5th Conference on Information-Theoretic Cryptography

ITC 2024, August 14–16, 2024, Stanford, CA, USA

Edited by

Divesh Aggarwal

LIPICS

LIPIcs – Vol. 304 – ITC 2024 www.dagstuhl.de/lipics

Editors

Divesh Aggarwal
National University of Singapore, Singapore
divesh@comp.nus.edu.sg

ACM Classification 2012
Mathematics of computing → Information theory; Theory of computation → Computational complexity and cryptography; Security and privacy → Cryptography

ISBN 978-3-95977-333-1

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/978-3-95977-333-1.

Publication date
August, 2024

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

Digital Object Identifier: 10.4230/LIPIcs.ITC.2024.0

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.

ISSN 1868-8969

https://www.dagstuhl.de/lipics

Contents

Preface

The fifth Conference on Information-Theoretic Cryptography (ITC 2024) took place from August 14–16, 2024, at Stanford University, USA. The general chairs were Mary Wootters and Dan Boneh, and the program chair was Divesh Aggarwal. As in previous editions, the conference was held in cooperation with the International Association for Cryptologic Research (IACR).

In its fifth 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. This year, we introduced a new Highlights Track, aimed at showcasing outstanding recent results from other venues.

We received a total of 21 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 11 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" that highlight exciting developments in the field. Additionally, the newly introduced Highlights Track featured invited talks on notable recent papers from top conferences such as STOC 2024, Eurocrypt 2024, FOCS 2023, Crypto 2023, Asiacrypt 2023, TCC 2023, and STOC 2023 presented by students or postdocs. These tracks aimed to provide a comprehensive overview of the most significant advancements in information-theoretic cryptography.

We are deeply grateful to everyone who contributed to the success of the 5th 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 Kai-Min Chung and Stefano Tessaro, 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.

Divesh Aggarwal

Steering Committee

Organization

General chairs

- Mary Wootters (Stanford University)
- Dan Boneh (Stanford University)

Program chair

- Divesh Aggarwal (National University of Singapore)

Program Committee

- Akshayaram Srinivasan (University of Toronto)
- Amos Beimel (Ben Gurion University)
- Damiano Abram (Aarhus University)
- Daniele Venturi (Sapienza University of Rome)
- Giulio Malavolta (Bocconi University & MPI-SP)
- Hemanta Maji (Purdue University)
- Ilan Komargodski (The Hebrew University of Jerusalem and NTT Research)
- Jesse Goodman (UT Austin)
- Joao Ribeiro (Universidade Nova de Lisboa)
- Maciej Obremski (National University of Singapore)
- Manoj Prabhakaran (IIT Bombay)
- Mark Simkin (Ethereum Foundation)
- Mingyuan Wang (UC Berkeley)
- Mor Weiss (Bar-Ilan University)
- Mukul Kulkarni (TII Abu Dhabi)
- Noah Stephens-Davidowitz (Cornell University)
- Noam Mazor (Tel Aviv University)
- Sruthi Sekar (UC Berkeley)
- Srijita Kundu (University of Waterloo)
- Tianren Liu (Peking University)
- Xin Li (Johns Hopkins University)
- Yiannis Tselekounis (Royal Holloway University of London)

External Reviewers

Albert Yu, Alexander Bienstock, Eldon Chung, Hamidreza Amini Khorasgani, Hannah Keller, Naty Peter, Pedro Branco, Seunghoon Lee, Suparno Ghoshal, Varun Narayanan, Wei Cheng, Xiuyu Ye, Zeyong Li.

Information-Theoretic Topology-Hiding Broadcast: Wheels, Stars, Friendship, and Beyond

D'or Banoun
Reichman University, Herzliya, Israel

Elette Boyle
Reichman University, Israel
NTT Research, Sunnyvale, CA, USA

Ran Cohen
Reichman University, Herzliya, Israel

Abstract

Topology-hiding broadcast (THB) enables parties communicating over an incomplete network to broadcast messages while hiding the network topology from within a given class of graphs. Although broadcast is a privacy-free task, it is known that THB for certain graph classes necessitates computational assumptions, even against "honest but curious" adversaries, and even given a single corrupted party. Recent works have tried to understand when THB can be obtained with *information-theoretic* (IT) security (without cryptography or setup assumptions) as a function of properties of the corresponding graph class.

We revisit this question through a case study of the class of *wheel* graphs and their subgraphs. The n^{th} wheel graph is established by connecting n nodes who form a cycle with another "center" node, thus providing a natural extension that captures and enriches previously studied graph classes in the setting of IT-THB.

We present a series of new findings in this line. We fully characterize feasibility of IT-THB for any class of subgraphs of the wheel, each possessing an embedded *star* (i.e., a well-defined center connected to all other nodes). Our characterization provides evidence that IT-THB feasibility may correlate with a more fine-grained degree structure – as opposed to pure connectivity – of the corresponding graphs. We provide positive results achieving *perfect* IT-THB for new graph classes, including ones where the number of nodes is unknown. Further, we provide the first feasibility of IT-THB on non-degenerate graph-classes with $t > 1$ corruptions, for the class of *friendship* graphs (Erdös, Rényi, Sós'66).

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques; Theory of computation → Cryptographic protocols

Keywords and phrases broadcast, topology-hiding protocols, information-theoretic security

Digital Object Identifier 10.4230/LIPIcs.ITC.2024.1

Funding *D'or Banoun*: Supported in part by AFOSR Award FA9550-21-1-0046.
Elette Boyle: Supported in part by AFOSR Award FA9550-21-1-0046 and ERC Project HSS (852952).
Ran Cohen: Supported in part by NSF grant no. 2055568, by ISF grant 1834/23, and by the Algorand Centres of Excellence programme managed by Algorand Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Algorand Foundation.

1 Introduction

Topology-hiding protocols over an incomplete communication network guarantee that colluding parties do not learn additional information about the topology of the network graph (from within a given class of graphs), beyond their own neighbor-set [12]. Such protocols may be of interest in settings where the communication structure itself is sensitive information, such as in social networks, or peer-to-peer networks based on geographical position. Perhaps the most fundamental goal is that of achieving topology-hiding *broadcast* (THB), where a designated sender wishes to convey an input to all participating parties.

Although broadcast is a privacy-free task, THB turned out to be a challenging goal on its own. It was recently shown that THB for certain graph classes necessitates computational assumptions, even in the "honest but curious" *semi-honest* setting (when corrupted parties follow the protocol honestly but try to learn more information from their joint view), and even given a *single* corrupted party [6, 5]. This lies in stark contrast to the topology-revealing case, in which broadcast is trivially achievable in the semi-honest setting.

Obtaining topology hiding based on computational assumptions has been the subject of a fruitful collection of works, leading to various THB, and in turn, general topology-hiding secure multiparty computation (THC) protocols [12, 8, 2, 1, 9, 6, 10, 11, 3]. It is known by now how to construct THB protocols for the class of all graphs (of polynomial size) that are secure against *any* subset of semi-honest corruptions under standard number-theoretic cryptographic hardness assumptions such as DDH, QR, and LWE,[1] or from unstructured assumptions such as constant-round constant-rate oblivious transfer [3].

Motivated by an analogous question within secure multi-party computation, the work of [5] asked whether existence of an honest majority can enable *information-theoretically* secure THB protocols in certain settings, without relying on cryptographic assumptions and withstanding computationally unbounded adversaries. We refer to this as IT-THB. The work of [5] ruled out 1-secure IT-THB on a path with four nodes (which is 1-connected) but devised a perfect 1-secure information-theoretic THC on cycles of known length (which are 2-connected); see Figure 1. Given these initial evidence, they conjectured that feasibility of IT-THB may depend on the *connectivity*[2] of the graphs within the class: namely, that $(t+1)$-connectivity is sufficient and/or necessary for t-secure IT-THB.

The special case of $t = 1$ was further investigated by [4], who proved that the conjecture holds in this case for the stronger notion of THC. They showed that information-theoretic THC with security against a single semi-honest corruption is possible if and only if the connectivity of every graph in the class is at least 2. However, they additionally showed that the conjecture does *not* hold for THB, by constructing a perfectly secure THB against a single corruption for the butterfly graph class (Figure 1), where each graph is only 1-connected.

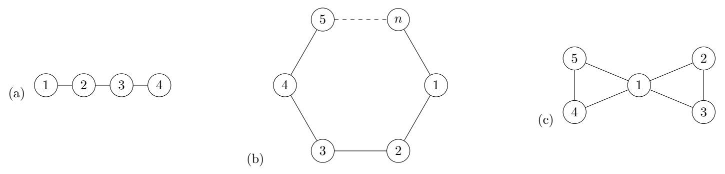


Figure 1 (a) Class $\mathcal{G}_{\text{4-path}}$, of all isomorphisms of 4 nodes on a path; 1-secure THB over $\mathcal{G}_{\text{4-path}}$ implies key agreement. (b) Class $\mathcal{G}_{\text{cycle}}(n)$ of all isomorphisms of n nodes on a cycle; admits 1-secure IT-THB. (c) Class $\mathcal{G}_{\text{butterfly}}$ of all isomorphisms of 5 nodes on a butterfly graph (two triangles with a common node); contains 1-connected graphs yet admits 1-secure IT-THB.

The results of [5, 4] open a rich domain of questions. As [4] showed, high connectivity is not the "right" criterion for feasibility of THB (in contrast to THC), and alternative graph-properties may serve as candidate conjectures. Therefore, our first question is:

Given a graph-class, which graph properties characterize feasibility of 1*-secure IT-THB?*

[1] DDH stands for the decisional Diffie-Hellman assumption, QR for the quadratic residuosity assumption, and LWE for the learning with errors assumption.

[2] We consider node-connectivity; that is, a graph is k-connected if and only if every pair of nodes is connected by k *vertex-disjoint* paths.

Zooming into [4], their general positive result, of 1-IT-THB over 2-connected graphs, has a nonzero (yet exponentially small) error probability. This means that THB with *perfect* security is only known for cycles [5], for the butterfly graph [4], and for graphs with at most four nodes [4]. Is it possible that for graphs with $n > 5$ nodes the source of perfect 1-THB is the highly symmetric structure of cycles, and other graph classes inherently require a positive error?

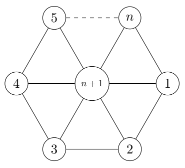
Are there additional graph-classes that support perfectly secure THB?

Finally, all feasibility results for IT-THB are secure against a single corruption. Indeed, 2-secure THB on a 4-node rectangle, possibly with a missing edge, requires oblivious transfer [6], and a 2-secure THB on a cycle with 7 nodes (or more) requires key agreement [5]. The statistically secure THB protocols for 2-connected graphs from [4] completely break if there are two corruptions, and in the butterfly class two corruptions trivialize the problem, as there is no information to hide. One may wonder if IT-THB simply cannot withstand multiple corruptions that provide several points of view about the graph topology, except for degenerate cases where the topology is already revealed by the corrupted parties' neighbor-sets. This leads to our third question:

Are there graph-classes that support IT-THB with more than a single corruption?

1.1 Our Contributions

In this work, we conduct an investigation of these questions through a case study of the class of *wheel graphs* and their subgraphs. The n^{th} wheel graph W_n is established by connecting a single node (the "center") to n nodes who form a cycle, as depicted below. The wheel graph-class $\mathcal{G}_{\mathsf{wheel}}(n)$ consists of all isomorphisms of the wheel graph, i.e., all assignments of the labels $\{1, \ldots, n+1\}$ to the nodes of the wheel graph W_n.



Wheel graphs and their subgraphs form a natural extension that captures and enriches previously studied graph classes in the setting of IT-THB: for example, paths, cycles, triangles, and butterfly graphs. Interestingly, although $\mathcal{G}_{\mathsf{wheel}}(n)$ has increased connectivity over the n-cycles, the corresponding state-of-the-art THB protocols for $\mathcal{G}_{\mathsf{wheel}}(n)$ are slightly worse. Note that the cycle protocol cannot simply be run directly, as parties on the perimeter of the graph do not know – in fact, must not know – which neighbor is the center node.

