mathcal{P}}, 2^{-k})$ -unpredictable and makes at most $q_{\mathcal{S}}$ oracle calls, and a distinguisher $\mathcal{D}$ making at most $q_{\mathcal{D}}$ oracle calls. We prove bound (3) via a sequence of games, whose formal description can be found in Figure 5:

- G_0 is the MSE-game for Mono played by $(\mathcal{S}, \mathcal{D})$, with bit $b = 1$ fixed.
- G_1 proceeds as G_0 , but we use a set Q to track all the queries made by the sources $\mathcal{S}_i$ and the distinguisher $\mathcal{D}$. Clearly, G_0 and G_1 are indistinguishable, since the only difference is that G_1 records the queries made by the parties, and otherwise all distributions are the identical. Therefore, $\Pr[G_1] = \Pr[G_0]$.
- G_2 proceeds as G_1 , but we return 1 if any source $\mathcal{S}_i$ or distinguisher $\mathcal{D}$ query vector x (which is used when computing $y = H(x)$) to the random oracle H . Notice that G_2 returns 1 whenever G_1 does, so that $\Pr[G_2] \geq \Pr[G_1]$.
- G_3 proceeds as G_2 , but we no longer compute y as $H(x)$, and instead sample it at random. We claim that G_2 and G_3 are indistinguishable. Indeed, if $x \in Q$, both games return 1 by definition. If $x \notin Q$, then $H(x)$ is a random value that is unknown to all parties, so computing y as $y = H(x)$ is equivalent to sampling it at random. Thus, $\Pr[G_3] = \Pr[G_2]$.
- G_4 proceeds as G_3 , but we remove the if-statement added in G_2 . Since we no longer need to track the oracle queries of $\mathcal{S}_i$ and $\mathcal{D}$, we replace O with H , and do not initialize Q . The resulting game coincides with the MSE-game for Mono played by $(\mathcal{S}, \mathcal{D})$ with bit $b = 0$ fixed and inverted winning condition.

To study the difference in success probability between G_3 and G_4 , let Bad_0 be the event that $\mathcal{D}$ queries x while playing G_3 , and Bad_i the event that $\mathcal{S}_i$ queries x in the same game, for $i \in [\ell]$. Then notice that G_3 and G_4 are identical until $\text{Bad} := \bigvee_{i=0}^{\ell} \text{Bad}_i$, and by the fundamental lemma of game playing we therefore have

$$|\Pr[G_4] - \Pr[G_3]| = \Pr[\text{Bad}] \leq \Pr[\text{Bad}_0] + \sum_{i=1}^{\ell} \Pr[\text{Bad}_i].$$

We study these probabilities separately. For $j \in [\ell]$, let $\bar{j} := (j \bmod \ell) + 1$, and consider the predictor $\mathcal{P}_j$ playing the predictability game for $\mathcal{S}_{\bar{j}}$ defined in Figure 5. Notice that $\mathcal{P}_j$ makes at most $(\ell - 1)q_{\mathcal{S}} + q_{\mathcal{D}}$ oracle calls, which means that

$$2^{-k} \geq \Pr[\text{Pred}_{\mathcal{S}_{\bar{j}}}^{\mathcal{P}_j}] \geq \Pr[\text{Pred}_{\mathcal{S}_{\bar{j}}}^{\mathcal{P}_j} \mid \text{Bad}_j] \Pr[\text{Bad}_j] \geq \frac{1}{q_{\mathcal{S}}} \Pr[\text{Bad}_j].$$

From here we get $\Pr[\text{Bad}_j] \leq q_S \cdot 2^{-k}$ for every $j \in [\ell]$. Similarly, let $\mathcal{Q}$ be the predictor playing the predictability game for $\mathcal{S}_1$ defined in Figure 5. As before, $\mathcal{Q}$ queries H at most $(\ell - 1)q_S + q_D$ many times, and therefore $\Pr[\text{Bad}_0] \leq q_D \cdot 2^{-k}$.

Combining the estimates above we obtain

$$\text{Adv}_{\mathcal{S}, \mathcal{D}}^{\text{mse}} = \Pr[\mathbf{G}_0] - \Pr[\mathbf{G}_4] \leq |\Pr[\mathbf{G}_4] - \Pr[\mathbf{G}_3]| \leq \epsilon. \quad \blacktriangleleft$$

► **Remark 20.** Observe that Theorem 19 is false when $\ell = 1$. Indeed, we bound the probability of $\mathcal{S}_j$ querying $\mathbf{x}$ by constructing a predictor against *another* source in $\mathcal{S}$ (e.g., $\mathcal{S}_{j+1}$). When $\ell = 1$, there is no “other” source to do that, and this step in the proof breaks down. Also notice that it is actually impossible to prevent $\mathcal{S}_1$ from querying $\mathbf{x}$ in this case, because $\mathbf{x} = \mathbf{x}_1$ is known to $\mathcal{S}_1$.

More importantly, we can present an attack which relies on this fact: Consider the source $\mathcal{S}_1$ which samples $x \leftarrow [N]$, sets $z \leftarrow (H(x) \leq M/2)$ and returns (x, z) , and the distinguisher $\mathcal{D}$ who returns the bit $((y \leq M/2) = z)$. Then it is easy to see that $\mathcal{S}_1$ is (q_P, δ) -unpredictable for any q_P and $\delta = 1/\lfloor M/2 \rfloor \approx 2/M$, and yet $\text{Adv}_{\mathcal{S}, \mathcal{D}}^{\text{mse}} \leq 1 - \lfloor M/2 \rfloor / M \approx 1/2$ for $\mathcal{S} = (\mathcal{S}_1)$. The issue is that when $\ell = 1$, $\mathcal{S}_1$ knows the entire input of H and can therefore leak information about the output $H(x)$, which then helps $\mathcal{D}$.

6 The Unbounded Case

In this section, we prove our main positive result, showing that the monolithic construction **Mono** is a multi-source extractor in the setting where the distinguisher is the only party with bounded-query access to the random oracle (i.e., the UBU corner in Figure 1). In particular, each source can fully depend on the random oracle.

Following Dodis et al. [16], our approach is as follows: To show that the output of H on the values $\mathbf{x}_i$ returned by the ℓ sources is pseudorandom, we first use Yao’s equivalence between distinguishing and next-bit prediction to turn any distinguisher with advantage ϵ into a next-bit guesser $\mathcal{P}$ with advantage $\epsilon/\log M$ for some bit in $H(\mathbf{x})$, say, the l th bit. Two cases are now possible:

1. $\mathcal{P}$ queries $\mathbf{x}$ with “good” probability $\epsilon/(2 \log M)$. We can then turn $\mathcal{P}$ into an adversary $\mathcal{Q}$ in the UR game with advantage $\epsilon/(2 \log M)$, by letting $\mathcal{Q}$ inspect the queries of $\mathcal{P}$ and returning the one mapping to $H(\mathbf{x})$. The advantage of $\mathcal{Q}$ can be bounded by a compression argument similar to the one in [16]. However, special care needs to be taken because unlike in [16], the challenge $\mathbf{x}$ is not chosen uniformly and independently of H , but is the output of sources that have full oracle access. We deal with this issue by decomposing the high-entropy sources into convex combinations of flat sources on large sets, and then proceed similarly to the uniform case.
2. $\mathcal{P}$ queries $\mathbf{x}$ with probability less than $\epsilon/(2 \log M)$, while still having advantage $\epsilon/\log M$ in guessing the l th bit of $H(\mathbf{x})$. We can then turn $\mathcal{P}$ into a guesser $\mathcal{Q}$ that never queries H on $\mathbf{x}$ but still has advantage $\epsilon/(2 \log M)$ in guessing the l th bit of $H(\mathbf{x})$, by simply running $\mathcal{P}$ and giving up whenever $\mathcal{P}$ wants to query $\mathbf{x}$. Predictor $\mathcal{Q}$ can then be used in an incompressibility argument to ultimately bound ϵ . For this part we follow De et al. [13, Lemma 8.4], bearing in mind that the $\mathbf{x}_i$ are non-uniform and oracle-dependent.

Following the roadmap outlined above, we begin by stating our unrecoverability result and a preparatory lemma for our main theorem. In the interest of space, we present the proofs of these statements in the full version of the paper [12].

► **Theorem 21** (Unrecoverability). *Let $\ell, M, N, q_A \in \mathbb{N}$ and $k \in \mathbb{R}_{>0}$. Then Mono is $(N^\ell, 2^{-k}, N^\ell, q_A, \epsilon)$ -unrecoverable, where ϵ is given in the right-hand side of (4). More precisely, for every tuple $\mathcal{S} = (S_1, \dots, S_\ell)$ of sources, each of which is $(N^\ell, 2^{-k})$ -unpredictable and has unbounded oracle access, and every adversary $\mathcal{A}$ making at most q_A oracle calls, we have*

$$\text{Adv}_{\mathcal{S}, \mathcal{A}}^{\text{ur}} = \mathcal{O}\left(\ell^{\ell+1} \sqrt{\frac{q_A}{2^{k\ell}}} (\sigma + \ell N)\right). \quad (4)$$

Here, $\sigma \geq \sum_{i=1}^{\ell} \sigma_i$, where $\sigma_i \in \mathbb{N}$ is a bound on the length of the leakage z_i returned by S_i .

► **Lemma 22.** *Let $N, \ell, q_A, \sigma, S \in \mathbb{N}$ and $\epsilon \in \mathbb{R}_{>0}$, so that $S^\ell \geq 40$ and $q_A \leq S^\ell / \log S^\ell$. Let $\mathcal{A}$ be a deterministic algorithm that makes at most q_A queries to its oracle, takes an advice string z of length at most σ , and is not allowed to query its input. Let $O \subseteq \text{Fun}(N^\ell, \{0, 1\})$ be a set such that, for each $O \in O$, there exist $z \in \{0, 1\}^\sigma$ and sets $S_i \subseteq [N]$ with $|S_i| = S$ for all $i \in [\ell]$ such that*

$$\Pr_{\mathbf{x} \leftarrow S_i} [\mathcal{A}^O(\mathbf{x}, z) = O(\mathbf{x})] \geq \frac{1}{2} + \epsilon.$$

Then either $q_A > \epsilon S^\ell$, or there exists a randomized encoding scheme (Enc, Dec) for O such that, for every $O \in O$,

$$\Pr[\text{Dec}(\text{Enc}(O, r), r) = O] = \Omega\left(\frac{\epsilon}{q_A}\right),$$

and the length of each codeword is at most

$$\sigma + 2\ell S \log\left(\frac{eN}{S}\right) + N^\ell - \Omega\left(\frac{\epsilon^2 S^\ell}{q_A}\right) + \mathcal{O}(1).$$

Finally, we come to the main theorem of this section, establishing that the random oracle is a good multi-source extractor.

► **Theorem 23** (MSE). *Let $\ell, M, N, q_D \in \mathbb{N}$ and $k \in \mathbb{R}_{>0}$. Then Mono is $(N^\ell, 2^{-k}, N^\ell, q_D, \epsilon)$ -MSE, where ϵ is given in the right-hand side of (5). More precisely, for every tuple of sources $\mathcal{S} = (S_1, \dots, S_\ell)$, each of which is $(N^\ell, 2^{-k})$ -unpredictable and has unbounded access to its oracle, and every distinguisher $\mathcal{D}$ making at most q_D oracle calls, we have*

$$\text{Adv}_{\mathcal{S}, \mathcal{D}}^{\text{mse}} = \mathcal{O}\left(\ell \log M^{\ell+1} \sqrt{\frac{q_D}{2^{k\ell}}} (\sigma + \ell N)\right). \quad (5)$$

Here, $\sigma \geq \sum_{i=1}^{\ell} \sigma_i$, where $\sigma_i \in \mathbb{N}$ is a bound on the length of the leakage z_i returned by S_i .

Proof. Let $\epsilon := \text{Adv}_{\mathcal{S}, \mathcal{D}}^{\text{mse}}$. By Yao's equivalence principle [36], any q_D -query distinguisher can be converted into a predictor $\mathcal{P}$ with the same query complexity, which can predict the l th output bit of the oracle for some $l \in [\log M]$, that is,

$$\Pr[\text{nbPred}_{\mathcal{S}, l}^{\mathcal{P}}] \geq \frac{1}{2} + \epsilon',$$

where $\epsilon' := \epsilon / \log M$ and $\text{nbPred}_{\mathcal{S}, l}^{\mathcal{P}}$ is the game defined in Figure 6 (left).

If $\mathcal{P}$ queries $\mathbf{x}$ with probability at least $\epsilon' / (2 \log M)$, then Theorem 21 gives

$$\epsilon = \mathcal{O}\left(\ell \log M^{\ell+1} \sqrt{\frac{q_D}{2^{k\ell}}} (\sigma + \ell N)\right), \quad (6)$$

Game nbPred$_{\mathcal{S},j}^{\mathcal{P}}$: $H \leftarrow \text{Fun}(N^\ell, M)$ for $i = 1$ to ℓ do $(x_i, z_i) \leftarrow S_i^H$ $y \leftarrow H(\mathbf{x}); y'_j \leftarrow \mathcal{P}^H(y_1, \dots, y_{j-1}, z)$ return $(y_j = y'_j)$	Game outPred$_{\mathcal{S},j}^{\mathcal{Q}}$: $H \leftarrow \text{Fun}(N^\ell, M)$ for $i = 1$ to ℓ do $(x_i, z_i) \leftarrow S_i^H$ $y \leftarrow H_j(\mathbf{x}); z \leftarrow \mathcal{Q}_0^H; y' \leftarrow \mathcal{Q}_1^{H_j}(\mathbf{x}, z, z)$ return $(y = y')$
--	---

■ **Figure 6** Definition of the j th next-bit prediction game, and of the output prediction game, both for a tuple of sources $\mathcal{S}$. In the latter game, $\mathcal{Q}_1$ is not allowed to query its input $\mathbf{x}$ to its oracle, and H_j is the projection on the j th bit of H .

that is, our bound (5) is already satisfied in this case. So assume for the remainder of the proof that this is not the case. We then construct a (single-bit) predictor $\mathcal{Q} = (\mathcal{Q}_0, \mathcal{Q}_1)$ that tries to predict $H_l(\mathbf{x})$ (the l th output bit of $H(\mathbf{x})$) without querying its input. Algorithm $\mathcal{Q}_0$ returns a table containing, in lexicographical order of the inputs, all but the l th bit of the corresponding oracle outputs. This table is of size $N^\ell(\log M - 1)$. Algorithm $\mathcal{Q}_1$ is a $(q_{\mathcal{Q}} = q_{\mathcal{P}})$ -query predictor that runs $\mathcal{P}$ on $(H(\mathbf{x})_1, \dots, H(\mathbf{x})_{l-1}, z)$, and combines the output of $\mathcal{P}$ with the generated table z , such that

$$\Pr[\text{outPred}_{\mathcal{S},l}^{\mathcal{Q}}] \geq \frac{1}{2} + \epsilon',$$

where $\text{outPred}_{\mathcal{S},l}^{\mathcal{Q}}$ is the output prediction game defined in Figure 6 (right).

Call a pair $(\mathbf{O}, \mathbf{z}) \in \text{Fun}(N^\ell, M) \times \text{Supp}(\mathcal{S}_{1,2}^{\mathbf{O}}) \times \dots \times \text{Supp}(\mathcal{S}_{\ell,2}^{\mathbf{O}})$ (α, β) -good if:

1. $\Pr[\text{outPred}_{\mathcal{S},l}^{\mathcal{Q}} \mid H = \mathbf{O} \wedge \mathbf{z} = \mathbf{z}] \geq 1/2 + \alpha\epsilon'$, and
 2. For every $i \in [\ell]$ and predictor $\mathcal{P}$ with unbounded access to its oracle, it holds that $\Pr[\text{Pred}_{\mathcal{S}_i}^{\mathcal{P}} \mid H = \mathbf{O} \wedge \mathbf{z} = \mathbf{z}_i] \leq \beta \cdot 2^{-k}$.
- In other words, a pair $(\mathbf{O}, \mathbf{z})$ as above is (α, β) -bad if either of the following conditions apply:
1. Pair $(\mathbf{O}, \mathbf{z})$ is low-win: $(\mathcal{S}, \mathcal{Q})$ succeeds with probability at most $1/2 + \alpha\epsilon'$ when $(H, \mathbf{z})$ is fixed to $(\mathbf{O}, \mathbf{z})$, i.e., $\Pr[\text{outPred}_{\mathcal{S},l}^{\mathcal{Q}} \mid H = \mathbf{O} \wedge \mathbf{z} = \mathbf{z}] < 1/2 + \alpha\epsilon'$.
 2. Pair $(\mathbf{O}, \mathbf{z})$ is low-entropy: There exist $i \in [\ell]$ and a predictor $\mathcal{P}_{\mathbf{O}, \mathbf{z}}$ with unbounded oracle access such that $\Pr[\text{Pred}_{\mathcal{S}_i}^{\mathcal{P}_{\mathbf{O}, \mathbf{z}}} \mid H = \mathbf{O} \wedge \mathbf{z} = \mathbf{z}_i] > \beta \cdot 2^{-k}$.

Denote by W_α the set of low-win pairs, $E_{\beta,i}$ set of pairs that are low-entropy for the i th source, and $E_\beta := \bigcup_{i=1}^\ell E_{\beta,i}$. Let $BP_{\alpha,\beta} := W_\alpha \cup E_\beta$ and $GP_{\alpha,\beta}$ be the sets of (α, β) -bad and (α, β) -good pairs, respectively. We start by bounding $p_{\text{bp}} := \Pr[(H, \mathbf{z}) \in BP_{\alpha,\beta}]$. By the union bound we have

$$p_{\text{bp}} \leq \Pr[(H, \mathbf{z}) \in W_\alpha] + \sum_{i=1}^\ell \Pr[(H, \mathbf{z}) \in E_{\beta,i}].$$

To bound $p_{\text{lw}} := \Pr[(H, \mathbf{z}) \in W_\alpha]$, notice that

$$\begin{aligned} \frac{1}{2} + \epsilon' &\leq \Pr[\text{outPred}_{\mathcal{S},l}^{\mathcal{Q}} \mid (H, \mathbf{z}) \in W_\alpha] \Pr[(H, \mathbf{z}) \in W_\alpha] \\ &\quad + \Pr[\text{outPred}_{\mathcal{S},l}^{\mathcal{Q}} \mid (H, \mathbf{z}) \notin W_\alpha] \Pr[(H, \mathbf{z}) \notin W_\alpha] < \left(\frac{1}{2} + \alpha\epsilon'\right) \cdot 1 + 1 \cdot (1 - p_{\text{lw}}), \end{aligned}$$

from which we obtain $p_{\text{lw}} < 1 - (1 - \alpha)\epsilon'$. To bound $p_{\text{le},i} := \Pr[(H, \mathbf{z}) \in E_{\beta,i}]$, let $\mathcal{P}$ be the predictor that queries its oracle H in full and then runs $\mathcal{P}_{H, \hat{z}}$, where $\hat{z} := (0, \dots, z, \dots, 0)$, if $(H, \hat{z}) \in E_{\beta,i}$, and returns the constant 1 otherwise. For this choice of $\mathcal{P}$ we have

$$\begin{aligned} 2^{-k} &\geq \Pr[\text{Pred}_{\mathcal{S}_i}^{\mathcal{P}}] \geq \Pr[\text{Pred}_{\mathcal{S}_i}^{\mathcal{P}} \mid (H, \hat{z}) \in E_{\beta,i}] \Pr[(H, \hat{z}) \in E_{\beta,i}] \\ &= \Pr[\text{Pred}_{\mathcal{S}_i}^{\mathcal{P}_{H, \hat{z}}} \mid (H, \hat{z}) \in E_{\beta,i}] \cdot p_{\text{le},i} \geq \beta \cdot 2^{-k} \cdot p_{\text{le},i}, \end{aligned}$$

from which we get $p_{le,i} < 1/\beta$ for every $i \in [\ell]$. Collecting all terms, we obtain a bound $p_{bp} < 1 - (1 - \alpha)\epsilon' + \ell/\beta$, and thus $p_{gp} = 1 - p_{bp} > (1 - \alpha)\epsilon' - \ell/\beta$.

To ensure that $GP_{\alpha,\beta} \neq \emptyset$, we must pick α and β so that $(1 - \alpha)\epsilon' - \ell/\beta > 0$. In particular, we can choose $0 < \alpha < 1$ to be a constant, and $\beta > \ell/((1 - \alpha)\epsilon')$, i.e., $\beta = b\ell/((1 - \alpha)\epsilon')$ for some constant $b > 1$. By at most doubling β if necessary, we can choose β so that additionally $2^{k-\log \beta} \in \mathbb{N}$ (as required by Lemma 3).

Now let $G_{\alpha,\beta}$ be the set of (α, β) -good oracles, defined as the $\mathbf{O} \in \text{Fun}(N^\ell, M)$ such that $(\mathbf{O}, \mathbf{z}) \in GP_{\alpha,\beta}$ for some leakage vector $\mathbf{z}$. For every $\mathbf{O} \in G_{\alpha,\beta}$, fix any leakage vector $\mathbf{z}_{\mathbf{O}}$ such that $(\mathbf{O}, \mathbf{z}_{\mathbf{O}}) \in GP_{\alpha,\beta}$. Then

$$p_{go} := \Pr[H \in G_{\alpha,\beta}] \geq \Pr[(H, \mathbf{z}) \in GP_{\alpha,\beta}] = p_{gp},$$

meaning that

$$|G_{\alpha,\beta}| = p_{go} \cdot |\text{Fun}(N^\ell, M)| \geq \left((1 - \alpha)\epsilon' - \frac{\ell}{\beta} \right) \cdot M^{N^\ell} > 0.$$

For each (α, β) -good $\mathbf{O}$ and every $i \in [\ell]$, let $\mathcal{X}_{\mathbf{O},i}$ be the conditional distribution of $\mathcal{S}_i^{\mathbf{O}}$ on $[N]$ given that $z = z_{\mathbf{O},i}$. We can then use Lemma 3 to decompose each $\mathcal{X}_{\mathbf{O},i}$ into a convex combination of flat sources $\mathcal{X}_{\mathbf{O},i,j}$, each having support size *exactly* $S := 2^{k-\log \beta}$. By an averaging argument, there exists a flat source $\mathcal{X}_{\mathbf{O},i}^*$ within each convex combination such that

$$\Pr[\text{outPred}_{\mathcal{X}_{\mathbf{O},i}^*}^{\mathbf{O}} \mid H = \mathbf{O} \wedge \mathbf{z} = \mathbf{z}_{\mathbf{O}}] \geq \frac{1}{2} + \alpha\epsilon'.$$

Before coming to the actual compression argument, we distinguish four cases: If $S < 40$, then $2^k/\beta < 40$, and substituting for β and therein for ϵ' we obtain

$$\epsilon < \log M \cdot \frac{40b\ell}{(1 - \alpha)2^k} = \mathcal{O}\left(\frac{\ell \log M}{2^k}\right).$$

Similarly, if $q_{\mathcal{D}} > S^\ell / \log S^\ell$, then $q_{\mathcal{D}} > 2^{k\ell}/(\beta^\ell \ell(k - \log \beta)) > 2^{k\ell}/(\beta^\ell \ell k)$, so that after the appropriate substitutions we get

$$\epsilon < \sqrt[\ell]{\frac{(\log M)^\ell q_{\mathcal{D}} b^\ell \ell^{\ell+1} k}{2^{k\ell}(1 - \alpha)^\ell}} = \mathcal{O}\left(\ell \log M \sqrt[\ell]{\frac{q_{\mathcal{D}} \ell k}{2^{k\ell}}}\right) = \mathcal{O}\left(\ell \log M \sqrt[\ell]{\frac{q_{\mathcal{D}} \ell N}{2^{k\ell}}}\right).$$

Finally, if $q_{\mathcal{D}} > \epsilon' S^\ell$, then again

$$\epsilon < \sqrt[\ell+1]{\frac{(\log M)^{\ell+1} q_{\mathcal{D}} b^\ell \ell^\ell}{(1 - \alpha)^\ell 2^{k\ell}}} = \mathcal{O}\left(\ell \log M \sqrt[\ell+1]{\frac{q_{\mathcal{D}}}{2^{k\ell}}}\right).$$

This shows that in all cases above, ϵ already satisfies our bound (5), so for the remainder of the analysis assume that $S \geq 40$, that $q_{\mathcal{D}} \leq S^\ell / \log S^\ell$, and that $q_{\mathcal{D}} \leq \epsilon' S^\ell$. Note that this clears all the “trivial” cases from Lemma 22.

We now define our randomized encoding of $G_{\alpha,\beta}$ as follows:

Encoding: Given $\mathbf{O} \in G_{\alpha,\beta}$, we encode it as follows:

- Generate a table containing, in lexicographical order of $\mathbf{x}$, all but the l th bit of the oracle outputs $\mathbf{O}(\mathbf{x})$, taking $N^\ell(\log M - 1)$ bits;
- Apply the encoding algorithm from Lemma 22 to obtain a randomized encoding of $\mathbf{O}_j$, which has an encoding length of at most $\sigma + 2\ell S \log(eN/S) + N^\ell - \Omega((\alpha\epsilon/\log M)^2 \cdot S^\ell/q_A) + \mathcal{O}(1)$ many bits.

Decoding: Given a codeword as above, we decode it as follows:

- Initialize an empty table for $\mathcal{O}$;
- Extract the table of all but the l th bit of outputs from the codeword and use this to fill the oracle table;
- Follow the decoding algorithm in Lemma 22 to extract the l th bit of each oracle output.

From Lemma 22 we also know that the recovery probability of this encoding scheme is at least $\delta = \Omega(\alpha\epsilon/q_{\mathcal{D}} \log M)$.

Adding up the lengths of the different parts in the encoding, and applying Lemma 2 with δ as above gives

$$\begin{aligned} N^\ell(\log M - 1) + \sigma + 2\ell S \log\left(\frac{eN}{S}\right) + N^\ell - \Omega\left(\left(\frac{\alpha\epsilon}{\log M}\right)^2 \cdot \frac{S^\ell}{q_{\mathcal{D}}}\right) + \mathcal{O}(1) \\ \geq \log\left(\left((1-\alpha)\epsilon' - \frac{\ell}{\beta}\right) \cdot M^{N^\ell}\right) + \log\left(\frac{\alpha\epsilon}{q_{\mathcal{D}} \log M}\right) \\ = N^\ell \log M + \log\left((1-\alpha)\epsilon' - \frac{\ell}{\beta}\right) + \log\left(\frac{\alpha\epsilon}{q_{\mathcal{D}} \log M}\right). \end{aligned}$$

We now distinguish two more cases: If $\epsilon \leq q_{\mathcal{D}} \log M / (\alpha N^\ell)$ then again we are done, since we obtain a bound that verifies (5). Indeed,

$$\epsilon \leq \frac{q_{\mathcal{D}} \log M}{\alpha N^\ell} \leq \frac{q_{\mathcal{D}} \log M}{\alpha S^\ell} = \frac{q_{\mathcal{D}} (\log M)^{\ell+1} b^\ell \ell^\ell}{\alpha 2^{k\ell} (1-\alpha)^\ell \epsilon^\ell},$$

giving

$$\epsilon = \mathcal{O}\left(\ell \log M \sqrt[\ell+1]{\frac{q_{\mathcal{D}}}{2^{k\ell}}}\right).$$

If the above is not the case then, rearranging, we have

$$\begin{aligned} \Omega\left(\left(\frac{\alpha\epsilon}{\log M}\right)^2 \cdot \frac{S^\ell}{q_{\mathcal{D}}}\right) &\leq \sigma + 2\ell S \log\left(\frac{eN}{S}\right) + \log\left(\frac{q_{\mathcal{D}} \log M}{\alpha\epsilon}\right) \\ &\quad - \log\left((1-\alpha)\epsilon' - \frac{\ell}{\beta}\right) + \mathcal{O}(1) \\ &\leq \sigma + 2\ell N \log e + \mathcal{O}(\ell \log N) - \log\left((1-\alpha)\epsilon' - \frac{\ell}{\beta}\right) \\ &= \sigma + \mathcal{O}(\ell N) - \log\left((1-\alpha)\epsilon' - \frac{\ell}{\beta}\right) = \mathcal{O}(\sigma + \ell N) - \log\left((1-\alpha)\epsilon' - \frac{\ell}{\beta}\right), \end{aligned}$$

where the inequality uses the fact that $x \mapsto x \log(eN/x)$ has a maximum at $x = N$, where it takes the value $N \log e$. Now recall that $S = 2^k/\beta$ and $\beta = b\ell/(1-\alpha)\epsilon'$, giving

$$\begin{aligned} \Omega\left(\frac{\epsilon^2}{(\log M)^2}\right) &= \mathcal{O}\left(\frac{q_{\mathcal{D}}}{S^\ell}(\sigma + \ell N)\right) - \frac{q_{\mathcal{D}}}{S^\ell} \log\left((1-\alpha)\epsilon' - \frac{\ell}{\beta}\right) \\ &= \mathcal{O}\left(\frac{q_{\mathcal{D}} \beta^\ell}{2^{k\ell}}(\sigma + \ell N)\right) - \frac{q_{\mathcal{D}} \beta^\ell}{2^{k\ell}} \log\left((1-\alpha)\epsilon' - \frac{\ell}{\beta}\right) \\ &\leq \mathcal{O}\left(\frac{q_{\mathcal{D}} (b\ell)^\ell}{2^{k\ell} ((1-\alpha)\epsilon')^\ell}(\sigma + \ell N)\right) - \frac{q_{\mathcal{D}} (b\ell)^\ell}{2^{k\ell} ((1-\alpha)\epsilon')^\ell} \log\left((1-\alpha)\epsilon' - \frac{(1-\alpha)\epsilon'}{b}\right) \\ &\leq \mathcal{O}\left(\frac{q_{\mathcal{D}} (b\ell)^\ell}{2^{k\ell} ((1-\alpha)\epsilon')^\ell}(\sigma + \ell N)\right) - \frac{q_{\mathcal{D}} (b\ell)^\ell}{2^{k\ell} ((1-\alpha)\epsilon')^\ell} \log\left(\frac{(1-\alpha)\epsilon'(b-1)}{b}\right). \end{aligned}$$

Hence,

$$\Omega\left(\frac{\epsilon^{\ell+2}}{(\log M)^{\ell+2}}\right) \leq \mathcal{O}\left(\frac{q_{\mathcal{D}}(b\ell)^\ell}{2^{k\ell}(1-\alpha)^\ell}(\sigma + \ell N)\right) + \mathcal{O}\left(\frac{q_{\mathcal{D}}(b\ell)^\ell}{2^{k\ell}(1-\alpha)^\ell} \log\left(\frac{\log M}{\epsilon}\right)\right).$$

We now once more distinguish two cases: If $\log M/\epsilon > N^\ell$, then $\epsilon < \log M/N^\ell$, and again our bound (5) holds. Otherwise, we obtain

$$\begin{aligned} \Omega\left(\frac{\epsilon^{\ell+2}}{(\log M)^{\ell+2}}\right) &\leq \mathcal{O}\left(\frac{q_{\mathcal{D}}(b\ell)^\ell}{2^{k\ell}(1-\alpha)^\ell}(\sigma + \ell N)\right) + \mathcal{O}\left(\frac{q_{\mathcal{D}}(b\ell)^\ell}{2^{k\ell}(1-\alpha)^\ell} \log(N^\ell)\right) \\ &\leq \mathcal{O}\left(\frac{q_{\mathcal{D}}(b\ell)^\ell}{2^{k\ell}(1-\alpha)^\ell}(\sigma + \ell N)\right), \end{aligned}$$

which gives

$$\epsilon^{\ell+2} \leq \mathcal{O}\left(\frac{q_{\mathcal{D}}(\log M)^{\ell+2}(b\ell)^{\ell+2}}{2^{k\ell}(1-\alpha)^{\ell+2}}(\sigma + \ell N)\right),$$

and thus

$$\begin{aligned} \epsilon &\leq \mathcal{O}\left(\sqrt[\ell+2]{\frac{q_{\mathcal{D}}(\log M)^{\ell+2}(b\ell)^{\ell+2}}{2^{k\ell}(1-\alpha)^{\ell+2}}(\sigma + \ell N)}\right) \leq \mathcal{O}\left(\frac{b\ell \log M}{1-\alpha} \sqrt[\ell+2]{\frac{q_{\mathcal{D}}}{2^{k\ell}}(\sigma + \ell N)}\right) \\ &\leq \mathcal{O}\left(\ell \log M \sqrt[\ell+2]{\frac{q_{\mathcal{D}}}{2^{k\ell}}(\sigma + \ell N)}\right). \end{aligned}$$

This concludes the proof. $\blacktriangleleft$

► **Remark 24.** Compared to the work of Dodis and Oliveira [17], note that our MSE theorem allows extracting many more bits (akin to a PRG security result), which cannot hold in the statistical setting of [17]. This allows directly generating random bits from multiple entropy sources, without further processing, using a standard hash function, as opposed to ad-hoc constructions.

As a contribution of independent interest, we show in the full version of our paper [12] that Mono is one-way in the UBU setting with respect to split sources that fully depend on the random oracle. Note that the split structure is necessary for one-wayness, as otherwise the source can output a fixed point of the hash function. And indeed, the bound that we achieve is meaningful only for $\ell \geq 2$.

References

- 1 Divesh Aggarwal, Maciej Obremski, João L. Ribeiro, Luisa Siniscalchi, and Ivan Visconti. How to extract useful randomness from unreliable sources. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 343–372. Springer, Cham, May 2020. doi:10.1007/978-3-030-45721-1_13.
- 2 Boaz Barak, Russell Impagliazzo, and Avi Wigderson. Extracting randomness using few independent sources. In *45th FOCS*, pages 384–393. IEEE Computer Society Press, October 2004. doi:10.1109/FOCS.2004.29.
- 3 Salman Beigi, Omid Etesami, and Amin Gohari. Deterministic randomness extraction from generalized and distributed Santha–Vazirani sources. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *ICALP 2015, Part I*, volume 9134 of *LNCS*, pages 143–154. Springer, Berlin, Heidelberg, July 2015. doi:10.1007/978-3-662-47672-7_12.

- 4 Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 398–415. Springer, Berlin, Heidelberg, August 2013. doi:10.1007/978-3-642-40084-1_23.
- 5 Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993. doi:10.1145/168588.168596.
- 6 Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Berlin, Heidelberg, May / June 2006. doi:10.1007/11761679_25.
- 7 Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge-based pseudo-random number generators. In Stefan Mangard and François-Xavier Standaert, editors, *CHES 2010*, volume 6225 of *LNCS*, pages 33–47. Springer, Berlin, Heidelberg, August 2010. doi:10.1007/978-3-642-15031-9_3.
- 8 Eshan Chattopadhyay and David Zuckerman. Explicit two-source extractors and resilient functions. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 670–683. ACM Press, June 2016. doi:10.1145/2897518.2897528.
- 9 Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity (extended abstract). In *26th FOCS*, pages 429–442. IEEE Computer Society Press, October 1985. doi:10.1109/SFCS.1985.62.
- 10 Sandro Coretti, Yevgeniy Dodis, Siyao Guo, and John P. Steinberger. Random oracles and non-uniformity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 227–258. Springer, Cham, April / May 2018. doi:10.1007/978-3-319-78381-9_9.
- 11 Sandro Coretti, Yevgeniy Dodis, Harish Karthikeyan, and Stefano Tessaro. Seedless fruit is the sweetest: Random number generation, revisited. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 205–234. Springer, Cham, August 2019. doi:10.1007/978-3-030-26948-7_8.
- 12 Sandro Coretti, Pooya Farshim, Patrick Harasser, and Karl Southern. Multi-source randomness extraction and generation in the random-oracle model. Cryptology ePrint Archive, Report 2025/1258, 2025. URL: <https://eprint.iacr.org/2025/1258>.
- 13 Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and PRGs. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 649–665. Springer, Berlin, Heidelberg, August 2010. doi:10.1007/978-3-642-14623-7_35.
- 14 Yevgeniy Dodis, Ariel Elbaz, Roberto Oliveira, and Ran Raz. Improved randomness extraction from two independent sources. In Dana Ron, editor, *RANDOM 04*, volume 3122 of *LNCS*, pages 334–344. Springer, August 2004. doi:10.1007/978-3-540-27821-4_30.
- 15 Yevgeniy Dodis, Rosario Gennaro, Johan Håstad, Hugo Krawczyk, and Tal Rabin. Randomness extraction and key derivation using the CBC, cascade and HMAC modes. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 494–510. Springer, Berlin, Heidelberg, August 2004. doi:10.1007/978-3-540-28628-8_30.
- 16 Yevgeniy Dodis, Siyao Guo, and Jonathan Katz. Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 473–495. Springer, Cham, April / May 2017. doi:10.1007/978-3-319-56614-6_16.
- 17 Yevgeniy Dodis and Roberto Oliveira. On extracting private randomness over a public channel. In Amit Sahai, editor, *RANDOM 2003*, volume 2764 of *LNCS*, pages 252–263. Springer, August 2003. doi:10.1007/978-3-540-45198-3_22.

- 18 Yevgeniy Dodis, Shien Jin Ong, Manoj Prabhakaran, and Amit Sahai. On the (im)possibility of cryptography with imperfect randomness. In *45th FOCS*, pages 196–205. IEEE Computer Society Press, October 2004. doi:10.1109/FOCS.2004.44.
- 19 Yevgeniy Dodis, Thomas Ristenpart, and Salil P. Vadhan. Randomness condensers for efficiently samplable, seed-dependent sources. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 618–635. Springer, Berlin, Heidelberg, March 2012. doi:10.1007/978-3-642-28914-9_35.
- 20 Yevgeniy Dodis, Adi Shamir, Noah Stephens-Davidowitz, and Daniel Wichs. How to eat your entropy and have it too – Optimal recovery strategies for compromised RNGs. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 37–54. Springer, Berlin, Heidelberg, August 2014. doi:10.1007/978-3-662-44381-1_3.
- 21 Yevgeniy Dodis and Joel Spencer. On the (non)universality of the one-time pad. In *43rd FOCS*, pages 376–387. IEEE Computer Society Press, November 2002. doi:10.1109/SFCS.2002.1181962.
- 22 Yevgeniy Dodis, Vinod Vaikuntanathan, and Daniel Wichs. Extracting randomness from extractor-dependent sources. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 313–342. Springer, Cham, May 2020. doi:10.1007/978-3-030-45721-1_12.
- 23 Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO’86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987. doi:10.1007/3-540-47721-7_12.
- 24 Peter Gazi, Krzysztof Pietrzak, and Stefano Tessaro. The exact PRF security of truncation: Tight bounds for keyed sponges and truncated CBC. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 368–387. Springer, Berlin, Heidelberg, August 2015. doi:10.1007/978-3-662-47989-6_18.
- 25 Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *41st FOCS*, pages 305–313. IEEE Computer Society Press, November 2000. doi:10.1109/SFCS.2000.892119.
- 26 Intel Corporation. *Intel Digital Random Number Generator (DRNG) Software Implementation Guide*, 2019. URL: <https://www.intel.com/content/www/us/en/developer/articles/guide/intel-digital-random-number-generator-drng-software-implementation-guide.html>.
- 27 Xin Li. Three-source extractors for polylogarithmic min-entropy. In Venkatesan Guruswami, editor, *56th FOCS*, pages 863–882. IEEE Computer Society Press, October 2015. doi:10.1109/FOCS.2015.58.
- 28 James L. McInnes and Benny Pinkas. On the impossibility of private key cryptography with weakly random keys. In Alfred J. Menezes and Scott A. Vanstone, editors, *CRYPTO’90*, volume 537 of *LNCS*, pages 421–435. Springer, Berlin, Heidelberg, August 1991. doi:10.1007/3-540-38424-3_31.
- 29 Noam Nisan and David Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996. doi:10.1006/JCSS.1996.0004.
- 30 Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indistinguishability framework. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 487–506. Springer, Berlin, Heidelberg, May 2011. doi:10.1007/978-3-642-20465-4_27.
- 31 Miklos Santha and Umesh V. Vazirani. Generating quasi-random sequences from slightly-random sources (extended abstract). In *25th FOCS*, pages 434–440. IEEE Computer Society Press, October 1984. doi:10.1109/SFCS.1984.715945.
- 32 Thomas Shrimpton and R. Seth Terashima. A provable-security analysis of Intel’s secure key RNG. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 77–100. Springer, Berlin, Heidelberg, April 2015. doi:10.1007/978-3-662-46800-5_4.

- 33 Dominique Unruh. Random oracles and auxiliary input. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 205–223. Springer, Berlin, Heidelberg, August 2007. doi:10.1007/978-3-540-74143-5_12.
- 34 Salil P. Vadhan. *Pseudorandomness*, volume 7 of *Foundations and Trends in Theoretical Computer Science*. Now Publishers, Boston-Delft, 2012. doi:10.1561/04000000010.
- 35 Joanne Woodage and Dan Shumow. An analysis of NIST SP 800-90A. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 151–180. Springer, Cham, May 2019. doi:10.1007/978-3-030-17656-3_6.
- 36 Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd FOCS*, pages 80–91. IEEE Computer Society Press, November 1982. doi:10.1109/SFCS.1982.45.