Several challenges arise when hiding the topology of $\mathcal{G}_{\mathsf{wheel}}(n)$. First, consider a node v on the perimeter; such a node has three neighbors, one of which is the center. To hide the identity of the center node, either the protocol does not utilize the power of the center, or each of the non-center neighbors must emulate the behavior of the center toward v, and further, v must emulate the center toward all its neighbors. Second, consider the center node; this node is connected to all other parties but must not learn how the parties on the perimeter are connected among themselves. Further, an adversary that corrupts two parties on the perimeter without a common neighbor must not learn their relative distance on the perimeter. Note that an adversary that corrupts $n-2$ nodes in a wheel graph knows the entire topology from the corrupted nodes' neighbor-sets; however, for $t \leq n-3$ corruptions not all is revealed (i.e., when there are 4 honest parties).[3]

[3] When considering arbitrary admissible graphs with $n+1$ nodes (as defined below), there is more information to hide; therefore, an adversary that corrupts n nodes knows the entire topology but for

Characterization of wheels and subgraphs with an embedded star. Our first result shows that perfectly secure THB is possible against a single semi-honest corruption on the class of wheel graphs $\mathcal{G}_{\mathsf{wheel}}(n)$, as well as on certain classes of its subgraphs. Concretely, given any family of subgraphs of the wheel with $n+1$ nodes, with an embedded star in each graph (i.e., where the center is fully connected and has degree n), we show that IT-THB with one corruption is possible if either the *minimal degree* of non-center nodes in the family is greater than 1, or if it is 1 but so is the *maximal degree*. Surprisingly, we show that this characterization is tight for any such subclasses that are closed under isomorphism (i.e., for each graph topology in the class, all relabelings of this graph are also contained in the class); that is, if the maximal degree is greater than 1 but the minimal degree is 1, then THB on this class implies key agreement.

This would suggest that feasibility of IT-THB may correlate with a more fine-grained degree structure, as opposed to connectivity, of graphs.

More concretely, we begin by defining *admissible subgraphs* as subgraphs of the wheel graph W_n in which the degree of the center is n and the degree of every other node is either 2 or 3. The *butterfly graph* is an example for an admissible subgraph for $n = 4$, as well as the $(2n+1)$-node *friendship graph* F_n,[4] see Figure 2.

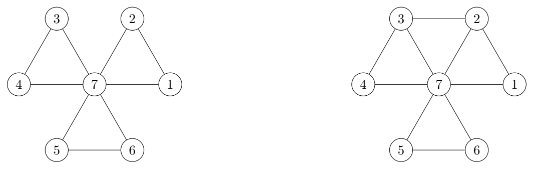


Figure 2 Examples of admissible subgraphs of $\mathcal{G}_{\mathsf{wheel}}(6)$. On the left is a friendship graph in which every non-center node has degree 2, and on the right is a subgraph where every non-center node has degree 2 or 3.

When considering graphs with an embedded star, i.e., with a fully connected center, *non-admissible graphs* are those who contain a non-center node of degree 1. The extreme example is the *star graph* in which the center node is connected to n nodes, and no other edges exist, see Figure 3.

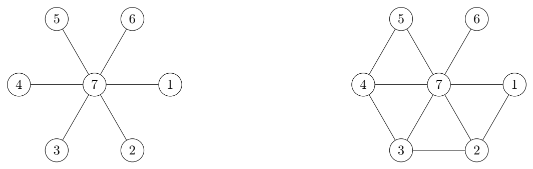


Figure 3 Example of non-admissible subgraphs of $\mathcal{G}_{\mathsf{wheel}}(6)$. On the left is the star graph with 7 nodes. On the right is a subgraph with a single node of degree 1.

Our characterization nearly shows that IT-THB is possible for a given graph-class with a fully connected center if and only if it consists only of admissible subgraphs. The single exception is the graph class $\mathcal{G}_{\mathsf{star}}(n)$ that only contain star graphs, which are not admissible; this class is degenerate (trivially providing topology hiding) since any node can identify the center and derive the whole topology.

$t \leq n-2$ not all is revealed (i.e., when there are 2 honest parties) .

[4] The friendship graph F_n, introduced in [7], is a planar, undirected graph with $2n+1$ nodes and $3n$ edges. F_n can be constructed by joining n triangles with a common node.

▶ **Theorem 1** (IT-THB for admissible graphs with fixed size, informal). *Let $n \in \mathbb{N}$ with $n \geq 4$, and let $\mathcal{G} \subseteq \mathcal{G}_{\mathsf{wheel}}(n)$ be a graph-class in which every graph has $n+1$ nodes and the center has degree n.*

Then, if either $\mathcal{G} = \mathcal{G}_{\mathsf{star}}(n)$ or if $\mathcal{G}$ consists of admissible graphs, there exists perfectly secure IT-THB against a single semi-honest corruption over $\mathcal{G}$. Otherwise, THB over $\mathcal{G}$ secure against a single semi-honest corruption exists if and only if key agreement exists.

Theorem 1 demonstrates another interesting phenomena: a nontrivial example of a graph-class in which $\mathcal{G}$ is the union of two sub-classes $\mathcal{G}_1$ and $\mathcal{G}_2$, such that each sub-class admits an IT-THB, yet the there is no IT-THB for $\mathcal{G}$. Specifically, while $\mathcal{G}_{\mathsf{wheel}}(n)$ and $\mathcal{G}_{\mathsf{star}}(n)$ each individually admits 1-IT-THB, any 1-THB protocol on $\mathcal{G}_{\mathsf{wheel}}(n) \cup \mathcal{G}_{\mathsf{star}}(n)$ requires key agreement.

Generalizing to variable-size subgraphs. We proceed to analyze subgraphs of $\mathcal{G}_{\mathsf{wheel}}(n)$ that are generated by removing some of the nodes. Note that when removing the center node, the resulting subgraph is either a cycle with n nodes $\mathcal{G}_{\mathsf{cycle}}(n)$, which supports 1-secure perfect THB, or a path with up to n nodes that necessitates key agreement. Therefore, we focus on keeping the center and removing nodes from the perimeter. An interesting observation is that when removing k neighboring nodes from the perimeter, the result is an admissible subgraph of the wheel with $n+1-k$ nodes with one edge removed from the perimeter. Similarly, removing arbitrary k nodes yields a subgraph of the wheel with $n+1-k$ nodes with m edges removed from the perimeter, where m is the number of sets of neighboring nodes that are removed.

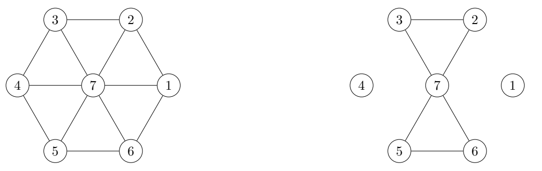


Figure 4 On the left is a wheel graph. On the right is the resulting graph when removing nodes ① and ④ together with their corresponding edges. The result is an admissible graph F_2.

A more interesting question is thus to characterize families of such subgraphs whose number of nodes is not a priori known. We remark that topology hiding on graphs of unknown size can be surprisingly complex: For example, THB with an additional sender-anonymity guarantee for the simple class of 2-paths and 3-paths implies *infinitely often oblivious transfer* [4, Thm 5.4].

We utilize a useful property of the protocol used for proving Theorem 1 (discussed further in Section 2) that effectively hides the number of nodes from non-center parties. We show that the protocol can be applied also to the current setting to obtain perfect IT-THB.

▶ **Theorem 2** (IT-THB for admissible graphs with varying size, informal). *Let $n \in \mathbb{N}$ and let $\mathcal{G}$ be a graph-class such that every $(V,E) \in \mathcal{G}$ is a subgraph of the wheel graph, and it holds that $4 \leq |V| \leq n+1$ and there is a center node with degree $|V|-1$. Then,*

- *if the maximal degree of non-center nodes is 1, i.e., $\mathcal{G}$ consists only of stars (possibly of different size), or*
- *if the minimal degree of non-center nodes is 2 or 3, i.e., $\mathcal{G}$ consists only of admissible graphs, or*
- *if $\mathcal{G}$ consists both of stars and admissible graphs but they are of different sizes,*

there exists perfectly secure IT-THB *against a single semi-honest corruptions over* $\mathcal{G}$. *Otherwise,* THB *over* $\mathcal{G}$ *secure against a single semi-honest corruption exists if and only if key agreement exists.*

We note that Theorem 2 subsumes Theorem 1; therefore, in the technical sections we directly prove Theorem 2.

Tolerating many corruptions: the case of friendship graphs. The feasibility results thus far were limited to a single corruption. The reason lies in the structure of the protocol, which enables two colluding parties with two common neighbors to learn which of them is the center; see Section 2 for an illustration. Therefore, it still remains open whether IT-THB tolerating $t > 1$ corruption is possible, aside from degenerate cases in which the topology is fully determined from neighbor-sets of any t nodes.

We proceed to analyze an interesting class of subgraphs of a wheel graph with varying size, which consists of *friendship graphs*. Recall that for $n \geq 1$, the friendship graph F_n is a $(2n+1)$-nodes graph constructed by joining n triangles with a common node. They were named after the friendship theorem [7], which states that if in a finite set of people every pair has one common friend, then there exists one person who is friend with everyone. We consider a class consisting of friendship graphs of different sizes. Note that the connectivity of each of those graphs is 1, and by their structure every two nodes can only have one common neighbor, so the attack discussed above no longer applies. We prove that indeed perfect IT-THB tolerating *any* number of corruptions can be achieved on this class. For an integer k, consider the graph class $\mathcal{G}_{\mathsf{friendship}}(k)$ containing all isomorphisms of the friendship graph F_k.

▶ **Theorem 3** (t-IT-THB over friendship graphs, informal). *Let* $n \in \mathbb{N}$ *with* $n \geq 2$, *let* $t < 2n+1$, *and consider a graph-class* $\mathcal{G} \subseteq \bigcup_{k=2}^{n} \mathcal{G}_{\mathsf{friendship}}(k)$. *There exists a perfectly secure* THB *protocol against* t *semi-honest corruptions over* $\mathcal{G}$.

We remark that Theorem 3 presents the first feasibility of information-theoretic THB on non-degenerate graph-classes with $t > 1$ corruptions.

Organization of the paper. Due to severe space restrictions, we defer most of the technical content, including the construction of our protocols, the formal statements, and the security proofs, to the full version of the paper. We proceed to provide an overview of the techniques in Section 2.

2 Technical Overview

We move on to describing some of our techniques. We begin by explaining in Section 2.1 the high-level ideas of the protocols used for our positive result. Next, in Section 2.2, we describe our usage of the *phantom-jump* technique from [4] for our negative result.

2.1 Feasibility Results: The "Oblivious Centralized Coordination" Technique

Our protocols are inspired by the THB protocol for the butterfly graph from [4]. We extend it in several aspects to support more involved graph classes that contain an embedded star, i.e., a well-defined center connected to all other nodes. In the overview below, we begin by describing the simpler case of friendship graphs, and then proceed to the wheel graph, and to arbitrary admissible graphs.

Starting point: the butterfly graph. Recall that the butterfly graph (Figure 1) is in fact the friendship graph F_2: a 5-node graph consisting of two triangles connected by a common center node. The high-level idea is to use the center node for coordinating the protocol. The protocol runs multiple instances of *reliable message transmission* (RMT), one for every potential receiver. In each RMT instance, the sender P_S sends its message to all its neighbors in the first step. Note that each party knows whether it is a neighbor of P_S, so it knows whether it should receive a message or not in the first round. At that point it is guaranteed that the center node holds the message and so can deliver it to the receiver (in case the receiver is not the center).

This, of course, will reveal to the receiver who is the center node. Therefore, the center must do so in an *oblivious* way, without exposing itself. In the butterfly graph, if the receiver P_R is not the center it has one more neighbor other than the center. The approach taken in [4] is to secret share the message m with the additional neighbor, and have each neighbor deliver one share. However, the center does not know who that neighbor is. Therefore, the center node prepares 2-out-of-2 shares of the message m for each potential neighbor, i.e., each non-receiver party.

To help the center hide its identity, each other party assists by acting as the center and preparing 2-out-of-2 shares of zero (so called, *blinding terms for addition*) for each of its non-receiver neighbors (a non-center party has either one or two non-receiver neighbors). Next, the receiver receives four values from each of its neighbors (recall that in the butterfly graph there are four nodes other than the receiver, see Figure 1), such that the center sends the sum of the share m for each party with the share of zero it received from that party, and the second neighbor sends the sum of the share received from its non-receiver neighbor with the share of zero sent to this neighbor, along with three random values (one for each other party). The receiver can then select the correct pair which corresponds to its true neighbors. Thus, P_R can reconstruct m without knowing which of its neighbors is the center.