New Results in Share Conversion, with Applications to Evolving Access Structures

Tamar Ben David  

Ariel University, Israel

Varun Narayanan  

University of California, Los Angeles, CA, USA

Olga Nissenbaum  

Ariel University, Israel

Anat Paskin-Cherniavsky  

Ariel University, Israel

Abstract

We say there is a share conversion from a secret-sharing scheme Π to another scheme Π' implementing the same access structure if each party can locally apply a deterministic function to their share to transform any valid secret-sharing under Π to a valid (but not necessarily random) secret-sharing under Π' of the same secret. If such a conversion exists, we say that $\Pi \geq \Pi'$. This notion was introduced by Cramer et al. (TCC'05), where they particularly proved that for any access structure, any linear secret-sharing scheme over a given field $\mathbb{F}$, has a conversion from a CNF scheme, and is convertible to a DNF scheme.

In this work, we initiate a systematic study of convertibility between secret-sharing schemes, and present a number of results with implications to the understanding of the convertibility landscape.

- In the context of linear schemes, we present two key theorems providing necessary conditions for convertibility, proved using linear-algebraic tools. It has several implications, such as the fact that Shamir secret-sharing scheme can be neither maximal or minimal. Another implication of it is that a scheme may be minimal if its share complexity is at least as high as that of DNF.
- Our second key result is a necessary condition for convertibility to CNF from a broad class of (not necessarily linear) schemes. This result is proved via information-theoretic techniques and implies non-maximality for schemes with share complexity smaller than that of CNF.

We also provide a condition which is both necessary and sufficient for the existence of a share conversion to some linear scheme. The condition is stated as a system of linear equations, such that a conversion exists if and only if a solution to the linear system exists. We note that the impossibility results for linear schemes may be viewed as identifying a subset of contradicting equations in the system.

Another contribution of our paper, is in defining and studying share conversion for evolving secret-sharing schemes. In such a schemes, recently introduced by Komargodski et al. (IEEE ToIT'18), the number of parties is not bounded apriori, and every party receives a share as it arrives, which never changes in the sequel. Our impossibility results have implications to the evolving setting as well. Interestingly, unlike in the standard setting, there is no maximum or minimum in a broad class of evolving schemes, even without any restriction on the share size.

Finally, we show that, generally, there is no conversion between additive schemes over different fields, even from CNF to DNF! However by relaxing from perfect to statistical security, it may be possible to convert, and exemplify this for (n, n) -threshold access structures.

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques; Security and privacy → Mathematical foundations of cryptography

Keywords and phrases secret sharing, linear secret sharing, evolving access structures, share conversion, feasibility

Digital Object Identifier 10.4230/LIPIcs.ITC.2025.11

Related Version *Full Version:* <https://eprint.iacr.org/2024/1781> [11]



© Tamar Ben David, Varun Narayanan, Olga Nissenbaum, and Anat Paskin-Cherniavsky;
licensed under Creative Commons License CC-BY 4.0
6th Conference on Information-Theoretic Cryptography (ITC 2025).
Editor: Niv Gilboa; Article No. 11; pp. 11:1–11:23



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

1 Introduction

Secret-sharing is a fundamental notion in cryptography. A secret-sharing scheme enables a dealer to distribute a secret among a set of parties so that any pre-specified subset of qualified parties can recover the secret while any other subset of parties remains oblivious to the secret. The monotone class of subsets of qualified parties constitutes the *access structure* realized by the secret-sharing scheme.

Secret-sharing is a building block for realizing several complex cryptographic tasks. Certain such tasks may require additional properties in the secret-sharing scheme – for instance, succinctness of the shares, or homomorphism and other algebraic properties. This suggests use cases where a protocol requires secret-sharing according to one scheme during one stage, and according to another scheme during another. This motivated non-interactive conversion between secret-sharing schemes, which was formalized in [9] by Cramer, Damgård, and Ishai as *share conversion*.

We say there is a share conversion from a secret-sharing scheme Π to another scheme Π' implementing the same access structure Γ if each party can locally apply a deterministic function to their share to transform any valid secret-sharing under Π to a valid (but not necessarily random) secret-sharing of the same secret under Π' . In full generality, share conversion may be defined from Π to Π' which implement different access structures $\Gamma \supseteq \Gamma'$, respectively. Moreover the secret under Π' after the transformation can be a pre-specified function of the secret under Π before transformation. In this work, we focus on the natural case where $\Gamma = \Gamma'$ and the above-mentioned function is identity.

In the sequel, we will say $\Pi \geq \Pi'$ if there is a share conversion from Π to Π' . This induces a partial ordering over secret-sharing schemes realizing any access structure Γ . Many important insights into the partial order $\geq$ of convertibility for *linear secret-sharing schemes* over a finite field were provided in [9]. Among other results, they proved that, for any access structure Γ and finite field $\mathbb{F}$, CNF-based secret-sharing scheme $CNF_{\Gamma, \mathbb{F}}$ is maximal, and DNF-based secret-sharing scheme $DNF_{\Gamma, \mathbb{F}}$ is minimal among the set of all linear secret-sharing schemes for Γ over $\mathbb{F}$. I.e., $CNF_{\Gamma, \mathbb{F}} \geq \Pi \geq DNF_{\Gamma, \mathbb{F}}$ for any linear secret-sharing scheme Π . Note, however, that the existence of additional minimal and maximal schemes is not ruled out in [9]. For certain access structures, specifically (2,3)-threshold, they demonstrated that certain linear schemes like Shamir secret-sharing scheme is not maximal, as it is not convertible to CNF. They also show that a limited class of linear secret-sharing schemes – the so called *replicated schemes*, that are similar in structure to CNF in the sense that the secret is defined as the sum of random elements, and every party gets a subset of them as its share, are not maximal for (k, n) -threshold access structures unless they have share complexity as high as that of CNF (see Section 3.3 in [9]).

In this paper, we initiate a systematic study of convertibility between secret-sharing schemes, and obtain new results in several directions:

- We develop new and easily checkable necessary conditions for share conversion between linear schemes implementing a given access structure. These necessary conditions are in the form of linear algebraic constraints on the *monotone span program (MSP)* corresponding to the linear schemes. Using these conditions, we are able to get a clearer view of the partial order induced by convertibility over the linear schemes.
- We develop a necessary and sufficient condition for a conversion between the linear schemes Π, Π' in form of a linear system decided by the MSP of Π and Π' which has a solution if and only if $\Pi \geq \Pi'$.

- Next, we address the broader problem of share conversions involving potentially non-linear secret-sharing schemes. To this end, we introduce the notion of *non-degenerate* secret-sharing schemes and establish a necessary condition for converting shares from general (potentially non-linear) schemes to non-degenerate ones. This condition has implications for share conversion to CNF and Shamir secret-sharing schemes, which are both non-degenerate.
- We apply our results to develop necessary conditions for conversion to the well-studied schemes, such as CNF, DNF, and Shamir secret-sharing schemes.
- The necessary conditions we develop also bear consequences for secret-sharing schemes for evolving setting, i.e. where the number of parties is not bounded, and the party gets its share when appears. We show that, for several interesting evolving access structures, there is no maximal or minimal scheme.
- We also initiate the study of share conversion between different fields. We show that, in a general case, there is no conversion between linear schemes over two different fields. To circumvent this, we propose a general approach of bounding the randomness domain in a source scheme. We build a leaky additive scheme over $\mathbb{Z}_p$ allowing conversion into $\mathbb{Z}_q$. As it's possible to see from our example, the proposed approach could result in a privacy leakage, which is often tolerable if small.

1.1 Overview of Our Results

In this section, we provide a brief exposition of our results, which we formally describe and prove in the subsequent sections.

Necessary conditions for conversion between linear schemes. A linear secret-sharing scheme Π over a field $\mathbb{F}$ implementing access structure Γ over n parties is characterized by a monotone span program described as a triple $(\mathbb{F}, M, \rho)$, where M is a matrix over $\mathbb{F}$ of dimension $m \times k$ and $\rho : [m] \rightarrow [n]$ defines the set of M 's rows corresponding to a certain party [16].

To share a secret $s \in \mathbb{F}$, the dealer samples a vector $\mathbf{r} \in \mathbb{F}^k$ such that its first coordinate is s , and computes $\mathbf{v} = M \cdot \mathbf{r}$. Then, the i 'th share in Π is $\text{sh}_i = \mathbf{v}[\rho^{-1}(i)]$, which is the sub-vector containing entries in the coordinates $\rho^{-1}(i)$. A qualified set of parties T can recover the secret using a reconstruction function $\alpha \in \mathbb{F}^{|\rho^{-1}(T)|}$ such that

$$(\alpha)^\top \cdot \mathbf{v}_T = (\alpha)^\top \cdot M_T \cdot \mathbf{r} = \mathbf{r}[1] = s.$$

Here, $\mathbf{v}_T = \mathbf{v}[\rho^{-1}(T)]$ and $M_T = M[\rho^{-1}(T), \cdot]$, i.e., the rows of M corresponding to the coordinates $\rho^{-1}(T)$ (under some pre-specified ordering).

One of the key tools in our paper is a necessary condition for conversion between a pair of linear schemes. Informally, it states that conversion is impossible, if the schemes satisfy certain linear-algebraic conditions.

► **Theorem 1** (Necessary condition for conversion between linear schemes – Informal). *There is no share conversion from a linear secret-sharing scheme described by MSP $(\mathbb{F}, M \in \mathbb{F}^{m \times k}, \rho)$ to another linear scheme $(\mathbb{F}, M' \in \mathbb{F}^{m' \times k'}, \rho')$ both realizing Γ , if there are sets of parties T , T^* and party $h \notin T \cup T^*$ such that $T \cup h \in \Gamma$, and no strict subsets of $T \cup h$ is qualified, and, when α and α' are reconstruction functions for $T \cup h$ in M and M' , respectively,*

1. $(\alpha'_h)^\top \cdot M'_h \in \text{Rowspan}(M_{T^*})$.
2. $(\alpha_h)^\top \cdot M_h \notin (\text{Rowspan}(M_T) \cap \text{Rowspan}(M_h)) + (\text{Rowspan}(M_{T^*}) \cap \text{Rowspan}(M_h))$.

This theorem is formally stated as Theorem 27 in the technical section. We demonstrate the power of this seemingly abstract necessary condition by providing a concrete application to well studied secret-sharing schemes, and its application to share conversions for evolving access structures.

We exploit the fact that the share conversion function is local and the secret is preserved during the share conversion. We can reach a contradiction if it is possible to produce a pair of fooling instances of sharing under the source scheme that result in shares after conversion that do not respect the dependencies present among the shares in the target distribution. This proof is a vast generalization of the proof of a result in [9] that showed that in $(2,3)$ -Shamir is not convertible to CNF, so that it applies to conversion between any pair of linear schemes. Specifically, it follows that any Shamir secret-sharing scheme is not convertible to CNF.

In Theorem 30 we prove a different type of necessary condition for convertibility between linear schemes. As a corollary from Theorem 1 and Theorem 30, we prove that *DNF* is not convertible to any linear scheme with lower share complexity. In particular, it is not convertible to Shamir, which proves the non-minimality of the latter.

Both Theorem 30 and Theorem 1 are based on a key technical result – Lemma 25, which involves a single minterm T , while the theorems refer to at least 2 minterms to establish a conversion does not exist. We believe it is a useful conceptual simplification towards understanding convertability.

Convertibility characterization for linear schemes. We devise a characterization of convertibility between linear schemes by solvability of a certain system of linear equations $\mathcal{L}_{\Pi,\Pi'}$ which we provide in Section 5.

► **Theorem 2 (Theorem 35, informal).** *There is a conversion from a linear scheme Π to a linear scheme Π' realizing the same access structure over the same field if and only if the linear system $\mathcal{L}_{\Pi,\Pi'}$ has a solution.*

A solution of the system encodes a conversion in a straightforward (although redundant) way. The high level idea is to solve for variables $X_{\mathbf{r},i,j}$, where $X_{\mathbf{r},i,j}$ represents the j 'th share of p_i , when converting from a sharing based on randomness $\mathbf{r}$ in Π (as is sometimes useful, here $\mathbf{r}$ is assumed to include s). We note that the impossibility in Theorem 1 may be viewed as identifying a subset of contradicting equations, so that a solution does not exist.

Share conversion from non-linear schemes. In a prior work [9], CNF was proved to be maximal among linear schemes over the same field. In this work, we prove new necessary conditions for share conversion from general (potentially non-linear) schemes to CNF secret-sharing.

For this, we introduce the notion of *non-degenerate* secret-sharing schemes. A secret-sharing scheme Π is non-degenerate if any sub-scheme of Π is equivalent to Π . Here, a secret-sharing scheme Π' is a sub-scheme of Π if the secret domains of both schemes are identical, and every valid secret-sharing under Π' is also a valid secret-sharing under Π . A formal definition of non-degenerate secret-sharing schemes is provided in Definition 36.

To drive down this subtle notion, we will demonstrate that a 2-out-of-3 DNF secret-sharing scheme for secret domain $\{0,1\}$ is *not* non-degenerate. In 2-out-of-3 DNF secret-sharing of a secret $s \in \{0,1\}$, the shares of the 3 parties are $(r_{1,2}, s \oplus r_{3,1})$, $(s \oplus r_{1,2}, r_{2,3})$ and $(s \oplus r_{2,3}, r_{3,1})$, respectively, where $r_{1,2}$, $r_{2,3}$ and $r_{3,1}$ are uniform and independent bits. Consider a modified secret-sharing scheme in which the secret s is shared among the three parties exactly as in 2-out-of-3 DNF secret-sharing except that $r_{1,2}$, $r_{2,3}$ and $r_{3,1}$ are uniform and only pairwise

independent bits. It is easy to see that the latter is a 2-out-of-3 secret-sharing which is also a sub-scheme of the former. However, they are not clearly not equivalent: the former uses more randomness; hence, 2-out-of-3 DNF is not non-degenerate.

On the other hand, CNF secret-sharing scheme for arbitrary access structures over a group $\mathbb{G}$ for secret domain $\mathbb{G}$, and Shamir secret-sharing scheme are non-degenerate as established in Lemma 38 and Lemma 39, respectively. Lemma 38 is proved by considering the correlation of the shares induced by picking a secret $s \in \mathbb{G}$ at random and secret-sharing it using any secret-sharing scheme Π that is a sub-scheme of the given CNF scheme. Appealing to correctness and privacy of Π , an information theoretic argument is used to show that the entropy of each share in this correlation is the same as that in the correlation obtained by secret-sharing a random secret s using the CNF scheme. Since the secret domain is the same as the group over which CNF is defined, this further implies in a straightforward way that Π coincides with the CNF scheme, establishing its non-degeneracy.

By appealing to non-degeneracy of CNF, and using standard entropy lower bounds, we show that share conversion to CNF scheme is possible only if the share size under the source scheme is at least as large as that in the CNF scheme. The following result is formally stated in Theorem 40.

► **Theorem 3** (Extended maximality of CNF). *Let Π be a secret-sharing scheme realizing an n -party access structure Γ with secret domain $\mathbb{G}$ – a finite group. There is a share conversion from Π to CNF over $\mathbb{G}$ realizing Γ only if, for each $i \in [n]$, size of the share i in Π is at least $\log |\mathbb{G}| \cdot |\{F \in \mathcal{F} \text{ s.t. } i \notin F\}|$, where $\mathcal{F}$ is the set of all maximal forbidden sets associated with Γ .*

We note that Theorem 1 also implies results of non-maximality by impossibility of conversion to CNF for certain linear schemes Π with low share complexity. These results are mostly subsumed by Theorem 3, both because it does not restrict Π to be linear, and in terms of share size.

Share conversion for evolving secret-sharing. Komargodski et al. [17] defined evolving secret-sharing schemes where the unbounded number of parties arriving one after another, obtain their shares of secret. The previously qualified sets remain qualified, and shares of parties are not refreshed as new parties come, but each newcomer is provided a (potentially) progressively larger share. An evolving access structure is an infinite monotone class of qualified subsets of $\mathbb{N}$.

We initiate a study of share conversion for evolving secret-sharing, starting with formal extension of the notion of share conversion, MSP and linearity to the evolving setting. Then, we apply the theory we develop for proving impossibility of share conversion in the standard setting to the evolving setting. In particular, some of our results apply to *evolving linear secret-sharing schemes*, which have been previously considered in the literature, but never explicitly defined.

We address the problem of maximal and minimal secret-sharing schemes for evolving access structures, and show that for several broad classes of evolving schemes there is no maximal and minimal scheme. In Theorem 46 we formally state and prove the following result.

► **Theorem 4** (No evolving maximal scheme – Informal). *For any non-trivial evolving access structure, there exists no maximal secret-sharing scheme for one-bit secrets.*

By a non-trivial evolving access structure (See Definition 45), we mean one that does not devolve into a finite secret-sharing scheme among the first n parties (for some n) with the remaining parties either being not part of the qualified set or are required to simply receive the secret.

In the other direction, we obtain a slightly weaker result, showing there is no minimal linear scheme for certain access structures. This is formally stated and proved as Theorem 44.

► **Theorem 5** (No evolving minimal scheme – Informal). *For a certain broad class of evolving access structures Γ , and for the finite field $\mathbb{F}_2$, there is no minimal linear evolving scheme for any Γ in the class.*

Conversion between different fields. The bit simultaneously shared in two different fields, is called *dBit*, and is an important primitive for many applications, such as [2, 6, 8, 12, 13, 19, 20, 24]. There exist bit share conversion protocols, here we point out only few of them, such as proposed in [6, 7, 10]. It is natural to raise the question if such a conversion can be done locally.

Generalizing our impossibility result for linear schemes over the same field, we prove the inconvertibility of the maximal *CNF* scheme over $\mathbb{Z}_p$ to any other linear scheme over $\mathbb{Z}_q$ for primes $p \neq q$ with the same secret domain $\{0, 1\}$.

We complement this result with a relaxed form of conversion between additive schemes for certain pairs of distinct primes for 1-bit shares. The relaxation allows for a privacy leakage (over the scheme randomness). It may be interesting to further explore share conversions with a small correctness errors. These may suffice for many applications, while potentially extending the set of convertible secret sharing scheme pairs (denote $\Pi \geq_\epsilon \Pi'$).

1.2 Future work

Our work leaves several fascinating questions open. The main question is to obtain a simpler characterization of convertibility between linear schemes. As a first step, identify pairs of linear schemes Π, Π' over the same field where Π is not convertible to Π' , which is not implied by Theorem 27 or Theorem 30. It may be particularly interesting to find a different type of conflicting requirements in the linear system in Section 5, thereby better understanding the easier linear case, which was also studied in the original paper on share conversion [9]. Another concrete question is to characterize the minimal and maximal schemes for various access structures (in other words, those convertible to CNF, or from DNF). As the linear systems introduced in Section 5 work also for non-linear source schemes, it could be also interesting to explore convertibility from such schemes to linear ones. This would require new techniques not based on theorems as above, that both rely on linearity of Π as well.

In evolving setting, proving impossibility results is potentially easier. In our context, it could be interesting to understand whether minimal and/or maximal schemes exist for access structures for which we have not resolved this question.

Finally, it is interesting to find new non-trivial examples of conversions which *are* possible. As an extension, it is interesting to study the direction of converting from a modified subset of a scheme Π where part of the randomness is removed, as we do for a modified version of additive over $\mathbb{Z}_p$ to $\mathbb{Z}_q$, and the incurred privacy losses. The motivation here is that some properties of the original Π may be preserved by such a transformation, which may suffice for certain applications.

2 Prior Work

Share conversion. Cramer et al. [9] first defined share conversion for secret-sharing schemes as a way for converting shares of a secret in one scheme into shares of the same secret in a different scheme using only local computation and no communication between parties. Referring to a conversion between schemes realizing the same access structure and defined over the same field, they showed that CNF can be converted to any linear scheme, and any linear scheme can be converted to DNF. Furthermore, they put forward an application of share conversion to improving efficiency of multiparty computation (MPC). Beimel et al. [5] use generalized share conversion including non-identity relation between secrets from $(2, 3)$ CNF to $(3, 3)$ additive secret-sharing over different groups to 3-party private information retrieval (PIR). In fact, they observe that certain share conversions are implicit in state-of-the-art 3-party PIR constructions from the literature, and devise another conversions along these lines that induces an improved PIR construction. They also put forward certain impossibility results for certain PIR induced conversions. The following papers [21, 22] show additional positive results for potential conversions for 3-party schemes from the PIR-induced family.

Evolving secret-sharing. Komargodski et al. [17] defined evolving secret-sharing schemes for a case that the number of parties is unbounded, parties are only added as they arrive one after the other, and previously qualified sets remain qualified. They constructed the following evolving linear secret-sharing schemes: (1) a scheme for every evolving access structure, such that, the share size of the t^{th} party is 2^{t-1} ; (2) a k -threshold schemes in which the size of the share of party p_t is $O(k \log t)$; (3) an undirected st-connectivity schemes in which the share of each party is a bit.