This approach is secure as long as the receiver is not the center. However, if P_R is the center, it may learn the neighbor-set of other nodes (e.g., by inspecting which pairs of values sum up to 0). This is solved by adding *suitable offset* values, which are multiplied by *blinding terms for multiplication*, and only come into play if P_R is the center. Specifically, if P_R is *not* the center, then P_R will send the same offset to both its neighbors (this will ensure that the offset will be canceled out). If P_R is the center, then P_R will send a *different* offset to each neighbor (this requires working over a larger field, e.g., $\mathbb{F}_4$, to support a different value per party); in this case, the pairs of values seen by P_R will induce a linear system of two equations with two variables, and the different offsets will guarantee that the system has full rank and always has a solution. This, in turn, will prove that the center cannot identify which pairs of parties are connected.

The friendship graph. As discussed above, we view the butterfly graph as two triangles connected in a joint node; that is, as the friendship graph F_2. In the full version, we prove that the 1-THB protocol for the butterfly graph-class $\mathcal{G}_{\mathsf{butterfly}}$ (consisting of all isomorphisms of 5 nodes to F_2) extends in natural way to 1-THB for the class of friendship graph $\mathcal{G}_{\mathsf{friendship}}(n)$, for $n \geq 2$, consisting of all isomorphisms of $2n+1$ nodes to F_n.[5] Namely, the receiver now receives a vector of $2n$ values from each of its neighbors, and those values are uniformly distributed conditioned on the corresponding values of its neighbors that sum up to the message. Further, recall that in case the receiver is the center, it must provide a different offset to each of its neighbors; hence, the underlying field $\mathbb{F}_q$ must grow and satisfy $q \geq 2n$.

[5] Note that for $n = 1$ a friendship graph is just a triangle, and there is no well-defined center.

Friendship of variable size. A second observation is that for non-center parties, the protocol behaves in a "local" manner, in the sense that the neighbors of a non-center node are neighbors on their own. When P_R is not the center, this enables the receiver's neighbors to jointly construct the shares of the message in a coordinated (yet oblivious) way. Only the center's actions truly depend on the actual number of parties, while non-center parties only need to know an upper bound on the number of parties.

In the full version, we prove that this locality property makes the protocol suitable for a variable number of nodes (i.e., a variable number of triangles). Non-center nodes proceed as if the graph has n triangles (where n is an upper bound), and the center node emulates missing nodes in its head. Formally, we consider the graph $F_{k,n}$ as an *augmented friendship graph* of $2n+1$ nodes, where $2k+1$ nodes form a connected component which is the friendship graph F_k, and all other $2n+1-(2k+1)=2(n-k)$ nodes are singletons (isolated parties). Each isolated party simply outputs 0 in this protocol (unless it is the sender, in which case it outputs its input), and the *agreement* and *validity* properties are only required for the connected component of the sender.

Friendship with many corruptions. Another interesting observation, is that locality enables tolerating an arbitrary number of $t < 2n+1$ corruptions, *without* any adjustments to the protocol. We prove this in the full version. Intuitively, to see why, we distinguish between an honest center and a corrupt center.

In case the center is honest, then once there is more than a single corruption, the adversary can immediately identify who the center is. This is not considered a violation of privacy, since this can be deduced just by observing the common neighbor of the corrupted parties, and *without* observing any protocol messages. When focusing on each triangle now, if both non-center nodes are corrupted there is nothing to hide within the triangle, whereas if none of the non-center nodes is corrupted the adversary learns nothing new from the protocol. The case where there is a single corrupted non-center in the triangle reduces to the single corruption case from before.

In case the center is corrupted, and there is another non-center corrupted party, then all the information in its triangle is already known, regardless of whether the second non-center party is honest or not. Further, consider the set of honest parties that have an honest neighbor, then the center together with all other corrupt parties do not learn the connectivity of this set.

We note that despite the technical simplicity of this result, it bares a more significant conceptual contribution, as it provides the first feasibility of IT-THB with more than one corruption beyond trivial graph classes.

Beyond friendship: the wheel graph. We proceed to extend the *oblivious centralized coordination* technique to more involved graph classes that admit an embedded star. As before, we begin by considering a *single* corruption. One can view the $(2n+1)$-nodes friendship graph F_n as a subgraph of the wheel W_{2n} in which every non-center node has degree 2. The wheel graph presents the other extreme in some sense, as every non-center node has degree 3.

A first attempt to extend the protocol to this new regime, is to use 3-out-of-3 secret sharing instead of 2-out-of-2. Stated differently, before, in F_n, if P_R is not the center it receives a *vector* of $2n$ values from each of its two neighbors such that the matching pair of values sum up to the message and all other values are independently and uniformly distributed. When considering the $(n+1)$-nodes wheel graph W_n, if P_R is not the center then it has 3 neighbors, and it receives a *matrix* of $n \times n$ values from each of its neighbors such that the corresponding entries in these matrices[6] sum up to the message and all other values are independently and uniformly distributed.

However, as opposed to the friendship regime, once a non-center node has degree 3 the protocol loses its locality property, as now not all neighbors of the receiver are neighbors on their own, and so the matrices are not "synchronized" like the vectors in the previous case. Indeed, if done without care, this approach leads to an attack. The reason is that the preparation of entry (v, w) for the matrix of party P_u is done as follows: if v and w are not neighbors of u sample a random value; if only one is a neighbor use the value that this party sent before (to ensure it will cancel out); and if both parties are neighbors of u then take the sum of their values. Therefore, the receiver can identify repeating entries in a matrix to deduce pairs of neighboring parties, as illustrated in Figure 5.

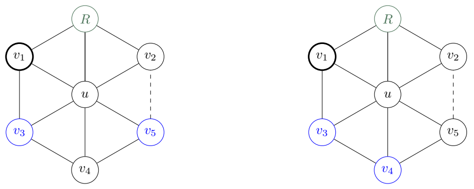


Figure 5 Illustration of an attack on a naïve protocol for $\mathcal{G}_{\mathsf{wheel}}(n)$. The receiver P_R has three neighbors: v_1, v_2, and the center u. Say that v_1 has another neighbor v_3, which has a third neighbor v_4, which has a third neighbor v_5. Then, P_R receives a matrix from v_1; however, since v_3 sends a single value to v_1, the entry (v_3, v_4) will be the same as the entry (v_3, v_5). This means that both v_4 and v_5 are *not* neighbors of v_1.

Our solution to this issue is to have each pair of neighbors (none of which is P_R) generate a vector of n *correlated values*, as opposed to a single value. This is done by having each party sample a vector of random values and send it to each of its non-receiver neighbors. In fact, those correlated values make the *blinding terms* and the *suitable-offset terms* redundant, so these values are no longer used in this protocol. In the full version, we prove that the resulting protocol is secure for the class of wheel graphs.

Admissible graphs. Having established 1-THB for the case when non-center nodes have degree 2 (friendship graphs) and the case where they have degree 3 (wheel graphs), we proceed to combine the ideas together and support any admissible graph. Intuitively, since the protocols share a similar structure, one can hope to execute both options concurrently. That is, the parties run two independent executions: one for the case where P_R has two neighbors, and one for the case where P_R has three neighbors. This, however, is vulnerable to an attack, since when a receiver has three neighbors it can find correlations in the messages it receives for the degree-2 execution and identify who the center is, as illustrated in Figure 6.

[6] That is, for neighbors u, v, w take entry (u, v) from the matrix of w, entry (v, w) from the matrix of u, and entry (w, u) from the matrix of v. In the protocol, we ensure the matrices are symmetric, i.e., $\mathbf{M}[u, v] = \mathbf{M}[v, u]$.

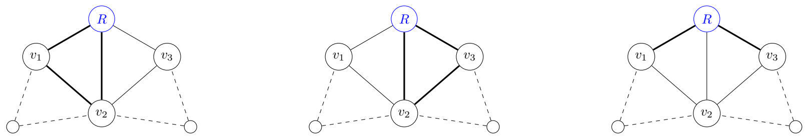


Figure 6 Illustration of an attack on a non-careful protocol for admissible graphs. Consider a non-center receiver P_R with neighbors v_1, v_2, and v_3; assume that v_2 is the center. Further consider running the friendship protocol over this graph. The left diagram, illustrates the view P_R obtains for the triangle with v_1 and v_2: here P_R will obtain the message m. The middle diagram, illustrates the view P_R obtains for the triangle with v_2 and v_3: again, P_R will obtain the message m. The right diagram, illustrates the view P_R obtains for the triangle with v_1 and v_3: here, there is no direct edge between v_1 and v_3; hence, P_R will *not* obtain the message m. Therefore, P_R can identify that v_2 is the center.

The main idea in overcoming this attack, is that although we need to run two executions in order to hide the degree of the receiver (when it is not the center), we only need one execution to deliver the message to the receiver, and the second does not need to convey any information. Further, the receiver already knows its degree, so it knows which execution is the “right” one, and can sabotage the “redundant” one. Specifically:

- In case the receiver’s degree is 3, in the degree-2 execution it will send a *different* offset for each neighbor (and the degree-3 execution will be executed correctly).
- In case the receiver’s degree is 2, in the degree-2 execution it will correctly send the *same* offset to its neighbors (and the degree-3 execution will not leak any information because the receiver does not have three neighbors).

In the full version, we prove that the resulting protocol is secure against one corruption for graph-classes consisting of admissible graphs.

Many corruptions. The protocol described above establishes feasibility of 1-IT-THB for any graph-class consisting of admissible graphs (even of variable size). This feasibility is tight for a single corruption, as stated in Theorem 2. It is tempting though to extend the resiliency of the protocol, similarly to the class of friendship graphs that support any number of corruptions. It turns out that the non-local nature of non-friendship, admissible graphs enables an attack on the protocol when the adversary controls two nodes.

We illustrate the attack in Figure 7. Consider a pair of corrupted parties ② and ④ , and assume that none of them is the center. Further, assume that each has degree 3, and that they have two common neighbors, denoted ③ and ⑥ . Clearly, by the structure of the graph, ② and ④ together can deduce that either ③ is the center, or ⑥ is the center.

However, when running in this setting the 1-secure protocol described above, the colluding parties may learn correlations that will expose which of their common neighbors is the center. Specifically, recall that when ③ is the receiver, it sends to its neighbors ② and ④ the suitable-offset values. In case ③ is the center, the offset value for ② is the same as the one for ④ , whereas in case ③ is *not* the center these are different values.

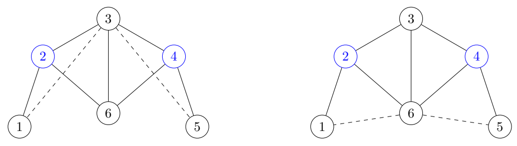


Figure 7 Attack on non-friendship admissible graphs with two corruptions.

We emphasize that in friendship graphs every non-center node has degree two; hence, the scenario from Figure 7 cannot occur. We leave it as an open question to find a protocol that is resilient to $t > 1$ corruptions for non-friendship admissible graphs.

2.2 Impossibility Results: The "Phantom Jump" Technique

The phantom-jump technique, introduced in [4], was used to show that key agreement is necessary for 1-secure THB over the class $\mathcal{G}_{\mathsf{triangle}}$ consisting of a triangle, with possibly one of its edges missing (see Figure 8). In this class, if a party has two neighbors it does not know whether its neighbors are directly connected or not, but a party with one neighbor knows the entire topology.

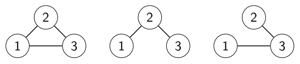


Figure 8 The class $\mathcal{G}_{\mathsf{triangle}}$ from [4], consisting of a triangle, with possibly one of its edges missing.

In the full version we prove the lower bound of Theorem 1 (namely that 1-THB on the union of an admissible graph-class of size $n + 1$ with $\mathcal{G}_{\mathsf{star}}(n)$ necessitates key agreement) by a direct reduction to the impossibility in [4]. Below we explain in a more explicit manner how the phantom-jump technique from [4] is used in this argument. We illustrate this for $\mathcal{G} = \mathcal{G}_{\mathsf{wheel}}(4) \cup \mathcal{G}_{\mathsf{star}}(4)$ where both graphs consist of 5 nodes.

The high-level idea, going back to [5], is to construct a key-agreement protocol from a 1-secure THB protocol π for $\mathcal{G}$. Recall the desired key-agreement protocol is run between two parties, Alice and Bob, and concludes with the parties outputting a bit $b \in \{0, 1\}$, such that a channel eavesdropper listening to communications cannot predict the value of b with non-negligible advantage. To construct a key-agreement protocol from π, Alice begins by choosing two long random strings m_1 and m_2 and sending them to Bob in the clear. Next, Alice and Bob continue in phases as follows:

- In each phase Alice and Bob locally toss coins A and B, respectively.
- They proceed to run two executions of π in which Alice always emulates ① and Bob always emulates ②. In addition, if $A = 0$ then Alice emulates ③ , ④ , and ⑤ as neighbors of ①, who acts as the center of the star, and ③ broadcasting m_1 in the first run; otherwise she emulates ③ , ④ , and ⑤ as neighbors of ①, who acts as the center of the star, and ③ broadcasting m_2 in the second run. Similarly, if $B = 1$ then Bob emulates ③ , ④ , and ⑤ as neighbors of ②, who acts as the center of the star, and ③ broadcasting m_1 in the first run; otherwise he emulates ③ , ④ , and ⑤ as neighbors of ②, who acts as the center of the star, and ③ broadcasting m_2 in the second run. See Figure 9 for an illustration.
- If parties ① and ② output m_1 in the first run and m_2 in the second, Alice and Bob output their bits A and B, respectively; otherwise, they execute another phase.

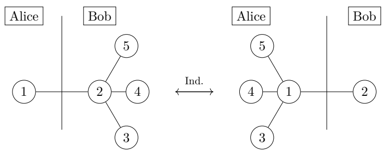


Figure 9 Using wheels and stars to construct a key-agreement protocol.

Clearly, if $A = B$ in some iteration then Alice and Bob will output the same coin, and by the assumed security of π, the eavesdropper Eve will not be able to learn who emulated ③ , ④ , and ⑤ in the first run and who in the second. If $A \neq B$, then in at least one of the runs nobody emulates the broadcaster ③ , so with overwhelming probability Alice and Bob will detect this case and execute another iteration.

In more detail, when $A = B$ the view of Eve consists of the communication between ① and ② , as depicted in Figure 9. By THB security, when ② acts as the center it cannot distinguish between the star and the wheel; in particular, the distribution of the messages on the channel between ① and ② is indistinguishable in both cases. Again, by THB security, when ① is not the center of the wheel it cannot know which of its neighbors is the center, so it cannot distinguish between the center being ② or ③ ; in particular, the distribution of the messages on the channel between ① and ② is indistinguishable in both cases. Similarly, when ② is not the center of the wheel, it cannot distinguish between the center being ① or ③ ; in particular, the distribution of the messages on the channel between ① and ② is indistinguishable in both cases. Finally, when ① acts as the center it cannot distinguish between the star and the wheel; in particular, the distribution of the messages on the channel between ① and ② is indistinguishable in both cases. By a simple hybrid argument it follows that the messages between ① and ② are indistinguishable when communicating in a star topology when ① is the center and when ② is the center, and it follows that the distinguishing advantage of Eve is negligible. See Figure 10 for an illustration of the hybrid argument.

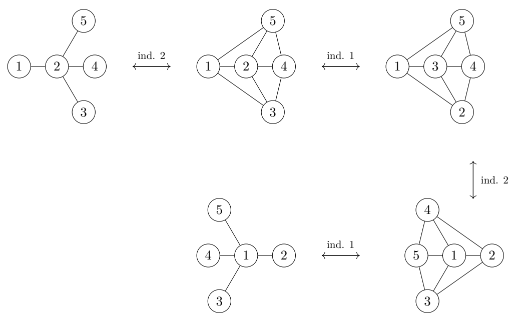


Figure 10 Hybrid steps in the phantom jump over wheels and stars.

References

1 Adi Akavia, Rio LaVigne, and Tal Moran. Topology-hiding computation on all graphs. In *37th Annual International Cryptology Conference (CRYPTO), part I*, pages 447–467, 2017.

2 Adi Akavia and Tal Moran. Topology-hiding computation beyond logarithmic diameter. In *36th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), part III*, pages 609–637, 2017.

3 Marshall Ball, Alexander Bienstock, Lisa Kohl, and Pierre Meyer. Towards topology-hiding computation from oblivious transfer. In *Proceedings of the 21st Theory of Cryptography Conference (TCC), part I*, pages 349–379, 2023.

4 Marshall Ball, Elette Boyle, Ran Cohen, Lisa Kohl, Tal Malkin, Pierre Meyer, and Tal Moran. Topology-hiding communication from minimal assumptions. In *Proceedings of the 18th Theory of Cryptography Conference (TCC), part II*, pages 473–501, 2020.

5 Marshall Ball, Elette Boyle, Ran Cohen, Tal Malkin, and Tal Moran. Is information-theoretic topology-hiding computation possible? In *Proceedings of the 17th Theory of Cryptography Conference (TCC), part I*, pages 502–530, 2019.

6 Marshall Ball, Elette Boyle, Tal Malkin, and Tal Moran. Exploring the boundaries of topology-hiding computation. In *37th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), part III*, pages 294–325, 2018.

7 Paul Erdös, Alfréd Rényi, and Vera T. Sós. On a problem of graph theory. *Studia Sci. Math. Hungar.*, 1:215–235, 1966.

8 Martin Hirt, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. Network-hiding communication and applications to multi-party protocols. In *36th Annual International Cryptology Conference (CRYPTO), part II*, pages 335–365, 2016.

9 Rio LaVigne, Chen-Da Liu Zhang, Ueli Maurer, Tal Moran, Marta Mularczyk, and Daniel Tschudi. Topology-hiding computation beyond semi-honest adversaries. In *Proceedings of the 16th Theory of Cryptography Conference (TCC), part II*, pages 3–35, 2018.

10 Rio LaVigne, Chen-Da Liu Zhang, Ueli Maurer, Tal Moran, Marta Mularczyk, and Daniel Tschudi. Topology-hiding computation for networks with unknown delays. In *Proceedings of the 23rd International Conference on the Theory and Practice of Public-Key Cryptography (PKC), part II*, pages 215–245, 2020.

11 Shuaishuai Li. Towards practical topology-hiding computation. In Shweta Agrawal and Dongdai Lin, editors, *28th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT), part I*, pages 588–617, 2022.

12 Tal Moran, Ilan Orlov, and Silas Richelson. Topology-hiding computation. In *Proceedings of the 12th Theory of Cryptography Conference (TCC), part I*, pages 159–181, 2015.

Communication Complexity vs Randomness Complexity in Interactive Proofs

Benny Applebaum
Tel-Aviv University, Israel

Kaartik Bhushan
IIT Bombay, India

Manoj Prabhakaran
IIT Bombay, India

Abstract

In this work, we study the interplay between the communication from a verifier in a general private-coin interactive protocol and the number of random bits it uses in the protocol. Under worst-case derandomization assumptions, we show that it is possible to transform any I-round interactive protocol that uses ρ random bits into another one for the same problem with the additional property that the verifier's communication is bounded by $O(I \cdot \rho)$. Importantly, this is done with a minor, logarithmic, increase in the communication from the prover to the verifier and while preserving the randomness complexity. Along the way, we introduce a new compression game between computationally-bounded compressor and computationally-unbounded decompressor and a new notion of conditioned efficient distributions that may be of independent interest. Our solutions are based on a combination of perfect hashing and pseudorandom generators.

2012 ACM Subject Classification Theory of computation → Interactive proof systems

Keywords and phrases Interactive Proof Systems, Communication Complexity, Hash Functions, Pseudo-Random Generators, Compression

Digital Object Identifier 10.4230/LIPIcs.ITC.2024.2

Funding The first and second authors are supported by ISF grant no. 2805/21 and by the European Union (ERC, NFITSC, 101097959). Views and opinions expressed are however those of the authors 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. The second and third authors are also supported by IIT Bombay Trust Lab. The third author is also supported by an Algorand Centre of Excellence grant by the Algorand Foundation.
Benny Applebaum: Supported by ISF grant no. 2805/21 and by the European Union (ERC-2022-ADG) under grant agreement no.101097959 NFITSC.
Kaartik Bhushan: Supported in part by IITB Trust Lab. Part of the research was done while visiting Tel Aviv Univerisity and was supported by ISF grant no. 2805/21 and by the European Union (ERC-2022-ADG) under grant agreement no.101097959 NFITSC. Also supported by the Prime Minister's Research Fellowship (PMRF), Government of India.
Manoj Prabhakaran: Supported in part by IITB Trust Lab and by an Algorand Centre of Excellence at IIT Bombay funded by the Algorand Foundation.

Acknowledgements We thank Gil Segev for valuable discussions about perfect hashing. Part of the research was done while the second author visited Tel Aviv University.

1 Introduction

Interactive probabilistic proof systems [15, 4] are central objects in cryptography and complexity theory. Roughly speaking, they allow a computationally unbounded prover to convince a computationally-bounded randomized verifier that a certain statement holds. The use of *interaction* and *randomness* extends the classical notion of NP problems (that can be deterministically verified given a single message from a prover) all the way up to polynomial-space computable languages [23, 33]. If the verifier does not use randomness, clearly interaction is useless: the prover can compute all the verifier's queries on her own and just send the answers. That is, when the verifier uses 0 bits of randomness, it needs to communicate 0 bits. In this paper, we try to extend this observation: if the verifier uses only ρ random bits, how many bits does it need to communicate?

Intuitively, since the entropy of the verifier's messages is at most ρ, there is no point in communicating more than ρ bits. Indeed, if the proof system is a public-coin system (in which the verifier simply sends random coins in each round) this intuition holds, and the randomness complexity equals the communication complexity. However, in the general case of private-coin proof system, the verifier's messages may be much longer than their entropy. We note that although it is possible to transform a given private-coin proof system into a public-coin system [16, 13, 22], existing transformations *increase* the overall communication. In particular, a transcript of the modified public-coin proof system contains at least one copy of a transcript of the original private-coin proof system plus some additional overhead which is polynomial in the original communication (to certify that the transcript is "typical" or "heavy"). Furthermore, these transformations also increase the randomness complexity of the verifier either polynomially [16, 13] or by a constant factor [22]. Overall, the following question remains open:

> Is it possible to transform a proof system with randomness ρ in which the prover sends out Com_P bits, into a new proof system for the same problem in which the total communication depends on the verifier's randomness complexity in the original proof system rather than the verifier's outgoing communication, i.e., the total communication is $\mathsf{poly}(\rho, \mathsf{Com}_\mathsf{P})$ or even $O(\rho + \mathsf{Com}_\mathsf{P})$? Further, can we do this while preserving the randomness complexity?

Closely related questions were studied in [2] and [22]. Specifically, [2] studied the "converse question" of upper-bounding the randomness complexity in terms of the communication complexity (aka "randomness sparsification"), and [22] studied the question of bounding the *round complexity* in terms of the randomness complexity. The latter work shows [22, Thm A.1] that a proof system that uses $\rho(n)$ random bits for n-long instances can be converted into a proof system with $O(\rho(n)/\log n)$ rounds, while preserving the randomness complexity.[1] Unfortunately, this transformation increases the total communication complexity (as the prover guesses in every round many possible extensions to the current transcript). In particular, even for constant-round protocol this transformation may increase the communication by a $\mathsf{poly}(n)$ factor.

1.1 Our Results

We partially resolve the above question by relying on a complexity theoretic assumption. In particular, we prove the following main result. (See Section 2 for a formal presentation of the hardness assumption).

▶ **Theorem 1.** *Suppose that a promise problem Π has an interactive proof system $\langle \mathsf{P}, \mathsf{V}\rangle$ with round complexity $I(n)$, randomness complexity $\rho(n)$, verifier communication $\mathsf{Com}_\mathsf{V}(n)$, prover communication $\mathsf{Com}_\mathsf{P}(n)$, where n denotes the length of the instance. Then, assuming that*

[1] In fact, if one is willing to increase the randomness complexity by a constant factor, then it is possible to derive an $O(\rho(n)/\log n)$-round *public-coin* system [22, Thm 1.1].