A natural generalization of an evolving threshold access structure is to allow the threshold to depend on the index of the arriving party. Komargodski and Paskin-Cherniavsky [18] showed that any dynamic-threshold access can be realized with an evolving linear secret-sharing scheme in which the size of the share of party p_t is $O(t^4 \cdot \log t)$. Infinite decision trees were used in [17, 18] to construct evolving secret-sharing schemes. Alon et al. [1] define formally this model. They showed how to construct evolving secret-sharing schemes for generalized infinite decision trees. We use this construction in our work.

Peter in [23] defined evolving conditional disclosure of secrets (CDS), where the number of parties is unbounded, and parties arrive sequentially. Each party holds a private input, and when arrives, it sends a random message to a referee. In turn, at any stage of the protocol, the referee should be able to reconstruct a secret string, held by all the parties, from the messages it gets, if and only if the inputs of the parties that arrived satisfy some condition.

3 Preliminaries

In this section, we present the necessary notation and formal definitions of secret-sharing schemes and evolving secret-sharing schemes.

Notation. For $n \in \mathbb{N}$ by $[n]$ we denote the set $\{1, 2, \dots, n\}$. We denote by $\log$ the logarithmic function with base 2. Vectors are denoted by bold letters (e.g., $\mathbf{r}$). For matrices M, M' with the same number of columns we denote by $[M; M']$ the concatenation of matrix M' below M . For matrices M, M' with the same number of rows, $[M|M']$ is the concatenation of M' to the right for M . By $\text{Rowspan}(M)$ we denote the set of all vectors spanned by rows of M .

For a set of parties $\mathcal{P} = \{p_1, \dots, p_n\}$, when it is clear from the context, we often abuse notation replacing parties by their indexes from $[n]$. When we refer to a subset of parties $\{p_{i_1}, p_{i_2}, \dots, p_{i_t}\}$, we assume that $i_1 < i_2 < \dots < i_t$.

We will need the following well known fact in linear algebra.

► **Fact 6.** *Let $\mathbb{F}$ be a field, and $A \in \mathbb{F}^{n \times k}$, $\mathbf{v} \in \mathbb{F}^{1 \times k}$ such that $\mathbf{v} \notin \text{Rowspan}(A)$. Then there exists a solution $\mathbf{r}$ to the linear system $[\mathbf{v}|A]\mathbf{r} = (1, 0, \dots, 0)$.*

Finally, we sometimes abuse notation, and use a linear subspace $L = \mathbb{F}^k$ as a matrix consisting of some basis of L as its rows, without explicitly stating so.

3.1 Secret-Sharing

We start by defining (perfect) secret-sharing schemes for a finite set of parties.

► **Definition 7 (Access Structures).** *Let $\mathcal{P} = \{p_1, \dots, p_n\}$ be a set of parties. A collection $\Gamma \subseteq 2^{\{p_1, \dots, p_n\}}$ is monotone if $B \in \Gamma$ and $B \subseteq C$ imply that $C \in \Gamma$. An access structure $\Gamma \subseteq 2^{\{p_1, \dots, p_n\}}$ is a monotone collection of non-empty sets. Sets in Γ are called authorized, and sets not in Γ are called unauthorized. We will represent an n -party access structure by a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, where an input (i.e., a string) $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n) \in \{0, 1\}^n$ represents the set $A_\sigma = \{p_i : i \in [n], \sigma_i = 1\}$, and $f(\sigma) = 1$ if and only if $A \in \Gamma$. We will also call f an access structure.*

In a monotone access structure, the set $A \in \Gamma$ is called a minterm if there is no $B \subset A$ such that $B \in \Gamma$. The set $A \notin \Gamma$ is called a maxterm if for all $p_i \notin A$ it holds that $A \cup \{p_i\} \in \Gamma$.

The most basic and well-known access structure is the threshold access structure:

► **Definition 8 (Threshold Access Structures).** *Let $1 \leq k \leq n$. A k -out-of- n threshold access structure Γ over a set of parties $\mathcal{P} = \{p_1, \dots, p_n\}$ is the access structure containing all subsets of size at least k , that is, $\Gamma = \{A \subseteq \mathcal{P} : |A| \geq k\}$.*

A secret-sharing scheme defines a way to distribute shares to parties. Such a scheme is said to realize an access structure Γ if the shares held by any authorized set of parties (i.e., a set in the access structure) can be used to reconstruct the secret, and the shares held by any unauthorized set of parties reveal nothing about the secret. The formal definition follows.

► **Definition 9 (Secret-Sharing Schemes).** *A secret-sharing scheme Π over a set of parties $\mathcal{P} = \{p_1, \dots, p_n\}$ with domain of secrets S and domain of random strings R is a mapping from $S \times R$ to a set of n -tuples $S_1 \times S_2 \times \dots \times S_n$ (the set S_j is called the domain of shares of p_j). A dealer distributes a secret $s \in S$ according to Π by first sampling a random string $r \in R$ with uniform distribution, computing a vector of shares $\Pi(s; r) = (\text{sh}_1, \dots, \text{sh}_n)$, and privately communicating each share sh_j to party p_j . For a set $A \subseteq \{p_1, \dots, p_n\}$, we denote $\Pi_A(s; r)$ as the restriction of $\Pi(s; r)$ to its A -entries (i.e., the shares of the parties in A).*

A secret-sharing scheme Π with domain of secrets S realizes an access structure Γ if the following two requirements hold:

Correctness. *The secret s can be reconstructed by any authorized set of parties. That is, for any authorized set $B = \{p_{i_1}, \dots, p_{i_{|B|}}\} \in \Gamma$, there exists a reconstruction function $\text{Recon}_B : S_{i_1} \times \dots \times S_{i_{|B|}} \rightarrow S$ such that for every secret $s \in S$ and every random string $r \in R$, it holds that $\text{Recon}_B(\Pi_B(s; r)) = s$.*

Security. *Every unauthorized set cannot learn anything about the secret from its shares. Formally, for any set $T \notin \Gamma$, every two secrets $s_1, s_2 \in S$, and every possible vector of shares $\langle \text{sh}_j \rangle_{p_j \in T}$, $\Pr[\Pi_T(s_1; r) = \langle \text{sh}_j \rangle_{p_j \in T}] = \Pr[\Pi_T(s_2; r) = \langle \text{sh}_j \rangle_{p_j \in T}]$, where the probability is over the choice of r from R with uniform distribution.*

The size of the share of party p_j is defined as $\log |S_j|$ and the size of the shares of Π as $\max_{1 \leq j \leq n} \log |S_j|$. The total share size of Π is defined as $\sum_{j=1}^n \log |S_j|$.

Next we give some widely known secret-sharing schemes.

► **Definition 10** (Additive Secret-Sharing Scheme [15]). *In the additive secret-sharing scheme $ADD_{\mathbb{F},n}$ over $\mathbb{F}$, shares $sh_1, \dots, sh_n$ are sampled uniformly at random from $\mathbb{F}$ on the condition that $s = \sum_{i=1}^n sh_i$, and $\Gamma = \{\mathcal{P}\}$.*

► **Definition 11** (Shamir Secret-Sharing Scheme [25]). *In the (n, k) -Shamir secret-sharing scheme over $\mathbb{F}$ realizing k -out-of- n threshold access structure Γ , the dealer sets a polynomial $p(x) = s + r_1x + \dots + r_{k-1}x^{k-1}$ by uniformly random sampling of $r_j \leftarrow \mathbb{F}$ for $j \in [k-1]$. The share of p_i for $i \in [n]$ is set as $sh_i = p(i)$.*

The properties of Shamir's scheme over $\mathbb{F}_{2^m}$ for an appropriate $m \in \mathbb{N}$ are summarized in the next theorem.

► **Theorem 12** (Shamir [25]). *For every $n \in \mathbb{N}$, and $k \in [n]$, there is a secret-sharing scheme for secrets of size ℓ (i.e., the domain of secrets is $S = \{0, 1\}^\ell$) realizing the k -out-of- n threshold access structure, in which the share size is $\max\{\ell, \lceil \log(n+1) \rceil\}$. Moreover, the shares of the scheme are elements of the field $\mathbb{F}_{2^{\ell + \log n}}$.*

Next two schemes realize any monotone access structure. A replicated secret-sharing scheme [14] is also known as a CNF secret-sharing scheme [15].

► **Definition 13** (Replicated Secret-Sharing Schemes [14]). *Let $\Gamma \subseteq 2^{[n]}$ be a (monotone) access structure, and let $\mathcal{T}$ is the set of all maxterms of Γ . The CNF secret-sharing schemes for Γ over $\mathbb{F}$, denoted $CNF_{\Gamma, \mathbb{F}}$, proceeds as follows. A secret $s \in S$ is shared in $ADD_{\mathbb{F}, |\mathcal{T}|}$, where each share r_T is labelled by a different set $T \in \mathcal{T}$. Then, the dealer distributes to each party p_j all shares r_T such that $j \notin T$, that is, $sh_j = (r_T)_{j \notin T}$. For correctness, since Γ is monotone, a qualified set $Q \in \Gamma$ cannot be contained in any unqualified set, hence, members of Q jointly view all shares r_T and can thus reconstruct the secret s . For privacy, the parties of every maxterm $T \in \mathcal{T}$ jointly miss exactly one additive share r_T , hence parties of any unqualified set miss at least one share.*

► **Definition 14** (DNF Secret-Sharing Scheme [15]). *In DNF secret-sharing schemes, denoted $DNF_{\Gamma, \mathbb{F}}$, the secret s is additively shared between the parties of each minterm, where each additive sharing uses independent randomness.*

More secret-sharing schemes can be defined using the notion of a monotone span program (MSP). We bring the definition of MSP below.

► **Definition 15** (Monotone Span Program [16]). *A monotone span program is a triple $\mathcal{M} = (\mathbb{F}, M, \rho)$, where $\mathbb{F}$ is a field, M is an $m \times k$ matrix over $\mathbb{F}$, and a mapping $\rho : [m] \rightarrow [n]$ labels each row of M by a party's index. The size of $\mathcal{M}$ is the number of rows of M (i.e., m).*

Next we give some notation which simplifies addressing to sets and operations of MSP. Let $M \in \mathbb{F}^{m \times k}$ be a matrix, and $A \subseteq [m]$. We denote by $M[A, \cdot]$ the $|A| \times k$ dimensional submatrix that restricts M to the rows labeled by $i \in A$. Hence, for an MSP $(\mathbb{F}, M \in \mathbb{F}^{m \times k}, \rho)$ describing an n -party linear secret-sharing scheme, and $h \in [n]$, $M[\rho^{-1}(h), \cdot]$ denotes the submatrix induced by rows of M corresponding to shares of party h . For any $S \subseteq [n]$, for brevity, we will refer to $M[\rho^{-1}(S), \cdot]$ by M_S , and when $S = \{h\}$ for some $h \in [n]$, we will further simplify the notation by referring to $M_{\{h\}}$ as M_h . Similarly, for a column vector $\alpha \in \mathbb{F}^m$, and set $A \subseteq [m]$, we denote by $\alpha[A]$ the sub-vector of α labeled by $i \in A$, and for the subset of parties $S \subseteq [n]$ we let $\alpha_S = \alpha[\rho^{-1}(S)]$, and $\alpha_{\{h\}} = \alpha_h$.

► **Definition 16** (Access structure accepted by MSP [16]). *We say that MSP $\mathcal{M}$ accepts $B \subseteq [n]$ if the rows of M_B span the vector $\mathbf{e}_1 = (1, 0, \dots, 0)$, called a target vector.¹ We say that $\mathcal{M}$ accepts an access structure Γ if $\mathcal{M}$ accepts a set B if and only if $B \in \Gamma$.*

A monotone span program implies a so called *linear secret-sharing scheme* for an access structure containing all the sets accepted by the program. Essentially, a dealer gives each party the rows of matrix M assigned to it, multiplied by the randomness vector.

▷ **Claim 17** ([4]). Let $\mathcal{M} = (\mathbb{F}, M, \rho)$ be a MSP accepting an access structure Γ , where $\mathbb{F}$ is a finite field and for every $j \in [n]$ there are a_j rows of M labeled by p_j . Then, there is a linear secret-sharing scheme realizing Γ for $S = \mathbb{F}$ such that the share of party p_j is a vector in $\mathbb{F}^{a_j}$ with the information equal to $\max_{1 \leq j \leq n} a_j$.

3.2 Evolving Secret-Sharing Schemes

In an evolving secret-sharing scheme, defined by [17], the number of parties is unbounded. Parties arrive one after the other; when a party p_t arrives the dealer gives it a share. The dealer cannot update the share later and does not know how many parties will arrive after party p_t . Thus, we measure the share size of p_t as a function of t . We start by defining an evolving access structure, which specifies the authorized sets. The number of parties in an evolving access structure is infinite, however every authorized set is finite.

► **Definition 18** (Evolving Access Structures). *Let $\mathcal{P} = \{p_i\}_{i \in \mathbb{N}}$ be an infinite set of parties. A collection of finite sets $\Gamma \subseteq 2^{\mathcal{P}}$ is an evolving access structure if for every $t \in \mathbb{N}$ the collections $\Gamma^t \triangleq \Gamma \cap 2^{\{p_1, \dots, p_t\}}$ is an access structure as defined in Definition 7. We will represent an access structure by a function $f : \{0, 1\}^* \rightarrow \{0, 1\}$, where an input (i.e., a string) $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n) \in \{0, 1\}^n$ represents the set $A_\sigma = \{p_i : i \in [n], \sigma_i = 1\}$,² and $f(\sigma) = 1$ if and only if $A_\sigma \in \Gamma$. We will also call f an evolving access structure.*

► **Definition 19** (Evolving Secret-Sharing Schemes). *Let S be a domain of secrets, where $|S| \geq 2$, and $\{R_t\}_{t \in \mathbb{N}}, \{S_t\}_{t \in \mathbb{N}}$ be two sequences of finite sets. An evolving secret-sharing scheme with domain of secrets S is a sequence of mappings $\Pi = \{\Pi^t\}_{t \in \mathbb{N}}$, where for every $t \in \mathbb{N}$, Π^t is a mapping $\Pi^t : S \times R_1 \times \dots \times R_t \rightarrow S_t$ (which returns the share sh_t of p_t).*

An evolving secret-sharing scheme $\Pi = \{\Pi^t\}_{t \in \mathbb{N}}$ realizes an evolving access structure Γ if for every $t \in \mathbb{N}$ the secret-sharing scheme $\Pi_t(s; r_1, \dots, r_t) \triangleq \langle \Pi^1(s; r_1), \dots, \Pi^t(s; r_1, \dots, r_t) \rangle$ (i.e., the shares of the first t parties) is a secret-sharing scheme realizing Γ^t according to Def. 9.

By default, the domain of secrets of an evolving scheme is $\{0, 1\}$. Known results show that every evolving access structure can be realized by an evolving secret-sharing scheme.

► **Definition 20** (Evolving Threshold Access Structures). *Let $k \in \mathbb{N}$. The evolving k -threshold access structure is the evolving access structure Γ , where Γ^t is the k -out-of- t threshold access structure.*

Komargodski et al. [17] showed that any evolving threshold access structure can be realized by an efficient evolving secret-sharing scheme.

¹ In [16] it is proven that one could define MSP's with any target vector $\epsilon \neq \mathbf{0}$, rather than $\mathbf{e}_1$, resulting in the same matrix size and labeling.

² In particular, the same set has infinitely many representations by inputs of various lengths, using sufficiently many trailing zeros.

► **Theorem 21** ([17]). *For every $k \in \mathbb{N}$, there is a secret-sharing scheme realizing the evolving k -threshold access structure such that the share size of party p_t is $(k-1) \cdot \log t + \text{poly}(k) \cdot o(\log t)$.*

More evolving access structures are defined by such structures as *undirected st-connectivity graphs* (*st-connectivity*), infinite decision trees (IDT), and *generalized infinite decision trees* (GIDT) [1]. Komargodski et al. [17] showed that every undirected st-connectivity access structure can be realized by an evolving secret-sharing scheme in which the share of each party is a bit. Infinite decision trees were used in [17, 18] to construct evolving secret-sharing schemes. Alon et al. [1] showed how to construct secret-sharing schemes for IDT and GIDT. For definitions and constructions we refer to the full version [11].

3.3 Share Conversion

Cramer et al. [9] defined the notion of a share conversion as a local mapping from the shares of a secret over one scheme into shares over another scheme, maintaining the secret value. We next include a formal definition of share conversion.³

► **Definition 22** (Share Conversion). *Let Π, Π' be two secret-sharing schemes over the same secret-domain S for n parties realizing the same access structure. We say that Π is locally convertible to Π' if there exist functions $g_1, \dots, g_n$ such that the following holds. If $(sh_1, \dots, sh_n)$ are valid shares of a secret s in Π (i.e., $\Pr[\Pi(s; \mathbf{r}) = (sh_1, \dots, sh_n)] > 0$), then $(g_1(sh_1), \dots, g_n(sh_n))$ are valid shares of the same secret s in Π' . We denote by g the concatenation of all g_i , namely $g(sh_1, \dots, sh_n) = (g_1(sh_1), \dots, g_n(sh_n))$, and refer to g as a conversion function.*

We next extend the definition of share conversion to the evolving setting.

► **Definition 23** (Evolving Share Conversion). *Let Π, Π' be two evolving secret-sharing schemes over the same secret-domain S realizing an access structure Γ . We say that Π is locally convertible to Π' if there exists a sequence of functions $g_1, g_2, g_3, \dots$ such that the following holds. For every $t \geq 1$, if $(sh_1, \dots, sh_t)$ are valid shares of a secret s in Π (i.e., $\exists \mathbf{r} \in R_1 \times \dots \times R_t$ such that $\Pi(s; \mathbf{r}) = (sh_1, \dots, sh_t)$), then $(g_1(sh_1), \dots, g_t(sh_t))$ are valid shares of the same secret s in Π' . We denote by g the concatenation of all g_i , namely $g(sh_1, sh_2, \dots) = (g_1(sh_1), g_2(sh_2), \dots)$, and refer to g as a conversion function.*

If the secret-sharing scheme Π is convertible to Π' , we say that $\Pi \geq \Pi'$. This defines a partial ordering over secret-sharing schemes. Next, we show that changing a target vector preserves much of the MSP structure, while being convertible to the original scheme.

▷ **Claim 24.** Let $\Pi = (\mathbb{F}, M \in \mathbb{F}^{m \times k}, \rho)$ is a linear scheme for an access structure Γ with the target vector $\epsilon \in \mathbb{F}^m \setminus \mathbf{0}$. Then for any target vector $\epsilon' \in \mathbb{F}^m \setminus \mathbf{0}$, there exists a linear scheme $\Pi' = (\mathbb{F}, M' \in \mathbb{F}^{m \times k}, \rho)$ for Γ , convertible to Π .

The proof is simply by observing the identity conversion works. See [11] for a full proof.

In linear (MSP-based) schemes, it is convenient to consider a secret s as part of the randomness vector $\mathbf{r}$, being its first coordinate. Sometimes, s is defined by $\mathbf{r}$ in a different manner, which results in a different than $\mathbf{e}_1$ target vector in MSP. For example, in CNF with $\mathbf{1}$ target, as used in [9], the secret is the sum of all elements in $\mathbf{r}$. Thus, we will sometimes consider conversions to a scheme Π' with a certain target vector, and implicitly rely on the implied conversion to Π' with a different target vector.

³ In [9], they in fact give a slightly more general definition.

4 Impossibility results for linear Share Conversion

Our impossibility results for linear schemes presented in this section follow from the lemma stated below.

► **Lemma 25.** *Let Π, Π' denote linear secret-sharing schemes realizing Γ and specified by MSPs $(\mathbb{F}, M \in \mathbb{F}^{m \times k}, \rho)$, $(\mathbb{F}, M' \in \mathbb{F}^{m' \times k'}, \rho')$. Let $T \cup \{h\}$ denote a minterm in Γ for $T \subseteq [n], h \in [n]$, and let $\alpha_{T \cup h}, \alpha'_{T \cup h}$ be its reconstruction functions for Π and Π' , respectively. Let $L = \text{Rowspan}(M_T) \cap \text{Rowspan}(M_h)$, and B denote a basis for L . Suppose g is some share conversion from Π to Π' . Then $(\alpha'_h)^\top g_h(M_h \mathbf{r}) = (\alpha_h)^\top M_h \mathbf{r} + c(B\mathbf{r})$ for some function c , where $\mathbf{r}$ is a randomness vector for Π .⁴*

► **Remark 26.** The lemma gives us a high level clue as to why CNF is high up in the convertibility (partial) order for linear schemes of a given access structure, while DNF is so low. In DNF, for every party h , and every minterm $T \cup \{h\}$, it is easy to see that the resulting $L = \{\mathbf{0}\}$. This way, there is “little freedom” in converting determining the converted (sub)share for h (this is formally proved in Theorem 31). On the other hand, in CNF, for a given h and minterm $T \cup \{h\}$, $\dim(L)$ is typically large – equal to the number of maxterms H where $H \cap (T \cup \{h\}) \subset T$. This allows for more freedom at choosing the conversion function.

Proof of Lemma 25. First observe that since $T \cup \{h\}$ is a minterm, $(\alpha_h)^\top M_h \notin \text{Rowspan}(M_T)$, and hence $(\alpha_h)^\top M_h \notin L$. Also, by definition, $B \subseteq \text{Rowspan}(M_h)$. Note that $\{(\alpha_h)^\top M_h\} \cup B$ may not constitute a basis of the rows of M_h , in which case, it is complemented by adding a set of appropriate linear combinations of an appropriate set X of rows in M_h . Let M_T^- denote a subset of M_T ’s rows constituting a basis of $\text{Rowspan}(M_T)$. By choice of B , for any scalar a and vector $\mathbf{v}$ (of the right dimension) there exists randomness $\mathbf{r}$ (one or more) such that $M_T^- \mathbf{r} = \mathbf{v}$, $(\alpha_h)^\top M_h \mathbf{r} = a$. Note that $B\mathbf{r}$ is determined by $\mathbf{v}$ (and is otherwise independent of $\mathbf{r}$). In the sequel, for (an implicit or explicit) randomness vector $\mathbf{r}$, we let $\mathbf{v}, a$ denote the share portions as above induced by it. As α' is a reconstruction function, and by locality of g and linearity of Π' , $\forall \mathbf{r}$ it holds that

$$(\alpha'_{T \cup h})^\top g_{T \cup h}(M_{T \cup h} \mathbf{r}) = (\alpha'_h)^\top g_h(M_h \mathbf{r}) + (\alpha'_T)^\top g_T(\mathbf{v}) = s \quad (1)$$

By Equation (1), and the definition of α we have that

$$(\alpha'_h)^\top g_h(M_h \mathbf{r}) = s - (\alpha'_T)^\top g_T(M_T \mathbf{r}) = (\alpha_h)^\top M_h \mathbf{r} + (\alpha_T)^\top M_T \mathbf{r} - (\alpha'_T)^\top g_T(M_T \mathbf{r}), \quad (2)$$

where the first term equals to a by definition, and the rest is some function of only $\mathbf{v}$. We denote $(\alpha_T)^\top M_T \mathbf{r} - (\alpha'_T)^\top g_T(M_T \mathbf{r})$ by $f(\mathbf{v})$. From locality of g , $f(\mathbf{v})$ in fact depends only on $B\mathbf{r}$. To see this, denote $\text{span}(M_T^-) = \text{span}(B) \oplus \text{span}(B_T^-)$ (direct sum of linear subspaces) for an independent set of vectors $B_T^- \subseteq \text{Rowspan}(M_T)$ complementing B into a basis. Using the fact that B is a basis of $\text{Rowspan}(M_h) \cap \text{Rowspan}(M_T)$, it is easy to show that B_T^- with $\text{span}(B_T^-) \cap \text{Rowspan}(M_h) = \{\mathbf{0}\}$ indeed exists.

The observation follows by locality, since g_h has no information on $\text{span}(B_T^-)\mathbf{r}$ (given $B\mathbf{r}$ that it knows). Also, $(\alpha'_h)^\top g_h(M_h \mathbf{r})$ may not depend on $X\mathbf{r}$, as $\text{span}(X) \cap \text{Rowspan}(M_T) = \{\mathbf{0}\}$, as then it would not be a (deterministic) function of $\mathbf{v}$.⁵ Thus, we have

$$(\alpha'_h)^\top g_h(M_h \mathbf{r}) = a + c(B\mathbf{r}) \quad (3)$$

for some function c , as required. ◀

⁴ Note that even if $L = \{\mathbf{0}\}$, we are free to pick the constant $c(\mathbf{0})$, which depends only on α in this case.

⁵ Note that the above does not imply that $g_h(M_h \mathbf{r})$ does not depend on $X\mathbf{r}$, but rather that it is of the form $g_h(B\mathbf{r}, a) + g_h''(X\mathbf{r})$, where $g_h''(X\mathbf{r}) = (g_h''_{h,1}(X\mathbf{r}), \dots, g_h''_{h,\ell}(X\mathbf{r}))$ is a vector of functions in the formal variables $X[1]\mathbf{r}, \dots, X[\ell]\mathbf{r}$, such that $(\alpha'_h)^\top g_h''(X\mathbf{r}) = 0$. Similarly, it implicitly follows that the vector

Next we state two corollaries defining necessary conditions for convertability between linear schemes Π, Π' (as sufficient conditions for non-convertability).

The following theorem is in a sense a generalization of the Claim in Section 3.3 in [9] of non-convertability from Shamir to CNF for $(2, 3)$ -threshold access structures.

► **Theorem 27.** *Let Γ be an access structure on n parties. Let Π and Π' be linear secret-sharing schemes realizing Γ and specified by MSP $(\mathbb{F}, M \in \mathbb{F}^{m \times k}, \rho)$ and $(\mathbb{F}, M' \in \mathbb{F}^{m' \times k'}, \rho')$, respectively. Then, Π has no share conversion to Π' if there exist $h \in [n]$, $\emptyset \neq T \subseteq [n] \setminus \{h\}$ such that $T \cup \{h\}$ is a minterm of Γ with the reconstruction functions α and α' in Π and Π' resp., and $\emptyset \neq T^* \subseteq [n] \setminus \{h\}$ that satisfy the following conditions:*

1. $(\alpha'_h)^\top \cdot M'_h \in \text{Rowspan}(M'_{T^*})$.
2. $(\alpha_h)^\top \cdot M_h \notin (\text{Rowspan}(M_T) \cap \text{Rowspan}(M_h)) + (\text{Rowspan}(M_{T^*}) \cap \text{Rowspan}(M_h))$.⁶

Proof. Let us assume for contradiction a conversion g exists, and denote $L = \text{Rowspan}(M_T) \cap \text{Rowspan}(M_h)$ and $L^* = \text{Rowspan}(M_{T^*}) \cap \text{Rowspan}(M_h)$. We first show that there exist randomness vectors $\mathbf{r}_1, \mathbf{r}_2$ such that:

1. $(L + \text{Rowspan}(M_{T^*}))\mathbf{r}_1 = (L + \text{Rowspan}(M_{T^*}))\mathbf{r}_2 = \mathbf{0}$.
2. $(\alpha_h)^\top M_h \mathbf{r}_1 \neq (\alpha_h)^\top M_h \mathbf{r}_2$.

Fix $\mathbf{r}_1 = \mathbf{0}$. Then, $(L + \text{Rowspan}(M_{T^*}))\mathbf{r}_1 = \mathbf{0}$ and $(\alpha_h)^\top M_h \mathbf{r}_1 = 0$. Next, let us show that $\mathbf{r}_2$ as required exists. We claim $\text{Rowspan}(M_h) \cap (L + L^*) = \text{Rowspan}(M_h) \cap (L + \text{Rowspan}(M_{T^*}))$.

Clearly, former is contained in the latter. For the other direction, consider $x \in \text{Rowspan}(M_h) \cap (L + \text{Rowspan}(M_{T^*}))$. Since L is a subspace of M_h , it suffices to show that if $x \in L + y$ such that $y \in \text{Rowspan}(M_{T^*})$, then $y \in \text{Rowspan}(M_{T^*}) \cap \text{Rowspan}(M_h) = L^*$. Indeed, if $y \in \text{Rowspan}(M_{T^*}) \setminus \text{Rowspan}(M_h)$, then $y + z \notin \text{Rowspan}(M_h)$ for all $z \in L$, which contradicts the fact $x \in \text{Rowspan}(M_h)$; the claim follows.

By this claim and condition 2 of the Theorem, $(\alpha_h)^\top M_h \notin L + \text{Rowspan}(M_{T^*})$ since $(\alpha_h)^\top M_h \in \text{Rowspan}(M_h)$. Therefore, there exists $\mathbf{r}_2$ such that $(L + \text{Rowspan}(M_{T^*}))\mathbf{r}_2 = \mathbf{0}$ and $(\alpha_h)^\top M_h \mathbf{r}_2 = 1 (\neq 0)$, by Fact 6.

By Lemma 25 applied to h, T , when B is the basis for L , we have

$$(\alpha'_h)^\top g_h(M_h \mathbf{r}_1) \neq (\alpha'_h)^\top g_h(M_h \mathbf{r}_2), \quad (4)$$

since $B\mathbf{r}_1 = B\mathbf{r}_2 (= \mathbf{0})$ and $(\alpha_h)^\top M_h \mathbf{r}_1 \neq (\alpha_h)^\top M_h \mathbf{r}_2$. On the other hand, by locality, $g_{T^*}(M_{T^*} \mathbf{r}_1) = g_{T^*}(M_{T^*} \mathbf{r}_2) = g_{T^*}(\mathbf{0})$, and thus

$$g_{T^*}(M_{T^*} \mathbf{r}_1) = g_{T^*}(M_{T^*} \mathbf{r}_2). \quad (5)$$

By condition 1 of the theorem, there exists $\mathbf{u}$ such that $(\alpha'_h)^\top M'_h = (\mathbf{u})^\top M'_{T^*}$. Further, by definition of g , there exists randomness vectors $\mathbf{r}'_1, \mathbf{r}'_2$ for Π' such that $g(M\mathbf{r}_i) = M'\mathbf{r}'_i$ for $i = 1, 2$. Hence,

$$\begin{aligned} (\alpha'_h)^\top g_h(M_h \mathbf{r}_1) &= (\alpha'_h)^\top M'_h \mathbf{r}'_1 = (\mathbf{u})^\top M'_{T^*} \mathbf{r}'_1 = (\mathbf{u})^\top g_{T^*}(M_{T^*} \mathbf{r}_1) \\ &= (\mathbf{u})^\top g_{T^*}(M_{T^*} \mathbf{r}_2) = (\mathbf{u})^\top M'_{T^*} \mathbf{r}'_2 = (\alpha'_h)^\top M'_h \mathbf{r}'_2 = (\alpha'_h)^\top g_h(M_h \mathbf{r}_2). \end{aligned}$$

The first equality in the second line uses Equation (5). We conclude our proof since the above sequence of equations contradicts Equation (4). ◀

$g_T(M_T \mathbf{r}) = g_T(B\mathbf{r}, B_T^- \mathbf{r})$, of functions is such that $(\alpha'_T)^\top g''_T(B_T^- \mathbf{r})$ is a formal function of $B\mathbf{r}$ only. That is in Fourier basis in the variables $y_1 = B[1]\mathbf{r}, \dots, y_{|B|} = B[|B|]\mathbf{r}, \dots, y_i = B_T^-[i - |B|]\mathbf{r}, \dots, y_k$, it does not contain minterms of the form $\prod_{j \in A} y_j$ where $A \setminus [|B|]$ is not empty. This also poses a certain restriction of g_T which can possibly be exploited by future work.

⁶ Recall that $U + V = \{u + v | u \in U, v \in V\}$.

The following corollary from Theorem 27 provides necessary conditions for $\Pi_\Gamma \leq \text{CNF}_\Gamma$ that are easy to check. These conditions involve only Γ , and the share size of a single party.

► **Corollary 28.** *Let Γ denote an access structure on n parties, such that there exists a party h and two minterms T_1, T_2 of size ≥ 2 each, such that $h \in T_1, h \in T_2, (T_1 \cup T_2) \setminus \{h\}$ is qualified. Let $\Pi = (\mathbb{F}, M \in \mathbb{F}^{m \times k}, \rho)$ be a linear scheme for Γ , and M_h consists of a single row. Then Π is not convertible to $\Pi' = \text{CNF}_{\Gamma, \mathbb{F}}$.*

In the proof we apply Theorem 27 to Π, Π' , with $T = T_1 \setminus \{h\}, T^* = T_2 \setminus \{h\}$. See [11] for details.

► **Remark 29.** From Corollary 28 it follows directly that Shamir secret-sharing scheme is not convertible to CNF, and hence is not maximal. Indeed, any party can be viewed as h , as it obtains a single field element as its share. The condition on the access structure automatically holds for threshold structures.

The following theorem exploits a different property implied by convertability than the previous theorem.

► **Theorem 30.** *Let Γ be an access structure on n parties. Let Π, Π' be linear secret-sharing schemes specified for Γ by MSP $(\mathbb{F}, M \in \mathbb{F}^{m \times k}, \rho)$ and $(\mathbb{F}, M' \in \mathbb{F}^{m' \times k'}, \rho')$, respectively. Let $h \in [n]$ denote a party, and let $\mathcal{T}^h = \{T_1, \dots, T_{v_h}\}$ denote minterms each of which is of size ≥ 2 , and contains h . Let $\{\alpha^{T_i}\}_{i \in [v_h]}$ denote a set of reconstruction functions for sets $T_1, \dots, T_{v_h}$ in Π respectively. Similarly, $\{\alpha'^{T_i}\}_{i \in [v_h]}$ denotes a set of reconstruction functions for $T_1, \dots, T_{v_h}$ in Π' . Assume the following conditions hold:*

1. $A = \{\alpha_h^{T_i}\}_{i \in [v_h]}$ constitutes a set of v_h linearly independent vectors.
2. $A' = \{\alpha_h'^{T_i}\}_{i \in [v_h]}$ constitutes a (multi)set of vectors with $\text{rank}(\text{span}(A')) < v_h$.
3. $\text{span}(\{\alpha^\top M_h | \alpha \in A\}) \cap \sum_{i \in [v_h]} (\text{Rowspan}(M_{T_i \setminus h}) \cap \text{Rowspan}(M_h)) = \{\mathbf{0}\}$.

Then Π has no share conversion to Π' .

Proof. Assume for contradiction that a conversion g exists. Fix some party $h \in [n]$ as guaranteed to exist by the theorem. Consider the matrix $\text{Rec}^h = [(\alpha_h^{T_1})^\top M_h; \dots; (\alpha_h^{T_{v_h}})^\top M_h]$. For each $i \in [v_h]$, let $L_i = \text{Rowspan}(M_{T_i \setminus h}) \cap \text{Rowspan}(M_h)$. By condition 1, it is of rank v_h . Therefore, there exists a set $R^h = \{\mathbf{r}^1, \dots, \mathbf{r}^{|\mathbb{F}|^{v_h}}\}$ of randomness values such that

1. $\{\text{Rec}^h \mathbf{r}^j\}_{j \in [|\mathbb{F}|^{v_h}]}$ goes over all distinct vectors in $\mathbb{F}^{v_h}$.
2. For every $j \in [|\mathbb{F}|^{v_h}], i \in [v_h], L_i \mathbf{r}^j = \mathbf{0}$.

Such a set R^h exists, by combining conditions 1 and 3 in the Theorem. Now, let us consider the matrix $\text{Rec}'^h = [(\alpha_h'^{T_1})^\top M_h; \dots; (\alpha_h'^{T_{v_h}})^\top M_h]$, and let $R'^h = \{\mathbf{r}'^1, \dots, \mathbf{r}'^{|\mathbb{F}|^{v_h}}\}$ denote the set of effective randomness vectors induced by the conversion g (that is $M' \mathbf{r}'^j = g(M \mathbf{r}^j)$). By condition 2 in the Theorem, $\text{rank}(\text{Rec}'^h) < v_h$. Therefore, $\text{span}(\{\text{Rec}'^h \mathbf{r}'^j\}_{j \in [|\mathbb{F}|^{v_h}]})$ is of dimension at most $v_h - 1$, and thus contains at most $|\mathbb{F}|^{v_h-1}$ distinct values. However, by Lemma 25, for every $i \in [v_h], j \in [|\mathbb{F}|^{v_h}]$ we have

$$\text{Rec}'^h[i] \mathbf{r}'^j = \text{Rec}^h[i] \mathbf{r}^j + c^i(\mathbf{0})$$

That is, $c^j(\mathbf{0})$ is some constant, independent of $\mathbf{r}^j$. Therefore,

$$R'^h \mathbf{r}'^j = R^h[j] \mathbf{r}^j + \mathbf{c}$$

where $\mathbf{c}$ is a constant vector. Going over all values of j , on the RHS we obtain a permutation of the set $\mathbb{F}^{v_h}$, but on the LHS, we only get a subset of at most $|\mathbb{F}|^{v_h-1}$ (constituting a linear subspace) – a contradiction. ◀

The above theorem implies that DNF can only be converted to linear schemes of size at least that of the DNF. More precisely:

► **Theorem 31.** *Let Γ be an access structure over $n > 1$ parties, and $\mathbb{F}$ be a finite field. For simplicity, assume Γ has no redundant parties. Let Π' be a linear secret-sharing scheme specified by MSP $\mathcal{M}' = (\mathbb{F}, M' \in \mathbb{F}^{m' \times k'}, \rho')$ for Γ . If $\mathcal{M} = \text{DNF}_{\Gamma, \mathbb{F}} = (\mathbb{F}, M \in \mathbb{F}^{m \times k}, \rho)$ is convertible to Π' , then ⁷*

$$m' \geq \sum_{T: T \text{ is a minterm of } \Gamma} |T|.$$

Proof. Consider some $h \in [n]$ that belongs to some minterm T of size at least 2. Let $\mathcal{T}^h = \{T_i \text{ is a minterm} \mid |T_i| \geq 2, h \in T_i\}$, and denote $v_h = |\mathcal{T}^h|$. The (unique, in case of DNF) reconstruction function projection set $A = \{\alpha_h^{T_i}\}$ are all linearly independent by construction of DNF. Namely, each selects a single row in M_h , and $(\alpha_h^{T_i})^\top M_h$ is either of the form $r_{i,h}$ or of the form $s - \sum_{k \in T_i \setminus h} r_{i,k}$, where $r_{i,j}$ is a random element used by the DNF in the additive sub-scheme for minterm T_i and party j .

For a fixed $k \in [v_h]$, by structure of DNF, we have $\text{Rowspan}(M_{T_k \setminus h}) \cap \text{Rowspan}(M_h) = \mathbf{0}$. Therefore, $\sum_{k \in [v_h]} (\text{Rowspan}(M_{T_k \setminus h}) \cap \text{Rowspan}(M_h)) = \{\mathbf{0}\}$. Thus, A satisfies conditions 1 and 3 of Theorem 30 relative to $\mathcal{T}^h$.

It follows from Theorem 30 that A' is of dimension $\geq v_h$. Thus, the matrix $A'M'_h$ (where A' is treated as a matrix constructed from the vectors of A' as rows) also has dimension at least v_h . This implies M'_h itself is of rank at least v_h , which in turn lower bounds its number of rows.

Parties that constitute a singleton minterm, contribute at least 1 to $\text{rank}(M'_h)$, as the party contained in the minterm is not redundant. Going over all parties h , the result follows by changing the order of summation, obtaining.

$$m' \geq \sum_h \sum_{T: T \text{ is a minterm}, h \in T} 1 = \sum_{T: T \text{ is a minterm}} |T| \quad (6)$$

◀

► **Remark 32.** Theorem 31, in fact, states that DNF is not convertible to any linear scheme with lower share complexity, in particular, in the Shamir secret sharing scheme. The non-minimality of Shamir follows.

► **Remark 33.** In fact, the above theorem can be fairly easily generalized to Π where $\text{span}(\{\alpha^\top M_h \mid \alpha \in A\}) \cap \sum_{i \in [v_h]} (\text{Rowspan}(M_{T_i \setminus h}) \cap \text{Rowspan}(M_h)) = U$ for some h has a “relatively small” dimension u . Then, we can prove $\dim(\text{span}(A)) - u$ lower bounds $\text{rank}(M'_h)$ if $\Pi \leq \Pi'$. That is, the smaller u relatively to $\dim(\text{span}(A))$ is, the higher the share complexity of party h in Π' must be.

5 A characterization of convertability between linear schemes

Let Γ be an access structure on n parties. Let Π, Π' be linear secret-sharing schemes specified by MSPs $(\mathbb{F}, M \in \mathbb{F}^{m \times k}, \rho)$ and $(\mathbb{F}, M' \in \mathbb{F}^{m' \times k'}, \rho')$, respectively, realizing Γ . We devise a characterization of convertibility from Π to Π' by solvability of a certain system of linear equations. Essentially, every solution of the system represents a conversion function g . Namely, for every randomness vector from Π , it defines converted shares for Π' .

⁷ In fact, the number of rows assigned to each party is at least as large as in DNF.