$\mathsf{E} = \mathsf{DTime}(2^{O(n)})$ *is hard for exponential-size non-deterministic NP-circuits, there exists an interactive proof system* $\langle \mathsf{P}', \mathsf{V}' \rangle$ *for* Π *in which the verifier and prover communication are*

$$\mathsf{Com}_{\mathsf{V}'}(n) = O(I(n)\rho(n)), \text{ and}$$
$$\mathsf{Com}_{\mathsf{P}'}(n) = \mathsf{Com}_{\mathsf{P}}(n) + O(I(n)\log n).$$

The randomness complexity of the proof system remains unchanged, the round complexity grows by (at most) 1, and the completeness error grows additively by 0.1.

For the special case of constant-round protocols, the verifier communicates $O(\rho(n))$ bits and the prover's additive overhead is at most $O(\log n)$; if the original prover communicates in each round a logarithmic number of bits (resp., super-logarithmic number of bits), the additive overhead is linear (resp., sub-linear) in $\mathsf{Com}_{\mathsf{P}}(n)$. In the general case, when the prover may communicate as little as one bit per round, we still have $I(n) \leq \mathsf{Com}_{\mathsf{P}}(n)$; then the overhead for the prover communication is a multiplicative factor of $O(\log n)$, and the total communication is $O(\mathsf{Com}_{\mathsf{P}}(n) \cdot (\rho(n) + \log n))$. Note that in all cases, the total communication is independent of $\mathsf{Com}_{\mathsf{V}}(n)$, the verifier's original communication. The question of achieving a total communication of $O(\rho(n) + \mathsf{Com}_{\mathsf{P}}(n))$ for protocols with polynomially-many rounds remains an interesting open question.

Theorem 1 is based on a worst-case assumption. This assumption asserts that one cannot significantly speed-up (uniform) Exponential-Time problems by adding non-uniformity and two levels of non-determinism (non-deterministic NP-circuits are the non-uniform analogue of $\mathsf{NP}^{\mathsf{NP}}$; see Section 2 for details). This assumption is somewhat strong but widely believed as it reflects our current understanding of the relations between time, nonuniformity and nondeterminism. Similar assumptions have been extensively used in cryptography and complexity theory (see, e.g.,[9, 20, 24, 37, 30, 14, 17, 31, 6, 32, 8, 1, 2, 5, 29]).

1.2 Technical Overview

Consider the simple case of a single-round protocol where the verifier's message is of length m bits that is much larger than the randomness complexity ρ. In this case the verifier is sending a long message that is sampled from a low-entropy distribution $\mathcal{D}$, and our goal is to reduce the communication. The crucial observation is that the prover has full knowledge of the distribution $\mathcal{D}$, and, being computationally unbounded, he can help the verifier to *compress* the message. Indeed, we can abstract this scenario as a special variant of the well-known *data compression* problem.

1.2.1 Single-Round Compression Game

In this data compression game there are two parties: a computationally bounded compressor CMP and a computationally-unbounded decompressor DCMP. At the beginning of the game, the compressor is given a string $x \in \{0,1\}^m$ that is efficiently sampled from a probability distribution $\mathcal{D}$ whose full description is given to the decompressor DCMP. The goal of the compressor is to deliver the string x to the decompressor with probability at least $1 - \epsilon$ taken over the choice of x. The parties are allowed to communicate in both directions, and the goal is to minimize the communication ideally up to the entropy of the distribution.

It is instructive to compare our game to a few other compression games. Shannon's original game [34] refers to compression of *multiple independent* samples from $\mathcal{D}$, and his celebrated source-coding theorem shows that the expected amortized communication approaches the entropy. In contrast, our game involves a single-shot challenge and worst-case communication and, accordingly, allows some error (i.e., the scheme may be *lossy*). In computer science literature, Ta-Shama et al. [38] studied the problem of compressing "computationally-weak" sources by a computationally efficient compressor and decompressor and provided compression schemes whose communication complexity is close to the entropy for several classes of such distributions. In contrast, in our setting the decompressor is allowed to be computationally-unbounded, and as a result we can bypass some of the lower-bounds of [38] (e.g., we can hope to compress pseudorandom distributions). Finally, Orlitsky [26] considered a one-shot compression game in which the parties are computationally-unbounded but have some information gap captured by some auxiliary information y about x that is given to the decompressor and is unknown to the compressor. Notably, Orlitsky's schemes use interaction (like in our setting) whereas the schemes of [34, 38] are non-interactive (the compressor sends a single message to the decompressor).

Getting back to our compression game, let us further simplify the problem and assume that we care only about the communication from the compressor to the decompressor. In this case, there is a simple solution that is described in Lemma 14. Take $k = H(\mathcal{D})/\epsilon$ where $H(\cdot)$ denotes Shannon's entropy, and let DCMP send a description of hash function $f : \{0,1\}^m \to \{0,1\}^k$ that is 1-1 over the set of 2^k heaviest strings in $\mathcal{D}$. The compressor CMP responds with the "digest" $y = f(x)$, and DCMP outputs the "heaviest" string x' in $\mathcal{D}$ that is consistent with y. It is not hard to show that the error is at most ϵ (see Lemma 12) which is essentially the best that one can hope for.[2] Indeed, this approach can be viewed as one-shot analog of Shannon's celebrated Source coding theorem [34]. Unfortunately, the decompressor has to communicate the description of a hash function f which is taken from a family F of 2^k*-perfect hash function.* That is, the family F contains, for each 2^k-subset of strings $X \subset \{0,1\}^m$, a function f that is injective on X and so it cannot be too small. In fact, it is known that the description size must be at least $\Omega(2^k)$ bits [25]. The cost can be significantly reduced by allowing some slackness, i.e., by expanding the output length of f to $k' > k$ (e.g., $k' = 3k$). In this case, the description length can be reduced to $\Omega(m + k')$ bits by using existing families of perfect hash functions (e.g., [10]). However, this is still too expensive for our purposes.[3]

1.2.2 Focusing on efficiently samplable distribution

We note that when the distribution $\mathcal{D}$ is taken from a family of efficiently samplable distributions (i.e., there is an efficient algorithm that given a random tape outputs a sample from $\mathcal{D}$) it is possible to compress the description length of the hash function. Indeed, in this case our family should be injective only over "nice" sets that correspond to heavy strings in distributions that can be described by a polynomial-size circuit. Specifically, we begin with a standard, off-the-shelf, family $F = \{f_z\}_{z \in \{0,1\}^{m+k'}}$ (e.g., based on pair-wise independent hash functions [7]) in which each function is identified by a long string $z \in \{0,1\}^{m+k'}$. Next, we reduce the description length of the functions via the use of an appropriate pseudorandom

[2] Indeed, for every $\epsilon > 0$, there exists a distribution $\mathcal{D}$ such that any event of probability $1-\epsilon$ must be supported over at least 2^k strings for $k = \Omega(H(\mathcal{D})/\epsilon)$. For example, consider the distribution $\mathcal{D}$ obtained by sampling, with probability 2ϵ, a uniform x from a set A of size $2^{\ell/2\epsilon}$, and, with probability $1-2\epsilon$, a uniform x from a disjoint set B of size 2^{ℓ}. The entropy of $\mathcal{D}$ is $\Theta(\ell)$ and in order to capture $1-\epsilon$ of the mass, one must collect at least ϵ fraction of the strings in A, i.e., 2^k strings for $k \geq \ell/2\epsilon - \log(1/\epsilon) = \Omega(H(\mathcal{D})/\epsilon)$.

[3] To the best of our knowledge, the description length of all existing constructions of 2^k-perfect hash functions is either linear in 2^k or in the input-length m, see e.g., [27].

generator (PRG) $G : \{0,1\}^{\ell} \to \{0,1\}^{m+k'}$. That is, each function f'_s in the new family F' is indexed by a seed s of the PRG and is defined to be $f'_s = f_{G(s)}$. We show that if the PRG fools $\mathsf{AM/poly}$ adversaries the family F' can be used to compress $\mathcal{D}$. By using standard derandomization assumptions (slightly weaker than the one stated in Theorem 1), we get such a PRG with exponential stretch which allows us to reduce the communication from the decompressor to logarithmic. It should be mentioned that the idea of using a PRG to (partially) derandomize a probabilistic construction is not new. This paradigm was abstracted by [20], and was also used in many relevant works. Interestingly, the same paradigm was used in [1] for the contrary purpose of constructing so-called *incompressible functions*.

1.2.3 Back to interactive proofs

Let us move back to the case of multi-round interactive proofs. A natural strategy is to apply the above approach for each round. Roughly, in each round, we let w denote the partial transcript and let $\mathcal{D}_w$ denote the distribution of the next message of the verifier. Instead of letting the verifier send his message x, the parties will run a compression protocol in which the prover selects a hash function f that is injective on the 2^k heaviest strings, where k is about $H(\mathcal{D}_w)/\epsilon$ for some error parameter ϵ. The problem is that $\mathcal{D}_w$ is not efficiently samplable, rather it is obtained by feeding the verifier with the prover messages and random coins that are *conditioned* on generating the partial transcript w. We abstract this property via the notion of *conditioned efficient distributions*. This notion generalizes the notion of efficiently samplable distributions by allowing the sampler A to output a special failure symbol $\perp$, and by letting $\mathcal{D}$ denote the outcome of A applied to random coins conditioned on not outputting $\perp$. By using slightly stronger PRGs, we extend our compression schemes to the case of conditioned efficient distributions, and employ them to reduce the interaction of interactive proofs as stated in Theorem 1.

▶ Remark 2 (More on conditioned efficient distributions). An equivalent way to define a conditioned efficient distribution is by considering a pair of algorithms, Sampler S and Conditioner E, such that sampling from $\mathcal{D}$ boils down to sampling a random tape r conditioned on $E(r) = 1$ and outputting $S(r)$. Thus this new notion can be viewed as a combination of two well-studied classes of distributions: distributions over circuit's outputs (i.e., efficiently samplable distributions) and distributions over circuit's inputs that lead to a given result (aka *efficiently recognizable distributions* [28]). This new notion is natural and may prove to be useful elsewhere.

1.2.4 Organization

Following some preliminaries in Section 2, we construct compression schemes in Section 3 and use them to prove our main theorem in Section 4.

2 Preliminaries

2.1 Probability distributions

For a discrete probability distribution $\mathcal{D}$, let $\Pr_{\mathcal{D}}(x)$ denote the probability of the string x being sampled according to the distribution. Throughout the paper, we will only work with probability distributions that are supported on a finite set. The *Shannon entropy* (or simply entropy) $H(\mathcal{D})$ of a discrete probability distribution $\mathcal{D}$ supported on a finite set D is defined as the quantity

$$H(\mathcal{D}) = \sum_{x \in D} \Pr_{\mathcal{D}}(x) \log\left(\frac{1}{\Pr_{\mathcal{D}}(x)}\right).$$

▶ **Definition 3** (Efficient and conditioned efficient probability distributions). *A family of probability distributions $\{\mathcal{D}_w\}$ is* efficiently samplable *(or simply,* efficient*) if there exist parameters ρ_w, m_w denoted as the* randomness complexity *and the* domain bit-length*, and a PPT* sampling *algorithm A that given an index $w \in \{0,1\}^*$ and a random tape with $r \in \{0,1\}^{\rho_w}$ samples a random string $x \in \{0,1\}^{m_w}$ according to the distribution $\mathcal{D}_w$. The* complexity *of $\{\mathcal{D}_w\}$ is at most T if $A(w;r)$ runs in time $T(|w|)$ for every w and r.*

A family of distributions $\{\mathcal{D}_w\}$ is said to be conditioned efficiently samplable *(or simply,* conditioned efficient*) if, with parameters ρ_w, m_w as above, there exists a PPT algorithm A which, on input $w \in \{0,1\}^*$ and a uniformly random string $r \in \{0,1\}^{\rho_w}$ on its random tape, outputs an element $x \in \{0,1\}^{m_w} \cup \{\perp\}$ such that, conditioned on not being $\perp$ the output is distributed as $\mathcal{D}_w$. We will refer to such an algorithm A as a conditional sampler for $\mathcal{D}_w$.*

2.2 Promise problems

A promise problem Π consists of a pair of disjoint sets of strings $\Pi_{\mathsf{yes}}, \Pi_{\mathsf{no}} \subset \{0,1\}^*$. Strings in Π_{yes} are referred to as *yes* instances and strings in Π_{no} are referred to as *no* instances. The standard definition of a *language* corresponds to the case where every string is either a yes instance or a no instance, i.e., $\Pi_{\mathsf{yes}} \cup \Pi_{\mathsf{no}} = \{0,1\}^*$. (See [11] for a discussion and references.) For two parties A and B engaging in a protocol on common input x, let $\langle \mathsf{A}, \mathsf{B} \rangle(x)$ denote the final output of the protocol.