The linear system $\mathcal{L}_{\Pi, \Pi'}$. For each randomness vector $\mathbf{r} \in \mathbb{F}^m$ we define a variable $X^{(\mathbf{r})} \in \mathbb{F}^{m'}$, that assumes a value for the purported sharing under M' induced by the share conversion of $M \cdot \mathbf{r}$. The constraints we define are as follows.

- **Locality.** For every $i \in [n]$ and $\mathbf{r}, \mathbf{r}' \in \mathbb{F}^m$ such that $M_i \cdot \mathbf{r} = M_i \cdot \mathbf{r}'$, add the constraint:

$$X_i^{(\mathbf{r})} = X_i^{(\mathbf{r}')}.$$

- **Consistency.** Let $A \subseteq [m']$ be a subset of rows of M' which form a basis of $\text{Rowspan}(M')$. Since $M'[A, \cdot]$ is a basis of $\text{Rowspan}(M')$, there exists a (unique) matrix $H \in \mathbb{F}^{m', |A|}$ such that $H \cdot M'[A, \cdot] = M'$. For every $\mathbf{r} \in \mathbb{F}^m$, we add the constraint

$$H \cdot X^{(\mathbf{r})}[A] = X^{(\mathbf{r})}.$$

- **Correctness.** For each minterm $T \subset [n]$ of Γ do the following: let $\alpha'^T \in \mathbb{F}^{|\rho'^{-1}(T)|}$ be a smallest reconstruction vector for T under some arbitrary ordering of $\mathbb{F}^{|\rho'^{-1}(T)|}$. For every $s \in \mathbb{F}$, and every $\mathbf{r}$ such that $\mathbf{r}[1] = s$, add the constraint

$$\left(\alpha'^T\right)^\top \cdot X_T^{(\mathbf{r})} = s.$$

► **Remark 34.** The characterization may be easily extended to non-linear Π with $S = \mathbb{F}$, keeping the system linear.

► **Theorem 35.** Let Γ be an access structure on n parties. Let Π, Π' be linear secret-sharing schemes specified by MSPs $(\mathbb{F}, M \in \mathbb{F}^{m \times k}, \rho)$ and $(\mathbb{F}, M' \in \mathbb{F}^{m' \times k'}, \rho')$, respectively, realizing Γ . Then, Π is convertible to Π' if and only if the linear system $\mathcal{L}_{\Pi, \Pi'}$ is solvable.

Proof. Both directions essentially follow from the fact that the solution set to $\mathcal{L}_{\Pi, \Pi'}$ exactly corresponds to the set of conversions from Π to Π' . See [11] for a formal proof. ◀

Theorems 27 and 30 may be viewed as a result of an inconsistency in the characterization equations. In Theorem 27, the inconsistent subset of equations supported only on $X^{(\mathbf{r}^1)}, X^{(\mathbf{r}^2)}$ for a pair $\mathbf{r}_1, \mathbf{r}_2$ as chosen in the proof, and the inconsistency involves a potentially large set of $\mathbf{r}^j$'s, already. Lemma 25 relies on equations of all three types to derive its conclusion.

Future work. We believe that there exist additional types of “inconsistencies” in the linear equations in the characterization that may result in proving non-existence of a conversion from between pairs of linear schemes Π to Π' , and it is an interesting open question to list all possible types of inconsistencies, and thereby make the above characterization easier to apply. Most importantly, we wish to understand when conversions between linear schemes *do* exist. Another interesting open question on convertability between pairs of linear schemes is understanding whether it is helpful to have non-linear conversion functions g .

6 Impossibility of conversion to CNF for general schemes

In this section, we introduce the class of non-degenerate secret-sharing schemes, which includes CNF and Shamir schemes, and, using properties of non-degenerate schemes, we prove a necessary condition of convertability to CNF from any (not necessarily linear) scheme.

► **Definition 36.** Let Π be a secret-sharing scheme with secret domain S and randomness domain R realizing an access structure Γ . Π is non-degenerate if the following holds: If Π' is a secret-sharing scheme with secret domain S and randomness domain R' realizing Γ such that, for all $s \in S, r' \in R'$, there exists $r \in R$ such that $\Pi'(s; r') = \Pi(s; r)$, then

$$(\Pi'(s; r') | r' \leftarrow R') \equiv (\Pi(s; r) | r \leftarrow R), \forall s \in S. \quad (7)$$

Suppose Π is a non-degenerate secret-sharing scheme. If a secret-sharing scheme Π' is locally convertible to Π , then the secret-sharing scheme induced by the applying the share conversion function to Π' coincides with Π .

► **Proposition 37.** *Let Π be a non-degenerate secret-sharing scheme with secret domain S and randomness domain R realizing access structure Γ over n parties. Suppose Π' be a secret-sharing scheme with the same secret domain and access structure and randomness domain R' that admits share conversion to Π using a share conversion function $g = (g_1, \dots, g_n)$. Then, for all $s \in S$,*

$$(g(\Pi'(s; r')) | r' \leftarrow R') \equiv (\Pi(s; r) | r \leftarrow R). \quad (8)$$

Proof. Consider the secret-sharing scheme in which $s \in S$ is secret shared as $(sh_1, \dots, sh_n) = g(\Pi'(s; r'))$ where $r' \leftarrow R'$. Since g is a share conversion function that converts Π' to Π , this induces secret-sharing scheme with secret domain S realizing the access structure Γ . Further, for each $s \in S$ and $r' \in R'$, $g(\Pi'(s; r')) = \Pi(s; r)$ for some $r \in R$. The proposition now follows from the fact that Π is a non-degenerate secret-sharing scheme (See Definition 36). ◀

We establish that CNF and Shamir secret sharing schemes are non-degenerate. The proofs of these claims follow the outline sketched in Section 1.1; their formal proofs are deferred to the full version.

► **Lemma 38.** *For any finite group $\mathbb{G}$, and access structure Γ over n parties, the CNF secret-sharing scheme for secrets in $\mathbb{G}$ realizing Γ is non-degenerate.*

► **Lemma 39.** *For any finite field $\mathbb{F}$ such that $|\mathbb{F}| > n$, and $1 \leq t \leq n$, a t -private n -party Shamir secret-sharing scheme over $\mathbb{F}$ is non-degenerate.*

We exploit the non-degeneracy of CNF secret sharing to establish lower bound on share size in any (potentially non-linear) secret sharing schemes that admit share conversion to CNF secret sharing.

► **Theorem 40.** *Let Π be a secret-sharing scheme with secret domain $\mathbb{G}$ and randomness domain R realizing an n -party access structure Γ . There is a share conversion from Π to CNF secret-sharing over $\mathbb{G}$ realizing Γ only if, for each $i \in [n]$, size of the share i in Π is at least $\log |\mathbb{G}| \cdot |\{F \in \mathcal{F} \text{ s.t. } i \notin F\}|$, where $\mathcal{F}$ is the set of all maximal forbidden sets associated with Γ .*

Proof. By Lemma 38, the CNF secret-sharing scheme is non-degenerate. Let $g = (g_1, \dots, g_n)$ be the share conversion function that induces the share conversion from Π to the CNF secret-sharing scheme. By Proposition 37, for any $s \in \mathbb{G}$, when $r \leftarrow R$, $g(\Pi(s; r))$ is identically distributed as CNF secret-sharing of s . Hence, $g_i(\Pi(s; r))$ corresponds to the share of party i in CNF secret-sharing: $\{\gamma_F : F \in \mathcal{F}, i \notin F\}$ where γ_F is uniformly chosen from $\mathbb{G}$ for each $F \in \mathcal{F}$ subject to $\sum_F \gamma_F = s$. Theorem follows immediately from this observation. ◀

7 Results for Evolving Linear Secret-Sharing Schemes

In this section, we extend the notion of Monotone Span Programs and the induced notion of a linear secret-sharing scheme to the evolving setting. We then apply our impossibility results obtained in Sections 4 and 6 for the finite case to study the convertibility hierarchy in this setting.

Monotone span programs [16] were used to construct linear secret-sharing schemes in [4]. In this section, we extend Definition 15 to define infinite monotone span programs and cast a few constructions from the literature as instances of this notion. We define the product of an infinite matrix $K \in \mathbb{F}^{[n] \times \mathbb{N}^+}$ by a finite vector $\mathbf{r} \in \mathbb{F}^{[m]}$ as $K'\mathbf{r}$, where K' is obtained by keeping the first m columns of K . We will typically use such products for matrices where all but the first m columns are 0. Generalizing this notion to the evolving setting, we dub IMSP, requires some care. Roughly, in an IMSP $M(\mathbb{F}, M, \rho)$ $|\rho^{-1}(i)|$ is finite for all i , every row, as well as the target vector (typically $\mathbf{e}_1 = (1, 0, \dots, 0)$) have finitely many non-0 elements. See [11] for a formal definition.⁸

In the following theorem, we generalize the MSP-based linear secret-sharing schemes to the evolving setting, essentially giving each party the linear combinations of a randomness vector (that also defines the secret s), as specified by the IMSP. As in the finite case, every finite subset $A \subseteq \mathbb{N}^+$ either reconstructs the secret, or learns nothing about it.

► **Theorem 41.** *Let $\mathcal{M} = (\mathbb{F}, M, \rho)$ be an IMSP accepting an access structure Γ . Then, Construction 42 instantiated with $\mathcal{M}$ implements Γ .*

► **Construction 42.** *Consider an IMSP $\mathcal{M}(\mathbb{F}, M, \rho)$.*

■ **INPUT:** *a secret $s \in \mathbb{F}$.*

We determine $r_0 = s$ and define $\mathbf{r} = (s, r_1, r_2 \dots)$.

■ **SHARE:** *To generate sh_i the dealer does the following:*

1. *Gets as input $(s, \text{sh}_1, \dots, \text{sh}_{i-1})$, that is, a secret s and the shares of parties $p_1, \dots, p_{i-1}$. For convenience, we assume it in fact receives the set of $r_1, \dots, r_j$'s sampled so far, for $j = \max(C_{[i-1]})$. It samples random independent elements $r_{j+1}, \dots, r_{j+d} \in \mathbb{F}$ for $j + d = \max(C_{[i]})$.*
2. *Set $\text{sh}_i = M_i \mathbf{r}$, where $\mathbf{r}$ is the prefix of $\mathbf{r}$ sampled so far.*

■ **RECON:** *Let α denote the (finite) reconstruction vector such that $\alpha^T M_B = \mathbf{e}_1$. Return $\langle \alpha, \Pi(B) \rangle$.*

The proof of this theorem is deferred to the main version [11] and re-interprets the construction of [1] as an IMSP based scheme. We define *evolving linear secret-sharing schemes* as the set of schemes so specified by an IMSP.

For constructions for the evolving threshold, and for the evolving undirected *st*-connectivity access structures families as instances of IMSP, we refer the reader to the full version [11].

► **Theorem 43.** *Let Γ denote an evolving access structure. Then GIDT [1, Construction 3.9] for $S = \mathbb{F}_2$ and Γ instantiated so that edge predicates are implemented by linear schemes over $\mathbb{F}_2$ is (evolving) linear.*

The proof immediately follows from observing the GIDT [1] given as Construction 3.9.

The following theorem states that for a large class of evolving access structures there is no minimal linear evolving secret-sharing scheme. This proof follows from Theorem 27 applied to any specific evolving scheme and a tailor crafted GIDT scheme [1, Construction 3.9], and is deferred to the full version [11].

⁸ We use a “working definition” of linear evolving secret-sharing schemes specified by an IMSP, which is a natural extension of the finite case. An arguably more intuitive definition requires that all shares are linear combinations of the r_i 's and s (over a field $\mathbb{F}$) without the restriction on reconstruction. Beimel has demonstrated in [3] that linear schemes imply MSPs of similar share complexity, so the definitions are equivalent. We do not demonstrate such a result in this paper, but it would be useful to demonstrate in future work on the theory of linear evolving schemes.

► **Theorem 44.** *Consider an evolving access structure Γ such that there exists $\tilde{h} \in [n]$, and an infinite collection of minterms $A = \{T_i\}_{i \in \mathbb{N}}$ where $\tilde{h} \in T_i$ for all $i \in \mathbb{N}$. Then, for any linear scheme $\tilde{\Pi}$ specified by M over $\mathbb{F}_2$, there exists an evolving linear scheme $\tilde{\Pi}'$ specified by M' over $\mathbb{F}_2$ for Γ , such that $\tilde{\Pi}'$ is not convertible to $\tilde{\Pi}$.*

Next, using results obtained in Section 6, we prove the absence of a maximal evolving scheme in a wide class of evolving secret-sharing schemes, even not necessarily linear.

► **Definition 45** (Trivial Evolving Access Structures). *An evolving access structure Γ is said to be trivial if there exists $N \in \mathbb{N}$ such that, for all $n > N$, $\{n\} \in \Gamma$ or for all finite set A , $A \in \Gamma$ only if $A \setminus \{n\} \in \Gamma$.*

► **Theorem 46.** *Any non-trivial evolving access structure Γ and finite field $\mathbb{F}$ has no maximal evolving secret-sharing scheme over $\mathbb{F}$.*

Proof. We will need the following technical claim on evolving access structures (see [11] for a proof).

▷ **Claim 47.** If Γ is nontrivial, then for any $k \in \mathbb{N}$, there exists $n \in \mathbb{N}$ such that $|\{F \in \mathcal{F}_n : 1 \notin F\}| \geq k$, where $\mathcal{F}_n$ is the set of max-terms of Γ_n .

Now, let Π be a purported maximal secret-sharing scheme for one bit secrets realizing the access structure Γ . Let $|\text{sh}_1|$ be the share size of the share assigned to party 1 by Π . By the above claim, there exists n such that, when $\mathcal{F}_n$ is the set of max-terms of Γ_n , $|\{F \in \mathcal{F}_n : 1 \notin F\}| > |\text{sh}_1|$. Consider the GIDT-based construction Π' for Γ of [1] with the first generation consisting of n parties, and a CNF implementation of Γ_n , in the edge going from the root to a leaf. By Theorem 40, Π does not have a share conversion to Π' since the size of share i is less than $|\{F \in \mathcal{F}_n : 1 \notin F\}| > |\text{sh}_1|$. ◀

8 Extensions and applications

Above, we considered secret-share conversion for schemes defined over the same field. In this section, we discover the possibility to perform share conversion between schemes over different fields. We show that the most of our impossibility results is applicable to the case when the source and target schemes are defined over the fields of the same characteristic, and when characteristics are different, the conversion is not possible for many access structures.

8.1 Extending impossibility results to schemes over different fields of the same characteristics

In this section, we extend our impossibility results following from Theorem 27 to secret sharing schemes defined over distinct fields of the same characteristic. For this, we slightly restate Lemma 25 and Theorem 27 such that Π and Π' are two secret sharing schemes defined over $\mathbb{F}$ and $\mathbb{F}'$, respectively, which are the extension fields of $\mathbb{F}_p$.

► **Observation 48.** *Lemma 25 holds for secret-sharing schemes Π defined by MSP $(\mathbb{F}, M \in \mathbb{F}^{m \times k}, \rho)$ and Π' defined by MSP $(\mathbb{F}', M' \in \mathbb{F}'^{m' \times k'}, \rho')$ with the secret domain $\mathbb{F}_p$, where $\mathbb{F}$ and $\mathbb{F}'$ are extension fields of $\mathbb{F}_p$ for prime p .*

Proof. The proof is the same as the proof of Lemma 25, with the difference that (1) holds over $\mathbb{F}'$, Equation (2) is over $\mathbb{F}^*$ which is the extension field for both $\mathbb{F}$ and $\mathbb{F}'$, and Equation (3) is over $\mathbb{F}'$ (where function c is over $\mathbb{F}^*$ with the output in $\mathbb{F}'$). ◀

► **Observation 49.** *Theorem 27 holds for secret-sharing schemes Π defined by MSP $(\mathbb{F}, M \in \mathbb{F}^{m \times k}, \rho)$ and Π' defined by MSP $(\mathbb{F}', M' \in \mathbb{F}'^{m' \times k'}, \rho')$ with the secret domain $\mathbb{F}_p$, where $\mathbb{F}$ and $\mathbb{F}'$ are extension fields of $\mathbb{F}_p$ for prime p .*

Proof. The proof of this observation is the same as the proof of Theorem 27, where all the equations before (4) are over $\mathbb{F}$. To obtain Equation (4), we apply Theorem 27 under Observation 48 which facilitates the transition to the field $\mathbb{F}'$. Equations (4), (5), and all the following proof are constructed over $\mathbb{F}'$. ◀

We rely on the above observation to extend our impossibility result in Corollary 28 for conversion from certain linear schemes to CNF. In the observation below, both source and target schemes work for sharing secrets in the larger domains $\mathbb{F}, \mathbb{F}'$, but are used only to share secrets in the base field $\mathbb{F}_p$. Any such scheme may be viewed as a scheme over the smaller field S , where each party, including h receives several field elements, which correspond to operations over $\mathbb{F}_p$. Shamir over a large extension field of $\mathbb{F}_p$ is one example of such a scheme.

► **Observation 50.** *Corollary 28 holds for secret-sharing schemes Π defined by MSP $(\mathbb{F}, M \in \mathbb{F}^{m \times k}, \rho)$ and $\Pi' = \text{CNF}_{\Gamma, \mathbb{F}'}$ with the secret domain $\mathbb{F}_p$, where $\mathbb{F}$ and $\mathbb{F}'$ are extension fields of $\mathbb{F}_p$ for prime p .*

Proof. The proof of this observation is the same as the proof of Corollary 28, where Theorem 27 is applied under Observation 48. ◀

The above result is not subsumed by Theorem 40, as it allows $\mathbb{F}$ to be much larger than $\mathbb{F}'$.

8.2 Negative result for the inter-field conversion

It is often useful in applications to perform share conversion between schemes over different fields of different characteristic. A natural choice of a secret domain for such a conversion is $\{0, 1\}$, as these values belong to all finite fields. Furthermore, these values can be viewed as bits, which is the most useful setting for most MPC protocols. In this section, we show that a local conversion, in general, is not possible (even from CNF) for many access structures. However, below we show a specially tailored leaky secret-sharing scheme over field $\mathbb{Z}_p$ which allows local conversion to a different field $\mathbb{Z}_q$ for $q < n/2$ for (n, n) -threshold.

Next, we observe that for all pairs $p \neq q$, and many access structures Γ , one can not convert from $\text{CNF}_{\Gamma, \mathbb{F}_p}$ to $\Pi_{\Gamma, \mathbb{F}_q}$, where $\Pi_{\Gamma, \mathbb{F}_q}$ is any linear scheme over q for that share. More precisely, we have

► **Theorem 51.** *Let Γ denote an access structure for $n > 1$ parties, such that for all maxterms B_1, B_2 , $B_1 \cup B_2 = [n]$.⁹ Let $p \neq q$ be primes, and $\text{CNF}_{\Gamma, \mathbb{F}_p}$ and $\Pi_{\Gamma, \mathbb{F}_q}$ linear schemes for Γ over $\mathbb{F}_p, \mathbb{F}_q$ respectively. Then $\text{CNF}_{\Gamma, \mathbb{F}_p}$ is not convertible to $\Pi_{\Gamma, \mathbb{F}_q}$ for secret domain $S = \{0, 1\}$ (that is, we do not care how other secrets are converted).*

The theorem follows almost immediately from a variant of Lemma 25 for different fields p, q which we provide below (see full version [11] for a proof).

▷ **Claim 52.** Let $\Pi = (\mathbb{F}_p, M \in \mathbb{F}_p^{m \times \ell}, \rho)$, $\Pi' = (\mathbb{F}_q, M' \in \mathbb{F}_q^{m' \times \ell'}, \rho')$ for a pair of primes $p \neq q$, and let $T \cup \{h\}$ denote a minterm of Γ . Let $\alpha_{T \cup h}, \alpha'_{T \cup h}$ be reconstruction functions for $T \cup \{h\}$ in Π and Π' respectively. Assume $L = \text{Rowspan}(M_T) \cap \text{Rowspan}(M_h) = \{0\}$. Then for every

⁹ For instance, the $(\lceil n/2 \rceil + 1, n)$ -threshold access structure.

conversion scheme g from Π to Π' there exists a sequence $\mathbf{r}_1, \dots, \mathbf{r}_i, \dots \in \mathbb{Z}^\ell$ and constant $c \in \mathbb{Z}$ such that (1) $(\alpha'_h)^\top g_h(M_h \mathbf{r}_i \bmod p) \equiv i + c \pmod{q}$; (2) $(\alpha_h)^\top M_h \mathbf{r}_i \equiv i \pmod{p}$ for all $i \in \mathbb{N}^+$, and (3) $\langle \mathbf{r}_i, \mathbf{e}_1 \rangle \bmod p \in \{0, 1\}$. We conclude that such a g does not exist if $p \neq q$.