▶ **Definition 4** (Interactive Proofs). *An interactive proof system for a promise problem $\Pi = (\Pi_{\mathsf{yes}}, \Pi_{\mathsf{no}})$ is defined by a computationally bounded probabilistic verifier V, with a polynomial T_{V} such that the running time of V on common input x is upper-bounded by $T_{\mathsf{V}}(|x|)$, and an unbounded prover P satisfying the following properties:*

- *if $x \in \Pi_{\mathsf{yes}}$, then $\Pr[\langle \mathsf{P}, \mathsf{V} \rangle(x) = 0] \leq \gamma$, and*
- *if $x \in \Pi_{\mathsf{no}}$, then $\forall \mathsf{P}^*, \Pr[\langle \mathsf{P}^*, \mathsf{V} \rangle(x) = 1] \leq \delta$,*

where γ and δ are constants in $[0,1)$ denoting the errors in completeness and soundness respectively. By default, we assume that $\gamma = \delta = 0.1$.

2.3 Arthur-Merlin Proofs, and NP/Non-Deterministic Circuits

An AM protocol is a constant-round public-coin proof system and $\mathsf{AM/poly}$ is the non-uniform analog in which the verifier is implemented by a family of polynomial-sized probabilistic circuits. The complexity class $\mathsf{AM/poly}$ consists of all promise problems that admit $\mathsf{AM/poly}$ protocols. (See standard textbooks like [3, 12] for formal definition.) A *nondeterministic* circuit C has additional "nondeterministic input wires". We say that the circuit C evaluates to 1 on x iff there exist an assignment to the nondeterministic input wires that makes C output 1 on x. An *NP-circuit C* (resp., *nondeterministic NP-circuit*) is a standard circuit (resp., nondeterministic circuit) which in addition to the standard gates uses SAT gates, where a SAT gate gets a formula φ as an input and returns 1 iff the formula is satisifiable. The *size* of the circuit is the total number of wires and gates. Polynomial-size nondeterministic circuits, NP-circuits, non-deterministic NP-circuits are the non-uniform analogues of NP, P^{NP} and $\mathsf{NP}^{\mathsf{NP}} = \Sigma_2^P$, respectively.

The literature on complexity theory and derandomization contains various hardness assumptions against $\mathsf{AM/poly}$/nondeterministic/nondeterministic NP circuits and their generalizations to higher levels of the polynomial hierarchy. (See, e.g., [1] and references therein). Specifically, we will make use of the following result.

▶ **Theorem 5** (PRGs from hardness assumptions [18, 21, 30, 31]). *Suppose that* $\mathsf{E} = \mathsf{DTime}(2^{O(n)})$ *is hard for exponential-size non-deterministic circuits (resp., exponential-size non-deterministic NP-circuits), i.e., there exists a language L in* E *and a constant $\beta > 0$, such that for every sufficiently large n, circuits of size $2^{\beta n}$ fail to compute the characteristic function of L on inputs of length n.*

Then for every polynomial $T(\cdot)$ and inverse polynomial $\epsilon(\cdot)$, for all sufficiently large m, there exists a pseudorandom generator G that stretches seeds of length $\rho = O(\log m)$ into a string of length m in time $\mathsf{poly}(m)$ *such that G ϵ-fools every promise problem $\Pi = (\Pi_{\mathsf{yes}}, \Pi_{\mathsf{no}})$ that can be decided by an* $\mathsf{AM/poly}$ *proof system with a T-size verifier (resp., by a non-deterministic NP-circuit of size T) in the following sense. For every sufficiently large m and $b \in \{\mathsf{yes}, \mathsf{no}\}$*

$$|\Pr_{z \xleftarrow{R} U_m}[z \in \Pi_b] - \Pr_{z \xleftarrow{R} G(U_\rho)}[z \in \Pi_b]| \leq \epsilon(m).$$

As noted in [1], the above assumptions can be seen as the nonuniform and scaled-up versions of assumptions of the form Exponential-Time is not equal to NP or to Σ_2^P (which are widely believed in complexity theory). As such, these assumptions are very strong, and yet plausible - the failure of one of these assumptions will force us to change our current view of the interplay between time, nonuniformity and nondeterminism. As a secondary advantage (also noted in previous works), one can base the PRG on any concrete E-complete problem, and an explicit PRG whose security reduces to the underlying assumption. (We do not have to consider and evaluate various different candidate functions for the hardness assumption.)

2.4 Set-lower bound

We will make use of the set lower-bound protocol of [16].

▶ **Theorem 6** (Set lower-bound protocol[16]). *Let $S \subset \{0,1\}^*$ be an* NP *set (i.e., membership in S can be efficiently verified). Then there exists an* AM *protocol $\langle \mathsf{P}, \mathsf{V} \rangle$ such that given $(1^n, k)$ as common inputs the following holds,*

- *if $|S \cap \{0,1\}^n| \geq k$, then $\Pr[\langle \mathsf{P}, \mathsf{V} \rangle(1^n, k) = 1] \geq 0.9$,*
- *if $|S \cap \{0,1\}^n| \leq k/2$ then for every prover P^* it holds that $\Pr[\langle \mathsf{P}^*, \mathsf{V} \rangle(1^n, k) = 1] < 0.1$.*

2.5 Approximate counting

We say that a number p is an ϵ-relative approximation to q if $(1-\epsilon) \cdot p \leq q \leq (1+\epsilon) \cdot p$. It is useful to note that if p' is an ϵ-approximation to p and q' is an ϵ-approximation to q, then a p'/q' is a 2ϵ-approximation to p/q. We use the following classical result on approximate counting of satisfying assignments.

▶ **Theorem 7** (approximate counting,[19, 36, 35]). *For every sufficiently large s and every $\epsilon > 0$, there is an NP-circuit of size* $\mathsf{poly}(s/\epsilon)$ *that given a (standard) circuit C of size s outputs an ϵ-approximation of $|\{x : C(x) = 1\}|$.*

2.6 Hashing

We say that a function family $\mathcal{F}_{m,d}$ is (δ, s)-injective if for every set $S \subset \{0,1\}^m$ of size at most 2^s, a random member $f \xleftarrow{R} \mathcal{F}_{m,d}$ is injective over S with probability at least $1 - \delta$. It is well known that pair-wise independent hash functions [7] have this property. Formally, the following statement follows from [10] where the pair-wise independent hash family is instantiated with hash functions that are based on, say, Toeplitz matrices.

▶ **Lemma 8** (Good hashing from 2-wise independence). *There exists a family of hash functions $\mathcal{F}$ such that for every δ, s and $d = 2s + \lceil \log\left(\frac{1}{\delta}\right) \rceil$, the restriction $\mathcal{F}_{m,d}$ of $\mathcal{F}$ to functions from m-bits to d-bits is (δ, s)-injective. Moreover, (1) Functions in $\mathcal{F}_{m,d}$ are indexed by strings $z \in \{0,1\}^L$ where $L = m + 2d$; (2) There exists an efficient universal evaluation algorithm F that given $(1^m, 1^d)$, an index $z \in \{0,1\}^L$, and an input $x \in \{0,1\}^m$ outputs $f_z(x)$; and (3) for $z \overset{R}{\leftarrow} \{0,1\}^{m+2d}$, the function $F(1^m, 1^d, z, \cdot)$ is uniform over $\mathcal{F}_{m,d}$.*

3 Hashing-Based Solution for the Compression Problem

In this section we construct compression schemes. We begin with a formal definition.

▶ **Definition 9** (Interactive Compression). *An interactive compression scheme for a family of distributions $\{\mathcal{D}_w\}$ with error $\epsilon(\cdot)$ is defined by a computationally bounded compressor* CMP *and a computationally-unbounded decompressor* DCMP *satisfying the following property: For every $w \in \{0,1\}^*$ given to* DCMP*, and $x \overset{R}{\leftarrow} \mathcal{D}_w$ given to* CMP *the probability that* DCMP *outputs x is at least $1 - \epsilon(w)$, where the probability is taken over the choice of x and the randomness of the parties (if the parties are randomized). The communication complexity of the decompressor and compressor, $\mathsf{Com}_{\mathsf{DCMP}}(w)$ and $\mathsf{Com}_{\mathsf{CMP}}(w)$, are defined to be the maximal number of bits communicated by the decompressor and compressor when the decompressor's input is w.*

It is natural to solve the compression problem by letting the compressor hash the string x to a shorter string. If the hash function is injective over a "heavy set" of strings then the decompressor will be able to recover x from the hash of x, with a low error probability. This idea resembles Shannon's celebrated source coding theorem [34] except that we use a single instance of the source and accordingly rely on a weaker concentration of measure results (Markov's inequality as opposed to Chernoff). We formalize this approach starting with the notion of "heavy strings," which will form our heavy set.

▶ **Definition 10** (set of heavy strings). *For a distribution $\mathcal{D}$ over m-bit strings and an error parameter ϵ, we define the set of ϵ-heavy strings, $\mathscr{X}(\mathcal{D}, \epsilon)$, to be the set of all strings whose weight under $\mathcal{D}$ is at least $2^{-h/\epsilon}$ where $h = H(\mathcal{D})$ is the Shannon's entropy of $\mathcal{D}$, i.e.,*

$$\mathscr{X}(\mathcal{D}, \epsilon) := \{x \in \{0,1\}^m \mid \Pr_{\mathcal{D}}(x) \geq 2^{-H(\mathcal{D})/\epsilon}\}.$$

We also define $\ell(\mathcal{D}, \epsilon) := \lceil \log |\mathscr{X}(\mathcal{D}, \epsilon)| \rceil$.

It is not hard to see that the set of heavy strings cannot be too large and also that it is a heavy set – i.e., it contains at least $1 - \epsilon$ mass of the distribution. Specifically, we record the following observations.

▶ **Lemma 11** (Set of heavy strings is small). *For every $\mathcal{D}$ and $\epsilon > 0$, it holds that $\ell(\mathcal{D}, \epsilon) \leq \lceil H(\mathcal{D})/\epsilon \rceil$.*

Proof. Let $h = H(\mathcal{D})$. For every string $x \in \mathscr{X}(\mathcal{D}, \epsilon)$ it holds that $\Pr_{\mathcal{D}}(x) \geq 2^{-h/\epsilon}$, and therefore

$$2^{-h/\epsilon} |\mathscr{X}(\mathcal{D}, \epsilon)| \leq \sum_{x \in \mathscr{X}(\mathcal{D}, \epsilon)} \Pr_{\mathcal{D}}(x) \leq 1.$$

It follows that $|\mathscr{X}(\mathcal{D}, \epsilon)| \leq 2^{h/\epsilon}$ which further implies that $\ell(\mathcal{D}, \epsilon) = \lceil \log |\mathscr{X}(\mathcal{D}, \epsilon)| \rceil \leq \lceil h/\epsilon \rceil$. ◀

▶ **Lemma 12** (Set of heavy strings is heavy). *For every $\mathcal{D}$ and $\epsilon > 0$, it holds that $\sum_{x \in \mathscr{X}(\mathcal{D},\epsilon)} \Pr_{\mathcal{D}}(x) \geq 1 - \epsilon$.*

Proof. For a string x, let $p_x := \Pr_{\mathcal{D}}(x)$ denote the weight of x under $\mathcal{D}$. Sample $x \overset{R}{\leftarrow} \mathcal{D}$ and consider the random variable $k_x = \log(1/p_x)$. By definition, the expected value of k_x is simply the entropy of $\mathcal{D}$, i.e., $\mathsf{E}_x[k_x] = h$, and so, by Markov's inequality, $\Pr_{x \overset{R}{\leftarrow} \mathcal{D}}[k_x \geq h/\epsilon] \leq \frac{\mathsf{E}_x[k_x]}{h/\epsilon} = \epsilon$. That is, at least $1-\epsilon$ of the mass belongs to elements x for which $\log(1/p_x) \leq h/\epsilon$, or equivalently, to elements whose weight is at least $2^{-h/\epsilon}$, which is nothing but our desired set $\mathscr{X}(\mathcal{D},\epsilon)$. ◀

▶ **Definition 13** (good hash functions). *Let $\mathcal{D} = \{\mathcal{D}_w\}$ be a family of distributions where $\mathcal{D}_w$ is supported over m_w-long strings and has entropy of h_w. We say that a function family $\mathcal{F}$ is (δ,ϵ)-*good *for $\mathcal{D}$ if there exists a length function $d(h_w,\epsilon,\delta)$ such that for every distribution $\mathcal{D}_w$ in the family:*

$$\Pr_{f \overset{R}{\leftarrow} \mathcal{F}_{m,d}} [f \text{ is injective over } \mathscr{X}(\mathcal{D}_w,\epsilon)] \geq 1 - \delta,$$

where $\mathcal{F}_{m,d}$ denotes the restriction of $\mathcal{F}$ to functions from $\{0,1\}^m$ to $\{0,1\}^d$, $m = m_w$ and $d = d(h_w,\epsilon,\delta)$. We refer to d as the compression *of h. We say that $\mathcal{F}$ has a* representation size *of $L(m,d)$ if each function from m bits to d bits can be represented by $L(m,d)$ bits. We also assume that the family is efficiently computable, i.e., given an index z and an input x one can evaluate $f_z(x)$ in polynomial time.*

Recalling that $\mathscr{X}(\mathcal{D}_w,\epsilon)$ is of size at most 2^s for $s = h_w/\epsilon + 1$, we can use Lemma 8 to derive a (δ,ϵ)-good family with compression $d = 2s + \lceil \log\left(\frac{1}{\delta}\right) \rceil = 2h_w/\epsilon + \log(1/\delta) + O(1)$ and description $L = m + 2d$.

3.1 Hashing-based compression

Let $\mathcal{D} = \{\mathcal{D}_w\}$ be a collection of distributions and assume that $\mathcal{F}$ is (δ,ϵ)-good for $\mathcal{D}$ with compression d. Define the single-round compression protocol $P_{\mathcal{F}}$ as follows: Given the index w of the distribution, the de-compressor specifies a hash function $f_z \in \mathcal{F}_{m,d}$ where $m = m_w$ and $d = d(h_w,\epsilon,\delta)$ that is injective over the set $\mathscr{X}(\mathcal{D}_w,\epsilon')$ and sends the description z of f_z to the compressor, who sends back the value of $y = f_z(x)$. The computationally unbounded de-compressor then checks if y has a pre-image x' in $\mathscr{X}(\mathcal{D},\epsilon')$, and if so it outputs the (unique) preimage x'. Otherwise, the de-compressor outputs a failure symbol $\perp$.

Assuming that $\delta < 1$, the de-compressor always finds a function f_z which is injective over the set $\mathscr{X}(\mathcal{D},\epsilon')$, and so the protocol errs only if x falls out of $\mathscr{X}(\mathcal{D},\epsilon')$. By Lemma 12, this happens with a probability of at most ϵ. Summarizing the above discussion, we get the following lemma.

▶ **Lemma 14** (compression from hashing). *Assuming that $\mathcal{F}$ is a (δ,ϵ)-good hash family for $\mathcal{D} = \{\mathcal{D}_w\}$ for some $\delta < 1$, the protocol $P_{\mathcal{F}}$ is a compression protocol for $\mathcal{D}$ with an error ϵ. For a distribution specified by w, the compressor communicates $d = d(h_w,\epsilon,\delta)$ bits and the de-compressor communicates $L = L(m_w,d)$ bits, and the compressor's computational complexity is $\mathsf{poly}(m_w,d)$, where d is the compression parameter of $\mathcal{F}$ and L is the description length. For the special case of pair-wise independent hashing and $\delta = 0.1$, we get $d = 2h_w/\epsilon + O(1)$ and $L = O(m_w + 2h_w/\epsilon)$.*

The factor of 2 overhead in the compressor's communication can be improved to $1+o(1)$ by using better hash functions (e.g., two-level hashing [10]). We omit the details since it hardly changes the final results of the paper.

The above lemma yields a protocol that obtains the desired compression for the communication from the compressor, but suffers from a high overhead on the communication from the decompressor's end. We will improve this in the next section.

3.2 Improving De-Compressor Communication

In order to improve the communication of the prover, we construct a succinct hash family that is good for efficiently samplable distributions. To sample a hash function we choose a random seed for a PRG, expand it to a long string and use this string to specify a hash function from a family of pair-wise independent hash functions. We show that a PRG against $\mathsf{AM/poly}$ allows us to exponentially compress the description length of a hash function $f : \{0,1\}^m \to \{0,1\}^d$ from $\Omega(m+d)$ to $O(\log(m+d))$. We also extend this result to the case of conditioned efficient distributions at the expense of using a slightly stronger hardness assumption. This extension will be useful for the proof of Theorem 1.

▶ **Theorem 15** (succinct hashing for efficient and conditioned efficient distributions). *Suppose that $E = \mathsf{DTime}(2^{O(n)})$ is hard for exponential-size non-deterministic circuits. Let $\mathcal{D} = \{\mathcal{D}_w\}$ be an efficient family of distributions over m_w-bit long strings having entropy h_w, and let $\epsilon(w)$ be inverse polynomial error parameter. Then, for any constant $\delta < 1$, there exists an efficient family of hash functions that is (δ, ϵ)-good for $\mathcal{D}$ with compression parameter $d = 2h_w/\epsilon + O(1)$ and description size of $L(m,d) = O(\log(m+d))$ bits.*

Moreover, the theorem extends to the case where $\mathcal{D} = \{\mathcal{D}_w\}$ is a family of conditioned efficient distributions, assuming that E is hard for exponential-size non-deterministic NP-circuits.

The high-level idea is to show that when $\mathcal{D}$ is efficiently samplable, there is an $\mathsf{AM/poly}$ protocol for checking whether a given hash function is injective over the set $\mathscr{X}(\mathcal{D}, \epsilon)$. Therefore, if we use a PRG that fools $\mathsf{AM/poly}$ to sample a hash function f from a collection $\mathcal{F}$, the probability that f will be injective over $\mathscr{X}(\mathcal{D}, \epsilon)$ is almost the same as the probability that a *random* member of $\mathcal{F}$ will be injective. The theorem then follows by taking $\mathcal{F}$ to be a family of pair-wise independent hash functions (for which we know that a random member is injective whp). Unfortunately, we do not know how to construct an $\mathsf{AM/poly}$ protocol that certifies injectivity, however, we can use an approximate version of this property that suffices for our purposes. We continue with formal proof.

Proof of Theorem 15. Let $T_1(|w|)$ be the complexity of $\mathcal{D}$ and let $T_2(|w|)$ denote the complexity of evaluating the pair-wise independent hash functions $\mathcal{F}_{m,d} = \{f_z\}_{z \in \{0,1\}^L}$ promised in Lemma 8 where $m = m_w$ and $d = 2h_w/\epsilon + \lceil \log\left(\frac{2}{\delta}\right) \rceil + 2$ and $L = m + 2d$. Note that for these parameters $\mathcal{F}_{m,d}$ is $(\delta/2, 1 + h/\epsilon)$-injective (Lemma 8). Let T be some fixed polynomial in $T_1(|w|) + T_2(|w|)$ whose value will be determined later, and let $\mathsf{G} : \{0,1\}^k \to \{0,1\}^L$ be a PRG that $\frac{\delta}{2}$-fools T-size $\mathsf{AM/poly}$ with seed length $k = O(\log L)$ whose existence is promised by Theorem 5. Consider the family of functions $\mathcal{F}'_{m,d}$ whose members f'_s are identified by an index $s \in \{0,1\}^k$, and are defined by $f'_s = f_{\mathsf{G}(s)}$ where f_z is the function from $\mathcal{F}_{m,d}$ whose index is z. Note that $\mathcal{F}'_{m,d}$ is computable in time $\mathsf{poly}(|w|)$. We claim that $\mathcal{F}'_{m,d}$ is (δ, ϵ)-good for $\mathcal{D}_w$ for every w.

Fix some w. We begin by introducing a promise problem Π_w over L-bit strings. (Recall that $L = m_w + 2d(w)$.)

- Yes instance: A string $z \in \{0,1\}^L$ is a yes instance if the function $f_z : \{0,1\}^m \to \{0,1\}^d$ is *not injective* over the set $\mathscr{X}(\mathcal{D}_w, \epsilon)$.
- No instance: A string $z \in \{0,1\}^L$ is a No instance if the function $f_z : \{0,1\}^m \to \{0,1\}^d$ is *injective* over the set $\mathscr{X}'(\mathcal{D}_w, \epsilon) = \{x \in \{0,1\}^m \mid \Pr_{\mathcal{D}_w}(x) \geq 0.5 \cdot 2^{-(h_w/\epsilon)}\}$.

We show that the above promise problem admits an $\mathsf{AM/poly}$ proof system. Let ρ_w denote the randomness complexity of the distribution $\mathcal{D}_w$. Let $A(w;\cdot)$ be the PPT algorithm for sampling from $\mathcal{D}_w$. Consider the following protocol with prover P and verifier V and common input z:

1. P sends two strings $(x_0, x_1) \in \{0,1\}^m \times \{0,1\}^m$ to V.
2. V checks if $x_0 \neq x_1$ and $f_z(x_0) = f_z(x_1)$. If the checks fail, then it aborts with output 0. Otherwise, the parties proceed further.
3. For $b \in \{0,1\}$, the parties run the following set membership protocol for the string x_b:
   a. Consider the set $R_{x_b} = \{r \in \{0,1\}^{\rho_w} \mid A(w;r) = x_b\}$.
   b. Run a set lower-bound protocol for set R_{x_b} with size parameter $\alpha = 2^{\rho_w} \cdot 2^{-h/\epsilon}$. For membership queries to the set R_{x_b} for a string r, just check whether $A(w;r) = x_b$.
4. V outputs 1 if both the checks succeed. Otherwise, it outputs 0.

▷ Claim 16. The above protocol is an $\mathsf{AM/poly}$ protocol for Π_w.

Proof. Suppose that z is a Yes instance. That is, f_z is not injective over the set $\mathscr{X}(\mathcal{D}_w, \epsilon)$. Then, an honest P will be able to find two strings $(x_0, x_1) \in \mathscr{X}(\mathcal{D}_w, \epsilon) \times \mathscr{X}(\mathcal{D}_w, \epsilon)$ such that $x_0 \neq x_1$ and $f_z(x_0) = f_z(x_1)$. In our protocol, the checks $x_0 \neq x_1$ and $f_z(x_0) = f_z(x_1)$ will always succeed in this case. Since both x_0 and x_1 belong to the set $\mathscr{X}(\mathcal{D}_w, \epsilon)$, this implies that both the sets R_{x_0} and R_{x_1} have size at least α according to the definition of $\mathscr{X}(\mathcal{D}_w, \epsilon)$. Hence, both the set lower-bound protocols correspond to YES instances and either of these will fail with probability at most 0.1. It follows that the total failure probability is at most 0.2.

We move on to the case where z is a No instance, i.e., the function f_z is injective over the set $\mathscr{X}'(\mathcal{D}_w, \epsilon)$ for all $w \in \{0,1\}^*$. Fix the pair $(x_0, x_1) \in \{0,1\}^m \times \{0,1\}^m$ that the prover sends in the first step and assume that the verifier did not reject in the second step, i.e., $x_0 \neq x_1$ and $f_z(x_0) = f_z(x_1)$. Then, at least one of x_0 or x_1 must lie outside the set $\mathscr{X}'(\mathcal{D}_w, \epsilon)$ since f_z is injective over this set. Therefore, either the size of R_{x_0} or that of R_{x_1} must be smaller than $\alpha/2 = 2^{\rho_w} \cdot 0.5 \cdot 2^{-(h/\epsilon)}$. It follows that, except with probability 0.1, the verifier rejects in at least one of the set lower-bound protocols. ◁

By hard-wiring w and h_w, we implement the verifier by a non-uniform circuit of size polynomial in $T_1(|w|)+T_2(|w|)$. We can therefore take $T(|w|)$ to be complexity of the verifier, and conclude that

$$\Pr_s[f_{\mathsf{G}(s)} \text{ is injective over } \mathscr{X}(\mathcal{D}_w, \epsilon)] > \Pr_s[f_{\mathsf{G}(s)} \text{ is injective over } \mathscr{X}'(\mathcal{D}_w, \epsilon)]$$
$$> \Pr_z[f_z \text{ is injective over } \mathscr{X}'(\mathcal{D}_w, \epsilon)] - \frac{\delta}{2} > 1 - \frac{\delta}{2} - \frac{\delta}{2} = 1 - \delta.$$

The last inequality follows by recalling that $\mathcal{F}_{m,d}$ is $(\delta/2, 1 + h_w/\epsilon)$-injective and since that set $\mathscr{X}'(\mathcal{D}_w, \epsilon)$ is of size at most $2^{1+h_w/\epsilon}$. We conclude that $\mathcal{F}'_{m,d}$ is (δ, ϵ)-good for $\mathcal{D}_w$, as required. The first part of the theorem follows.