Proof of Theorem 51. Finally, the theorem follows from Claim 52 by observing that for $n > 2$, in $CNF_{\Gamma, \mathbb{F}_p}$ each party p_i gets a subset of independent random vectors over $\mathbb{F}_p$, namely $\mathbf{r}_T$ of each maxterm T such that $i \notin T$. The sets of $\mathbf{r}_H$'s that h holds, vs those T holds. Assume the contrary – that $h \notin H$ and $T \cap H = \emptyset$. In that case $i \notin T \cup H$, contradicting the assumptions that $T \cup H = [n]$ (as T, H are maxterms). This implies that for $CNF_{\Gamma, \mathbb{F}_p}$, $L = \{0\}$, so the conditions of Claim 52 are indeed satisfied by $\Pi = CNF_{\Gamma, \mathbb{F}_p}$ and $\Pi' = \Pi_{\Gamma, \mathbb{F}_q}$. $\blacktriangleleft$

8.3 The specially tailored additive secret-sharing scheme allowing inter-field conversion

Next, we build the secret-sharing scheme over the field $\mathbb{Z}_p$, and define the conversion function to the field $\mathbb{Z}_q$. The scheme implies some restrictions on the randomness of the dealer, and also is not perfectly secure. However its existence raises a question if there are statistical secure secret-sharing schemes allowing an inter-field conversion, and how small the leakage could be. The other question is about the possibility to convert the original additive scheme over $\mathbb{F}_p$ to a perfectly private scheme over $\mathbb{F}_q$, by using the entire randomness domain of the former. This way, privacy of the converted scheme automatically holds by locality of the conversion scheme. Correctness is the only measure that suffers, with some small probability. This way, we are talking about a conversion scheme between the same pair of perfect schemes Π, Π' , while the conversion scheme itself is statistically correct.

The n -party additive convertible scheme $ADD_{p \rightarrow q}$.

Parameters: $p \neq q$ are primes such that $q < n/2$.

Sharing algorithm:

- (1) the dealer for each $i \in [n]$ samples $r_i \leftarrow \mathbb{Z}_p$.
- (2) If $\sum_{i=1}^n r_i = kpq + s$, where $s \in \{0, 1\}$ for some k then output $\text{sh}_i := r_i$ and terminate. Otherwise go to Step 1.

Conversion function: Each p_i computes $\text{sh}'_i = \text{sh}_i \bmod q$.

This scheme is **correct** by construction, with **polynomial computational complexity**, and statistical **leakage** less or equal to $p_{\text{leak}} = \frac{1}{p} + o\left(\frac{1}{p^{2n}}\right)$ (see the full version [11]).

The convertible additive scheme is the basic case for creating the convertible CNF and DNF schemes. However, even directly it could have several applications, similar to applications of *dBits*. For more details, we refer to the full version [11]. We leave the existence of practical inter-field convertible secret-sharing schemes with the statistical, or even computational security, as the open question for the future research.

References

- 1 Bar Alon, Amos Beimel, Tamar Ben David, Eran Omri, and Anat Paskin-Cherniavsky. New upper bounds for evolving secret sharing via infinite branching programs. Cryptology ePrint Archive, Paper 2024/419, 2024. URL: <https://eprint.iacr.org/2024/419>.
- 2 Abdelrahman Aly, Emmanuela Orsini, Dragos Rotaru, Nigel P Smart, and Tim Wood. Zaphod: Efficiently combining LSSS and garbled circuits in SCALE. In *Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 33–44, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3338469.3358943.

- 3 A. Beimel. Secure schemes for secret sharing and key distribution. *Phd thesis*, 1996.
- 4 Amos Beimel. Secret-sharing schemes: A survey. In Yeow Meng Chee, Zhenbo Guo, San Ling, Fengjing Shao, Yuansheng Tang, Huaxiong Wang, and Chaoping Xing, editors, *Coding and Cryptology*, pages 11–46, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. doi:10.1007/978-3-642-20901-7_2.
- 5 Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Ilan Orlov. Share conversion and private information retrieval. In *Proceedings - 2012 IEEE 27th Conference on Computational Complexity, CCC 2012*, Proceedings of the Annual IEEE Conference on Computational Complexity, pages 258–268, September 2012. IEEE Computer Society Technical Committee on Mathematical Foundations of Computing ; Conference date: 26-06-2012 Through 29-06-2012. doi:10.1109/CCC.2012.23.
- 6 Aner Ben-Efraim. On multiparty garbling of arithmetic circuits. In *Advances in Cryptology—ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part III 24*, pages 3–33, Cham, 2018. Springer, Springer International Publishing. doi:10.1007/978-3-030-03332-3_1.
- 7 Aner Ben-Efraim, Lior Breitman, Jonathan Bronshtein, Olga Nissenbaum, and Eran Omri. MYao: Multiparty “Yao” garbled circuits with row reduction, half gates, and efficient online computation. *Cryptology ePrint Archive*, 2024. URL: <https://eprint.iacr.org/2024/1430>.
- 8 Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J Wu. Exploring crypto dark matter: New simple PRF candidates and their applications. In *Theory of Cryptography Conference*, pages 699–729, Cham, 2018. Springer, Springer International Publishing. doi:10.1007/978-3-030-03810-6_25.
- 9 Ronald Cramer, Ivan Damgård, and Yuval Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In Joe Kilian, editor, *Theory of Cryptography*, pages 342–362, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. doi:10.1007/978-3-540-30576-7_19.
- 10 Ivan Damgård and Rune Thorbek. Efficient conversion of secret-shared values between different fields. *Cryptology ePrint Archive*, 2008. URL: <https://eprint.iacr.org/2008/221>.
- 11 Tamar Ben David, Varun Narayanan, Olga Nissenbaum, and Anat Paskin-Cherniavsky. New results in share conversion, with applications to evolving access structures. *Cryptology ePrint Archive*, 2024. URL: <https://eprint.iacr.org/2024/1781>.
- 12 Itai Dinur, Steven Goldfeder, Tzipora Halevi, Yuval Ishai, Mahimna Kelkar, Vivek Sharma, and Greg Zaverucha. MPC-friendly symmetric cryptography from alternating moduli: candidates, protocols, and applications. In *Advances in Cryptology—CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part IV 41*, pages 517–547, Cham, 2021. Springer, Springer International Publishing. doi:10.1007/978-3-030-84259-8_18.
- 13 Daniel Escudero, Satrajit Ghosh, Marcel Keller, Rahul Rachuri, and Peter Scholl. Improved primitives for MPC over mixed arithmetic-binary circuits. In *Advances in Cryptology—CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part II 40*, pages 823–852, Cham, 2020. Springer, Springer International Publishing. doi:10.1007/978-3-030-56880-1_29.
- 14 Niv Gilboa and Yuval Ishai. Compressing cryptographic resources. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 591–608. Springer, 1999. doi:10.1007/3-540-48405-1_37.
- 15 Mitsuru Ito, Akira Saito, and Takao Nishizeki. Secret sharing scheme realizing general access structure. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 72(9):56–64, 1989.

- 16 Mauricio Karchmer and Avi Wigderson. On span programs. [1993] *Proceedings of the Eighth Annual Structure in Complexity Theory Conference*, pages 102–111, 1993. doi:10.1109/SCT.1993.336536.
- 17 Ilan Komargodski, Moni Naor, and Eylon Yogev. How to share a secret, infinitely. *IEEE Trans. Inf. Theory*, 64(6):4179–4190, 2018. doi:10.1109/TIT.2017.2779121.
- 18 Ilan Komargodski and Anat Paskin-Cherniavsky. Evolving secret sharing: Dynamic thresholds and robustness. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography*, pages 379–393, Cham, 2017. Springer International Publishing. doi:10.1007/978-3-319-70503-3_12.
- 19 Eleftheria Makri, Dragos Rotaru, Frederik Vercauteren, and Sameer Wagh. Rabbit: Efficient comparison for secure multi-party computation. In *International Conference on Financial Cryptography and Data Security*, pages 249–270, Berlin, Heidelberg, 2021. Springer, Springer Berlin Heidelberg. doi:10.1007/978-3-662-64322-8_12.
- 20 Eleftheria Makri and Tim Wood. Full-threshold actively-secure multiparty arithmetic circuit garbling. In *Progress in Cryptology–LATINCRYPT 2021: 7th International Conference on Cryptology and Information Security in Latin America, Bogotá, Colombia, October 6–8, 2021, Proceedings 7*, pages 407–430, Cham, 2021. Springer, Springer International Publishing. doi:10.1007/978-3-030-88238-9_20.
- 21 Anat Paskin-Cherniavsky and Olga Nissenbaum. New bounds and a generalization for share conversion for 3-server PIR. *Entropy*, 24(4), 2022. doi:10.3390/e24040497.
- 22 Anat Paskin-Cherniavsky and Leora Schmerler. On share conversions for private information retrieval. *Entropy*, 21(9), 2019. doi:10.3390/e21090826.
- 23 Naty Peter. Evolving conditional disclosure secrets. In *Information Security: 26th International Conference, ISC 2023, Groningen, The Netherlands, November 15–17, 2023, Proceedings*, pages 327–347, Berlin, Heidelberg, 2023. Springer-Verlag. doi:10.1007/978-3-031-49187-0_17.
- 24 Dragos Rotaru and Tim Wood. Marbled circuits: Mixing arithmetic and boolean circuits with active security. In *International Conference on Cryptology in India*, pages 227–249, Cham, 2019. Springer International Publishing. doi:10.1007/978-3-030-35423-7_12.
- 25 Adi Shamir. How to share a secret. In *Communications of the ACM*, 22, pages 612–613, 1979. doi:10.1145/359168.359176.

Key-Agreement with Perfect Completeness from Random Oracles

Noam Mazon 

Simons Institute, UC Berkeley, CA, USA

Abstract

In the Random Oracle Model (ROM) all parties have oracle access to a common random function, and the parties are limited in the number of queries they can make to the oracle. The Merkle's Puzzles protocol, introduced by Merkle [CACM '78], is a key-agreement protocol in the ROM with a quadratic gap between the query complexity of the honest parties and the eavesdropper. This quadratic gap is known to be optimal, by the works of Impagliazzo and Rudich [STOC '89] and Barak and Mahmoody [Crypto '09].

When the oracle function is injective or a permutation, Merkle's Puzzles has perfect completeness. That is, it is certain that the protocol results in agreement between the parties. However, without such an assumption on the random function, there is a small error probability, and the parties may end up holding different keys. This fact raises the question: Is there a key-agreement protocol with *perfect* completeness and super-linear security in the ROM?

In this paper we give a positive answer to the above question, showing that changes to the query distribution of the parties in Merkle's Puzzles, yield a protocol with perfect completeness and roughly the same security.

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques

Keywords and phrases Key-Agreement, Random Oracle, Merkle's Puzzles, Perfect Completeness

Digital Object Identifier 10.4230/LIPIcs.ITC.2025.12

Related Version *Full Version*: <https://eprint.iacr.org/2023/1265.pdf>

Funding Supported in part by NSF CNS-2149305 and NSF CNS-2128519. This work was done while being at Cornell Tech.

Acknowledgements We thank Tal Yankovitz and Jad Silbak for very useful discussions.

1 Introduction

Key-Agreement ([9]) is a fundamental primitive in modern cryptography. Key-Agreement protocols allow two parties, Alice and Bob, to agree on a secret key that is hidden from any eavesdropper, just by *publicly* communicating with each other. While the above is one of the most important tasks in cryptography, still there are a respectively small number of assumptions from which we know how to construct such protocols ([26, 24, 9, 11, 21, 1, 25, 6, 2, 3, 27]). Moreover, these assumptions are more structured compared to the assumptions which are used to base private-key cryptography, making them more vulnerable to attacks, and arguably less reliable.

The Random Oracle Model

An important alternative to such assumptions is the use of key-agreement protocols with proven security in the *Random Oracle Model* (ROM). In this model we assume that the parties (and the eavesdropper) have oracle access to an idealized hash function – a perfectly random function $F: [N] \rightarrow [M]$. Then, the efficiency of the eavesdropper is measured with respect to the number of oracle calls it needs to make in order to break the security.



© Noam Mazon;

licensed under Creative Commons License CC-BY 4.0

6th Conference on Information-Theoretic Cryptography (ITC 2025).

Editor: Niv Gilboa; Article No. 12; pp. 12:1–12:11



Leibniz International Proceedings in Informatics

LIPIcs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Unfortunately, while we usually want the gap between the running time of the honest parties in the protocol and the running time of the adversary to be exponential (or at least super-polynomial), this cannot be achieved in the random oracle model. Indeed, as shown by [18, 4], any protocol in which the honest parties make ℓ queries can be broken by an adversary that makes $O(\ell^2)$ queries. Yet, while this quadratic gap is not ideal, in the lack of other reliable protocols it may be sufficient (see for example [14]).

Merkle's Puzzles

The above quadratic gap is achievable, using a very elegant and simple protocol, called Merkle's Puzzles [22]. In this protocol, Alice and Bob each choose ℓ random queries, $(x_1, \dots, x_\ell)$ and $(y_1, \dots, y_\ell)$ respectively, from a domain of size $O(\ell^2)$. Then, Alice and Bob use the images of the queries to find an intersecting query (which exists with high probability due to the birthday paradox). Specifically, Alice sends $F(x_1), \dots, F(x_\ell)$; Bob then search for indexes j and k such that $F(x_j) = F(y_k)$, and sends j to Alice; Lastly, Alice outputs x_j and Bob outputs y_k as their keys.

For security, it is not hard to see that an eavesdropper that wants to learn Alice's output needs to invert the function F on a random image $F(x_j)$, and thus essentially needs to query all the $O(\ell^2)$ elements in the domain of the function.

When the function F is a random function with a large enough range, the probability that the protocol succeeds, that is, results in an agreement between Alice and Bob, is approaching to 1. Indeed, an error can occur only when two distinct queries have the same image. Moreover, when using a more structured function, such as a permutation or any injective function, it follows that Merkle's Puzzles has perfect agreement probability. Yet, it is an interesting question whether there exists such a key-agreement protocol without assuming any structure on the oracle function. That is:

Is there a key-agreement protocol with perfect agreement and super-linear security in the ROM?

1.1 Our Results

In this paper, we give a positive answer to the above question. More specifically, by changing the distribution of queries in the Merkle's Puzzles protocol, we get a protocol with perfect agreement and roughly the same security.

► **Theorem 1** (Main theorem, informal). *There exists a ℓ -query key-agreement protocol with perfect agreement in the random oracle model, secure against all $o(\ell^2)$ -query adversaries.*¹

See Theorem 3 for the formal statement. As mentioned above, we get this result by changing the query distribution of Merkle's Puzzles. The main observation is that in our distribution, the parties always have a single query in common. Then, by verifying that the sets of images of the parties only intersect in (exactly) one point, the parties can be sure that the outputs are equal.²

¹ In fact, like Merkle's puzzles, our protocol is a two-message protocol. Therefore, it can be used as a *public-key encryption*.

² Choosing a distribution such that Alice and Bob always have an intersection is not hard. Indeed, for $F: [\ell] \times [\ell] \rightarrow [M]$, consider the distribution in which Alice samples a random index $i \leftarrow [\ell]$, and query $F(i, 1), \dots, F(i, \ell)$, while Bob queries $F(1, j), \dots, F(\ell, j)$ for a random $j \leftarrow [\ell]$. It is not hard to see that $F(i, j)$ is always in the intersection. However, if the parties continue as in Merkle's Puzzles (namely, Alice sends $F(i, 1), \dots, F(i, \ell)$, and Bob finds the intersection), an eavesdropper can learn *all* the queries made by Alice, just by simulating Bob.

The protocol

We next describe the query distribution we consider. For simplicity, we think of F as a function over the domain $[\ell] \times [\ell]$. To choose her queries, Alice samples ℓ uniformly random points $x_1, \dots, x_\ell \in [\ell]$. Then, Alice queries $(1, x_1), \dots, (\ell, x_\ell)$. On the other side, Bob chooses a random index $j \in [\ell]$, and queries $(j, 1), \dots, (j, \ell)$. This distribution guarantees that (j, x_j) is a (unique) intersection between Alice's and Bob's queries.

The protocol proceeds similarly to Merkle's Puzzles: Alice sends $F(1, x_1), \dots, F(\ell, x_\ell)$ to Bob, but in a random order. Bob then searches for an intersection between $F(1, x_1), \dots, F(\ell, x_\ell)$ and his own images, $F(j, 1), \dots, F(j, \ell)$. Crucially, if Bob finds more than one intersection, he notifies Alice and they both output 0. Otherwise, he sends the intersection and both parties output its preimage, (j, x_j) .

The above protocol can be seen as an alternative implementation of Merkle's original "puzzles" concept in the random oracle setting (a protocol with a query distribution similar to the one considered here was also mentioned in [4] as part of such implementation). In more detail, Merkle considers the notions of puzzles, in which Alice encrypts using N puzzles pairs $\{(i, s_i)\}_{i \in [N]}$ of id i and secret s_i , and sends the encryptions to Bob in a random order. Bob then chooses one random encryption and tries to learn the underline pair (i, s_i) . It then sends the ID i to Alice, and both Alice and Bob output its related secret s_i . The idea is that while Bob only needs to solve one such challenge, Eve needs to solve roughly N to learn the secret. In the above protocol, the encryption of a pair (i, s_i) is done by simply computing $F(i, s_i)$.

Security

While the agreement of this protocol follows immediately by the fact that there is always an intersection, the security is a bit less obvious. Indeed, an adversary that tries to invert $z = F(j, x_j)$ can learn some information on j by observing the other images sent by Alice. Specifically, if the adversary finds a query (a, b) such that $F(a, b) \neq z$ appears in the communication from Alice to Bob, she learns that $j \neq a$. This can potentially rule out ℓ possible preimages. Nevertheless, we are able to show by a simple reduction that this cannot help too much.

To prove the above we consider a *balls and boxes* game, defined as follows. There are ℓ^2 boxes ordered in ℓ rows and ℓ columns. In each row, we chose a box uniformly at random and place one ball in it (all other boxes are left empty). All the balls are blue, except exactly one random ball which is red. The goal of the player is to find the red ball, by opening as few boxes as possible.

The security of Merkle's Puzzles corresponds to a similar game, in which there is only one red ball within a single random row, and there are no other (blue) balls. In this case, it is not hard to see (by the union bound) that any player that opens q boxes, succeeds in probability at most q/ℓ^2 in finding the red ball. To prove that our protocol is secure we first show that the security of the protocol is equivalent to the hardness of the above box and balls game. Then, we show that the presence of the blue balls cannot help the player too much. This is done by observing a simple reduction between the games with and without the blue balls. Specifically, we show that any player that opens q boxes, cannot win with probability larger than $2q/\ell^2$.

Comparison with Merkle’s Puzzles

Similarly to Merkle’s Puzzles, the above protocol is a two-message protocol, and has non-adaptive queries. Both protocols also have near-optimal communication complexity for protocols with these properties [14] (and can be generalized to get other possible tradeoffs between communication and security). The communication complexity of the protocol presented here is also tight with the recent lower bound of [17] for protocols with perfect completeness and non-adaptive queries.

To compare the security of Merkle’s Puzzles and the protocol given in this work, recall that to have an intersection with $1 - o(1)$ probability, the parties in Merkle’s Puzzles should query ℓ queries each from a domain $\mathcal{D}$ of size at most $o(\ell^2)$. Thus, an q -query adversary can learn the key with probability $q/|\mathcal{D}| = \omega(q/\ell^2)$. In this case, we are able to get a slightly better security due to the fact that, in the protocol presented here, there is an intersection with probability 1, when the domain of the function is of size (exactly) ℓ^2 .

One advantage of Merkle’s Puzzles is that it is non-interactive, meaning that both Alice and Bob can send their message simultaneously, without waiting to the other message to arrive. The protocol presented here does not have this property, and it is interesting question if such non-interactive protocol with perfect completeness exists.

Perspective and Additional Related Work**Perfect primitives in the ROM**

Being an idealized hash function, the random oracle model is used to prove black-box separation between cryptographic primitives. Notably, [19] prove a black-box separation between one-way permutations and one-way functions, showing that it is impossible to construct a one-way permutation from random oracles alone.

As mentioned above, [18] prove that there is no key-agreement with super-polynomial security in the ROM, and [4] prove a tight quadratic bound on the security of key-agreement protocols in the ROM. The latter result was generalized by [15] to hold for any semi-honest, no-input, two-party task. Prior to the work of [4], [8] proved a similar quadratic bound on the security of any key-agreement protocol with *perfect* completeness in the ROM. However, as mentioned by [4], before the work presented here, it was conceivable that every key-agreement protocol with perfect completeness in the ROM can be broken with a linear number of queries to the oracle.

Public-key encryption with perfect completeness

A long line of work is dedicated to the task of eliminating errors in public-key encryption schemes (or, equivalently, two-message key-agreement) and in other cryptographic primitives [12, 10, 16, 20, 5]. As mentioned in [10, 5], even a small decryption error in a public-key encryption scheme can be problematic for some uses. Moreover, as demonstrated by [23, 7], such decryption error can be used by an adversary to attack the cryptographic system.

2 Preliminaries

2.1 Notations

We use calligraphic letters to denote sets and distributions, uppercase for random variables, and lowercase for values and functions. When unambiguous, we will naturally view a random variable as its marginal distribution. For a set $\mathcal{S}$, let $x \leftarrow \mathcal{S}$ denote that x is drawn uniformly from $\mathcal{S}$. For $n \in \mathbb{N}$, let $[n] = \{1, \dots, n\}$. For two sets $\mathcal{A}, \mathcal{B}$, let $\mathcal{A} \times \mathcal{B} = \{(a, b) : a \in \mathcal{A}, b \in \mathcal{B}\}$.

2.2 Interactive Protocols and Oracle-Aided Protocols

A no-input, two-party protocol $\Pi = (A, B)$ is simply a pair of probabilistic interactive algorithms. The interaction between the parties A and B is carried out in rounds, where in each round one party is active and the other is idle. In the j -th round of the protocol, the active party sends a message to the other party, according to its partial view (its randomness and the messages sent in the protocol so far). The transcript of a given execution of the protocol is the list of messages exchanged between the parties in the execution of the protocol.

For a protocol Π , we use $(\text{trans}, \text{out}_A, \text{out}_B) \leftarrow \Pi$ to denote that $(\text{trans}, \text{out}_A, \text{out}_B)$ is a triplet sampled according to the joint distribution induced by Π , which contains a transcript of the protocol, an output of party A , and an output of party B in a random execution of Π .

An oracle-aided no-input two-party protocol $\Pi = (A, B)$ is a pair of interactive oracle-aided algorithms. For a function f , we use $\Pi^f = (A^f, B^f)$ to denote the protocol Π when using f as the oracle function. Such a protocol $\Pi = (A, B)$ is a ℓ -query protocol, if for every function f , each party always queries f on at most ℓ points during the interaction between A^f and B^f .

2.3 Key-Agreement Protocols

We now define key-agreement protocols with respect to a function family $\mathcal{F}$. In the random oracle model, we choose $\mathcal{F}$ to be the all-function family, over some fixed domain and range.

► **Definition 2** (Key-agreement protocol). *A no-input oracle-aided two-party protocol (A, B) is a (q, α, γ) key-agreement protocol with respect to a function family $\mathcal{F}$ if the following holds:*

1. Agreement:

$$\Pr_{f \leftarrow \mathcal{F}, (\text{trans}, \text{out}_A, \text{out}_B) \leftarrow \Pi^f} [\text{out}_A = \text{out}_B] \geq 1 - \alpha.$$

2. Security: *For every algorithm E , making at most q queries, it holds that*

$$\Pr_{f \leftarrow \mathcal{F}, (\text{trans}, \text{out}_A, \text{out}_B) \leftarrow \Pi^f} [E^f(\text{trans}) = \text{out}_A] \leq \gamma.$$

Note that the above security definition requires that the adversary will not be able to guess the key, but the adversary may know some partial information on the output of A (for example, the first bit). Given such a security guarantee, the parties can amplify it to get a stronger security notion, in which the key is (somewhat) indistinguishable from random in the eyes of the eavesdropper. This can be done by either extracting randomness from the outputs (for example, using the Goldreich-Levin hardcore predicate [13]), or using the random oracle, by querying the common output and using its image as the key.³

³ A problem can occur in the last approach if the image of the key is used in the protocol, as happens in Merkle's Puzzles. In this case, the parties can, for example, use only the prefix of the images in the protocol, and use the suffix as the secret key.

Let $\mathcal{D}$ and $\mathcal{R}$ be two sets, and let $\mathcal{F} = \{f \mid f: \mathcal{D} \rightarrow \mathcal{R}\}$ be the all-function family over the domain $\mathcal{D}$ and the range $\mathcal{R}$. When $|\mathcal{D}| = \Omega(\ell^2)$, and $|\mathcal{R}| \gg \ell^2$, the Merkle's Puzzles protocol, mentioned in the introduction, is a ℓ -query, $(q, 0.01, O(q/\ell^2))$ key-agreement protocol with respect to $\mathcal{F}$, and for every $q \in \mathbb{N}$.

3 The Key-Agreement Protocol

We now describe our key-agreement protocol. Let $\ell, M \in \mathbb{N}$ be two parameters, and let $\mathcal{F}_{\ell, M} = \{f \mid f: [\ell^2] \rightarrow [M]\}$ be the family of all functions from $[\ell^2]$ to $[M]$. For simplicity, we think of the domain of every $f \in \mathcal{F}_{\ell, M}$ as $[\ell] \times [\ell]$. Namely, we think of every $f \in \mathcal{F}_{\ell, M}$ as $f: [\ell] \times [\ell] \rightarrow [M]$.

► **Theorem 3.** *The following holds for every $\ell, M \in \mathbb{N}$. There exists an ℓ -query two-party protocol, which is a $(q, 0, \frac{2(q+1)}{\ell^2} + \frac{2\ell^2}{M})$ key-agreement protocol with respect to $\mathcal{F}_{\ell, M}$, for every $q \in \mathbb{N}$.*

The protocol is as follows.

► **Protocol 4.**

Parameters: $\ell, M \in \mathbb{N}$.

Common function: $f: [\ell] \times [\ell] \rightarrow [M]$.

Operation:

1. Party A samples $x_1, \dots, x_\ell \leftarrow [\ell]$ uniformly and independently at random, and queries $(1, x_1), \dots, (\ell, x_\ell)$. Let $a_i = f(i, x_i)$ for every $i \in [\ell]$.
2. Party B samples an index $j \leftarrow [\ell]$ uniformly at random, and queries $(j, 1), \dots, (j, \ell)$. Let $b_i = f(j, i)$ for every $i \in [\ell]$.
3. A sends $a_1, \dots, a_\ell$ in a random order.
4. B checks if the number of different images in $\{a_1, \dots, a_\ell, b_1, \dots, b_\ell\}$ is exactly $2\ell - 1$ (that is, if there is exactly one collision). If not, B notifies A, and both parties output 0. Otherwise, B finds the unique $z \in \{a_1, \dots, a_\ell\} \cap \{b_1, \dots, b_\ell\}$, and sends z to A.
5. Party A outputs (i, x_i) for the unique $i \in [\ell]$ for which $f(i, x_i) = z$. B outputs (j, k) for the unique $k \in [\ell]$ for which $f(j, k) = z$.

We turn to analyze the agreement and security of Protocol 4. Towards this, fix the parameters M and ℓ , and let $\mathcal{F} = \mathcal{F}_{\ell, M}$. Let $F \leftarrow \mathcal{F}$ be a random variable representing a random function from $\mathcal{F}$, and let O_A and O_B be the outputs of A and B, respectively, in a random execution of Protocol 4 using the function F . Finally, Let T be the transcript of the protocol in this execution. Theorem 3 is a direct implication of the following two claims. The first states that Protocol 4 has perfect agreement.

▷ **Claim 5 (Agreement).**

$$\Pr[O_A = O_B] = 1.$$

The second claim establishes the security of the protocol.

▷ **Claim 6 (Security).** For every q -query algorithm E, it holds that

$$\Pr[E^F(T) = O_A] \leq \frac{2(q+1)}{\ell^2} + \frac{2\ell^2}{M}.$$

We prove Claim 5 and Claim 6 below, but first let us conclude the proof of Theorem 3.

Proof of Theorem 3. The proof follows by Claim 5 and Claim 6, together with the observation that the parties in Protocol 4 make ℓ queries each. ◀

3.1 Agreement – Proving Claim 5

We start by showing that Protocol 4 has perfect agreement.

Proof of Claim 5. Claim 5 follows immediately by the query distribution of the parties. Specifically, observe that for every sampling of the ℓ points of A , $x_1, \dots, x_\ell$, and for every sampling of j , A and B have a common query, (j, x_j) . Thus, the parties have at least one common image. If further there are no other collisions in the union of the two sets of images, then $O_A = O_B = (j, x_j)$.

On the other hand, when there are other collisions in the union of the sets, $O_A = O_B = 0$. Overall, we get that $O_A = O_B$, with certainty, as wanted. $\triangleleft$

3.2 Security – Proving Claim 6

In this part we prove that Protocol 4 has roughly the same security as Merkle's Puzzles. We do it by defining simplified security games, and showing a reduction between the security of Protocol 4 and these games.

Balls and Boxes games

To analyze the security of Protocol 4, in the following we define two “balls and boxes” games, to which we refer as the “unique-ball” game, and the “multi-ball” game. Intuitively, breaking the security of Merkle's Puzzles is equivalent to winning the unique-ball game, and, on the other hand, we will show that breaking the security of Protocol 4 is equivalent to winning the multi-ball game. In the following we define these two games, and show that every strategy for the multi-ball game is an (almost) as good strategy for the unique-ball game. This will imply that Protocol 4 has security similar to the security of Merkle's Puzzles.

We start with defining the unique-ball game.

► **Definition 7** (The unique-ball game). *Let $\ell, q \in \mathbb{N}$ be two parameters. In the (ℓ, q) -unique-ball game, there are ℓ^2 boxes ordered in ℓ rows and ℓ columns. All the boxes are empty, except for one box, chosen uniformly at random, that contains a red ball in it.*

The player is allowed to (adaptively) open up to q boxes. The player wins the game if it finds the ball.

It is not hard to see that for every strategy of the player, the probability of finding the ball is at most q/ℓ^2 . We now define the multi-ball game.

► **Definition 8** (The multi-ball game). *Let $\ell, q \in \mathbb{N}$ be two parameters. In the (ℓ, q) -multi-ball game, there are ℓ^2 boxes ordered in ℓ rows and ℓ columns. In each row, there is a single box chosen uniformly and independently at random, that contains a ball in it. All the other boxes are empty. The ball in one row, chosen uniformly at random, is red, and the other balls are blue.*

The player is allowed to (adaptively) open up to q boxes. The player wins the game if it finds the red ball.

That is, as in the unique-ball game, the goal of the player is to find the box that contains the red ball, out of ℓ^2 possible boxes. However, this time when the player finds a blue ball, it knows that the red ball is not in this row, and thus can stop opening boxes in this row. Intuitively, this corresponds to the images A sends in Protocol 4, which can help the adversary to rule out some possible indices. The connection between the security of Protocol 4 to the multi-ball game is stated in the following claim.

▷ **Claim 9.** Assume there exists a q -query algorithm E such that $\Pr[E^F(T) = O_A] = p$. Then there exists a player P that wins the $(\ell, q + 1)$ -multi-ball game with probability at least $p - \frac{2\ell^2}{M}$.

We prove Claim 9 in Section 3.3, but first let us use it to prove Claim 6. To do so, we need to prove an upper bound on the winning probability of every strategy for the multi-ball game. This is done by showing that every strategy to the multi-ball game can be used in the unique-ball game with roughly the same success probability. Formally,

▷ **Claim 10.** Assume there exists a strategy for the (ℓ, q) -multi-ball game that wins with probability p . Then there exists a strategy for the (ℓ, q) -unique ball game with winning probability $p/2$.

Proof of Claim 10. Let P be a player for the (ℓ, q) -multi-ball game, that wins with probability p . We show how to use P to construct P' that wins in the (ℓ, q) -unique-ball game with probability $p/2$.

Given ℓ^2 boxes, P' samples $x_1, \dots, x_\ell \leftarrow [\ell]$ uniformly and independently at random. Informally, P' will “plant” a blue ball in the box at location (i, x_i) for every $i \in [\ell]$, and then run P . More formally, P' simulates the game for P . Every time P chooses to open a box (i, j) , P' opens the same box. If this box contains the red ball, P' wins. If this box is empty and $j = x_i$, P' reports to P that the box contains a blue ball. Otherwise, P' reports to P that the box is empty.

Observe that P' wins if and only if P finds the red ball in the simulated game.⁴ To see that P wins the simulated game with probability at least $p/2$, first observe that in the simulated game, there is one row i^* that contains two balls (one red ball and one blue ball, possibly at the same box), and all the other rows contain exactly one (blue) ball. Notice that if both of the balls in row i^* were red, the winning probability of P in the simulated game was at least p (as we only added a red ball to the grid). That is, P finds (at least) one ball on row i^* with probability at least p . By symmetry, the probability that the first ball in row i^* found by P is indeed the true red ball is $1/2$. Thus, the winning probability of P is at least $p/2$, as stated. ◁

As a direct corollary from Claim 5 and the fact that there is no strategy for the unique-ball game with a winning probability better than q/ℓ^2 , we get that every strategy for the (ℓ, q) -multi-ball game wins with probability at most $2q/\ell^2$.

► **Corollary 11.** *The winning probability of any strategy for the (ℓ, q) -multi-ball game is at most $2q/\ell^2$.*

We can now conclude Claim 6.

Proof of Claim 6. Immediately follows by Corollary 11 and Claim 9. ◁

3.3 Proving Claim 9

Lastly, we show that an adversary E that makes q queries to F and breaks Protocol 4 with probability p , can be used to win the $(\ell, q + 1)$ -multi-ball game with roughly the same probability.

⁴ Notice that P may find ℓ blue balls in the simulated game, and this cannot happen in the real multi-ball game. However, this event is not counted as a win.

To prove Claim 9, consider the following player P that plays the multi-ball game by simulating E on a random function F and transcript T . In the following, assume without loss of generality that E never asks the same query twice. Moreover, at the cost of one additional query, assume that E always queries F on its output.

► **Algorithm 12** (P).

Parameters: $\ell, M \in \mathbb{N}$.

Operation:

1. P samples $a_1, \dots, a_\ell \leftarrow [M]$ uniformly and independently at random, and $j \leftarrow [\ell]$.
2. P simulates E on the transcript $\text{trans} = ((a_1, \dots, a_\ell), z = a_j)$. Every time E makes a query (i, j) , P opens the box at location (i, j) . P answers to E 's query as follows:
 - If the box contains a red ball, P wins and halts.
 - If the box is empty, P answers with a random element from $[M]$.
 - If the box contains a blue ball, P answers with a random element from $\{a_1, \dots, a_\ell\} \setminus \{a_j\}$ that was not in use before.

We are now ready to prove Claim 9.

Proof of Claim 9. Let E be a $(q+1)$ -query algorithm that, given the transcript of Protocol 4, queries the function F on the output O_A of A with probability at least p . Let P be Algorithm 12.

We start with considering a modified version of Protocol 4 (which cannot be implemented as a protocol), in which B does not notify A in step 4 in case B sees a collision in the function f . Instead, in this case we assume that B sends the correct image $z = f(j, x_j)$, and that A outputs the correct intersection (j, x_j) .

As the above modified version and Protocol 4 differ only in the case that there is a collision in f , it is not hard to see that in this modified version, E guesses the output of party A with probability at least

$$\begin{aligned}
 & p - \Pr[F(q) = F(q') \text{ for two distinct queries } q \neq q' \text{ of } A \text{ and } B] \\
 & \geq p - \binom{2\ell}{2} \cdot \frac{1}{M} \\
 & \geq p - \frac{2\ell^2}{M},
 \end{aligned}$$

where the first inequality holds by the union bound, since every two queries have probability $1/M$ to collide, and there are at most 2ℓ queries made by the parties.

To see that the success probability of Algorithm 12 is $p - \frac{2\ell^2}{M}$, note that it readily follows by inspection that the distribution seen by E in a random execution of Algorithm 12 on a random multi-ball game, is exactly the same distribution seen by E in a random execution of the above modified protocol. Moreover, Algorithm 12 finds the red ball in the multi-ball game if and only if E queries the secret key in the modified protocol. The claim follows. $\triangleleft$

References

- 1 Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *stoc29*, pages 284–293, 1997. See also ECCC TR96-065. doi:10.1145/258533.258604.
- 2 Michael Alekhnovich. More on average case vs approximation complexity. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 298–307. IEEE, 2003. doi:10.1109/SFCS.2003.1238204.

- 3 Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 171–180, 2010. doi:10.1145/1806689.1806715.
- 4 Boaz Barak and Mohammad Mahmoody. Merkle puzzles are optimal – An $o(n^2)$ -query attack on any key exchange from a random oracle. In *Annual International Cryptology Conference*, pages 374–390. Springer, 2009.
- 5 Nir Bitansky and Vinod Vaikuntanathan. A note on perfect correctness by derandomization. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 592–606. Springer, 2017. doi:10.1007/978-3-319-56614-6_20.
- 6 Andrej Bogdanov, Miguel Cueto Noval, Charlotte Hoffmann, and Alon Rosen. Public-key encryption from homogeneous clwe. In *Theory of Cryptography: 20th International Conference, TCC 2022, Chicago, IL, USA, November 7–10, 2022, Proceedings, Part II*, pages 565–592. Springer, 2022. doi:10.1007/978-3-031-22365-5_20.
- 7 Dan Boneh, Richard A DeMillo, and Richard J Lipton. On the importance of eliminating errors in cryptographic computations. *Journal of cryptology*, 14:101–119, 2001. doi:10.1007/S001450010016.
- 8 Zvika Brakerski, Jonathan Katz, Gil Segev, and Arkady Yerukhimovich. Limits on the power of zero-knowledge proofs in cryptographic constructions. In *Theory of Cryptography: 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28–30, 2011. Proceedings 8*, pages 559–578. Springer, 2011. doi:10.1007/978-3-642-19571-6_34.
- 9 Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, pages 644–654, 1976. doi:10.1109/TIT.1976.1055638.
- 10 Cynthia Dwork, Moni Naor, and Omer Reingold. Immunizing encryption schemes from decryption errors. In *Advances in Cryptology – EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2–6, 2004. Proceedings 23*, pages 342–360. Springer, 2004. doi:10.1007/978-3-540-24676-3_21.
- 11 Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Annual International Cryptology Conference (CRYPTO)*, pages 10–18, 1984.
- 12 Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Eliminating decryption errors in the ajtai-dwork cryptosystem. In *Advances in Cryptology – CRYPTO’97: 17th Annual International Cryptology Conference Santa Barbara, California, USA August 17–21, 1997 Proceedings 17*, pages 105–111. Springer, 1997. doi:10.1007/BFB0052230.
- 13 Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing (STOC)*, pages 25–32, 1989. doi:10.1145/73007.73010.
- 14 Iftach Haitner, Noam Mazon, Rotem Oshman, Omer Reingold, and Amir Yehudayoff. On the communication complexity of key-agreement protocols. *10th Innovations in Theoretical Computer Science*, 2019. doi:10.4230/LIPIcs.ITCS.2019.40.
- 15 Iftach Haitner, Eran Omri, and Hila Zarosim. Limits on the usefulness of random oracles. *Journal of Cryptology*, 29(2):283–335, 2016. doi:10.1007/S00145-014-9194-9.
- 16 Thomas Holenstein and Renato Renner. One-way secret-key agreement and applications to circuit polarization and immunization of public-key encryption. In *Annual International Cryptology Conference*, pages 478–493. Springer, 2005. doi:10.1007/11535218_29.
- 17 Mi-Ying Miryam Huang, Xinyu Mao, Guangxu Yang, and Jiapeng Zhang. Communication lower bounds of key-agreement protocols via density increment arguments. *Cryptology ePrint Archive*, 2023. URL: <https://eprint.iacr.org/2023/1349>.
- 18 Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing (STOC)*, pages 44–61. ACM Press, 1989. doi:10.1145/73007.73012.
- 19 Jeff Kahn, Michael Saks, and Cliff Smyth. A dual version of Reimer’s inequality and a proof of Rudich’s conjecture. In *15th Annual IEEE Conference on Computational Complexity*, pages 98–103, 2000. doi:10.1109/CCC.2000.856739.

- 20 Huijia Lin and Stefano Tessaro. Amplification of chosen-ciphertext security. In *Advances in Cryptology—EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26–30, 2013. Proceedings 32*, pages 503–519. Springer, 2013. doi:10.1007/978-3-642-38348-9_30.
- 21 Robert J McEliece. A public-key cryptosystem based on algebraic. *Coding Thv*, 4244:114–116, 1978.
- 22 Ralph C Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21(4):294–299, 1978. doi:10.1145/359460.359473.
- 23 John Proos. Imperfect decryption and an attack on the ntru encryption scheme. *Cryptology ePrint Archive*, 2003. URL: <https://eprint.iacr.org/2003/002>.
- 24 Michael O Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, Massachusetts Inst of Tech Cambridge Lab for Computer Science, 1979.
- 25 Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009. doi:10.1145/1568318.1568324.
- 26 Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. doi:10.1145/359340.359342.
- 27 Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999. doi:10.1137/S0036144598347011.