The "Moreover" part

The proof of the second part is similar with the following modification. We take G to be a PRG that 0.1-fools T-size non-deterministic NP-circuits (whose existence follows from the underlying assumption via Theorem 5), and show that Π_w can be decided by such circuits. Given a string z, the circuit C_w non-deterministically guesses a pair of m-bit strings (x_0, x_1) and verifies that $x_0 \neq x_1$ and $f_z(x_0) = f_z(x_1)$. (If any of these conditions fail, the circuit rejects.) Next, C_w derives for $b \in \{0,1\}$, an α-approximation q_b for the quantity $p_b = \Pr[\mathcal{D}_w = x_b]$ for $\alpha = 0.2$, and accepts if and only if q_0 and q_1 are both larger than $0.7 \cdot 2^{-(h_w/\epsilon)}$. (Here h_w is hard-wired to C_w.) The approximation q_b is obtained by using the Approximate Counting algorithm (Theorem 7) as follows. Recall that $\mathcal{D}$ is defined by a PPT conditional sampler $A(w; \cdot)$ with randomness complexity ρ_w such that

$$p_b = \Pr[\mathcal{D}_w = x_b] = \frac{|\{r \in \{0,1\}^{\rho_w} : A(w; r) = x_b\}|}{|\{r \in \{0,1\}^{\rho_w} : A(w; r) \neq \bot\}|}.$$

Hence, to derive an α-approximation of p_b it suffices to get a $\alpha/2$-approximation of both the denominator and numerator. This can be done by a polynomial-size NP-circuit since these sets are recognizable by polynomial-size circuits (whose size is the sum of the complexity of A). It remains to prove the following claim.

▷ Claim 17. The circuit C_w accepts Yes instances and rejects No instances of Π_w.

Proof. Suppose that z is a Yes instance. That is, f_z is not injective over the set $\mathscr{X}(\mathcal{D}_w, \epsilon)$. Then there exists an f_z-collision $x_0 \neq x_1 \in \mathscr{X}(\mathcal{D}_w, \epsilon) \times \mathscr{X}(\mathcal{D}_w, \epsilon)$. Since both x_0 and x_1 belong to the set $\mathscr{X}(\mathcal{D}_w, \epsilon)$, this implies that p_0 and p_1 are at least $2^{-h_w/\epsilon}$ and so q_0 and q_1 are larger than $0.8 \cdot 2^{-h_w/\epsilon}$ and C_w accepts.

We move on to the case where z is a No instance, i.e., the function f_z is injective over the set $\mathscr{X}'(\mathcal{D}_w, \epsilon)$ for all $w \in \{0,1\}^*$. Then, for any f_z-collision $x_0 \neq x_1$ either $p_0 < 0.5 \cdot 2^{-h_w/\epsilon}$ or $p_1 < 0.5 \cdot 2^{-h_w/\epsilon}$. This means that either q_0 or q_1 must be smaller than $1.2 \cdot 0.5 \cdot 2^{-h_w/\epsilon} \leq 0.6 \cdot 2^{-h_w/\epsilon}$, and C_w rejects. ◁

The rest of the argument is identical to the proof of the first part of the theorem. ◀

Together with Lemma 14, we derive the following theorem.

▶ **Theorem 18.** *Suppose that E = $\mathsf{DTime}(2^{O(n)})$ is hard for exponential-size non-deterministic circuits. Let $\mathcal{D} = \{\mathcal{D}_w\}$ be an efficient family of distributions over m_w-bit long strings having entropy h_w, and let $\epsilon(w)$ be inverse polynomial error parameter. Then, there exists a single-round compression protocol for $\mathcal{D}$ with an error ϵ and communication of $d = 2h_w/\epsilon + O(1)$ for the compressor and $L = O(\log(m_w + h_w/\epsilon))$ for the de-compressor. Furthermore, the compressor is efficient.*

Moreover, the theorem extends to the case where $\mathcal{D} = \{\mathcal{D}_w\}$ is a family of conditioned efficient distributions, assuming that E is hard for exponential-size non-deterministic NP-circuits.

4 Reducing the Communication in Interactive Proofs

In this section, we prove the main theorem (Theorem 1). Roughly speaking, we compress each message that the verifier sends by using a properly chosen hash function. We begin with some notations and definitions.

Let $\langle \mathsf{P}, \mathsf{V} \rangle$ be an interactive proof for a promise problem Π. For an input x, let $T(|x|)$ and $\rho(|x|)$ denote the running-time and randomness complexity of the verifier.We assume that the parties speak in alternating turns and that the prover sends the first and last message. (The latter assumption always holds and the former can be guaranteed at the expense of adding an additional empty round). Letting $I'(|x|)$ denote the number of rounds in which the verifier speaks, we get that the the total number of rounds is $I(|x|) = 2I'(|x|) + 1$. We let a_i and b_i denote the ith message of the prover and verifier, respectively, and let $b_{I(|x|)+1}$ denote the final verdict of the verifier (accept/reject).

We can think of the V as a machine that takes an input x a random tape r and a sequence of prover's messages $a = (a_i)_{1 \le i \le k}, k \le I' + 1$ and outputs the message b_k. For a partial transcript $w = (x, a = (a_i)_{i \in [k]}, b = (b_i)_{i \in [k-1]})$, consider the probability distribution $\mathcal{D}_w$ of the verifier's next message b_k conditioned on seeing the partial transcript w. We can describe $\mathcal{D}_w$ as the output of the following randomized process: Sample a random tape $r \overset{R}{\leftarrow} \{0,1\}^{\rho(|x|)}$ conditioned on the event

$$\bigwedge_{1 \le j \le k-1} \mathsf{V}(x, a_1, \ldots, a_j; r) = b_j \tag{1}$$

and output the string $b_k = \mathsf{V}(x, a_1, \ldots, a_{k-1}; r)$. Note that $\mathcal{D} = \{\mathcal{D}_w\}$ is *conditioned efficient*: consider a PPT conditional sampler A, which on input w as above and random tape $r \in \{0,1\}^{\rho(|x|)}$, outputs $\perp$ if (1) does not hold, and outputs b_k otherwise. Let h_w and m_w denote the entropy and domain bit-length of $\mathcal{D}_w$, and let $\epsilon(w) = 0.01/I(|x|)$ where x is the first entry of w. Let $\mathcal{F}$ denote a family of hash functions (promised by the second part of Theorem 15) which is $(0.2, \epsilon)$-good for $\mathcal{D}$ and for $\mathcal{D}_w$ achieves compression of $2h_w/\epsilon + O(1)$ and description size of $O(\log(m_w + h_w/\epsilon)) < O(\log(m_w/\epsilon)) < O(\log m_w) + O(\log I(|x|))$ where the first inequality follows by noting that $h_w \le m_w$.

The new proof system

We define the new proof system $\langle \mathsf{P}', \mathsf{V}' \rangle$ as follows. Given x as a common input and randomness ρ for the verifier, the parties initialize an "emulated" transcript $w = x$ and proceed for $i = 1, \ldots, I' - 1$ rounds as follows.

1. P': Compute a_i by calling $\mathsf{P}(w)$ and locally update $w = (w, a_i)$. Choose hash function f_i from $\mathcal{F}$ that is injective over $\mathscr{X}(\mathcal{D}_w, \epsilon)$, and send (a_i, f_i). (Recall that $\mathscr{X}(\mathcal{D}_w, \epsilon)$ is the set of strings whose weight under $\mathcal{D}_w$ is at least $2^{-h/\epsilon}$ where h is the Shannon's entropy of $\mathcal{D}_w$; See Definition 10.)
2. V': Compute b_i by calling $\mathsf{V}(x, a_1, \ldots, a_i; \rho)$ where a_i is the ith message sent by the prover, and send $b'_i = f_i(b_i)$.
3. Before proceeding to the next iteration the prover locally computes b_i by choosing the unique string in $\mathscr{X}(\mathcal{D}_w, \epsilon)$ that maps to b'_i. If this string is not unique the prover sends a special abort symbol and the verifier terminates with rejection. Otherwise, the prover updates its view to $w := (w, b_i)$.

At the last round, the prover sends $a_{I'}$ by calling $\mathsf{P}(w)$ and the verifier outputs its verdict $b_{I'+1}$ by calling $\mathsf{V}(x, a_1, \ldots, a_t; \rho)$.

Completeness and Soundness

Let γ be the completeness error of the original proof system. For a yes instance x, it holds that

$$\Pr[\langle \mathsf{P}', \mathsf{V}' \rangle(x) = 1] \ge \Pr_r[\langle \mathsf{P}, \mathsf{V} \rangle(x) = 1] - \Pr_r[\text{decoding failure}] \ge 1 - \gamma - \epsilon \cdot (I' - 1) \ge 1 - \gamma - 0.01,$$

where the second inequality follows from a union-bounds over all the rounds. For soundness, fix a No instance x, and observe that any cheating strategy for the prover P' in the new proof system translates to a cheating strategy in the original proof system. Indeed, if for each $i \in [I']$, P' maliciously chooses a_i and f_i based on b'_{i-1} and its internal state

$S = (x, a = (a_j)_{j<i}, f = (f_j)_{j<i}, b' = (b'_j)_{j<i-1})$, we can define a cheating prover for the original system that maintains the same state S and given b_{i-1} executes P' on the state S and on $b'_{i-1} = f_{i-1}(b_{i-1})$. The probability that the verifier V accepts is exactly the same probability that V' does. Therefore the soundness error of the new system is the same as the soundness error of the original system.

Communication complexity

We begin by analyzing the expected communication complexity under the assumption that P is honest. Fix x, and let $I = I(|x|)$,$I' = I'(|x|)$. Let $w = (x, (a_i, b_i)_{i \in [I']})$ denote the random variable that describes a random transcript and let $w[k] = (x, (a_i, b_i)_{i \in [k]})$ denote the kth prefix of the transcript. We can assume that the honest prover is deterministic (i.e., a_i is a deterministic function of $x, (b_j)_{j<i}$) and so all the randomness is due to the b part. Recall that in each round i, the verifier sends a string b'_i of length $2h_{w[i]}/\epsilon + O(1)$ where $h_{w[i]}$ is the entropy of $\mathcal{D}_{w[i]}$. Note that $h_{w[i]}$ is a random variable (since $w[i]$ is a random variable). Thus, the communication complexity of the verifier is given by the following random variable

$$\sum_{i \in [I']} 2h_{w[i]}/\epsilon + O(1) = O(I') + 2/\epsilon \sum_{i \in [I']} h_{w[i]} = O(I) + O(I) \sum_{i \in [I']} h_{w[i]}.$$

By the chain rule, the expected value of $\sum_{i \in [I']} h_{w[i]}$ is the entropy of w which is at most the randomness complexity of the verifier. Overall, the expected communication complexity of the verifier is $O(I \cdot \rho)$. The communication of the prover in the ith iteration consists of the original communication (the a_i part) and the description of the hash functions which is of length $O(\log m_{w[i]}) + O(\log I(|x|))$. Overall the prover communication grows by at most $O(I(\log I + \log m))$ where m is the maximal length of the verifier's message in the original scheme. Since m and I are polynomially bounded in n, this can be written as $O(I(\log n))$.

Deriving Theorem 1

The communication analysis is only on expectation and it assumes that the prover is honest. To get a worst case bound, we slightly modify the proof system by letting the verifier halt the interaction (with rejection) if she communicates more than, say 100 times the expected communication complexity. By Markov's inequality, this increases the completeness error by at most 0.01. Theorem 1 follows.

5 Conclusion

Compressing interactive protocols is a problem of fundamental nature in information theory. When computational constraints are also involved, it leads to a question that combines complexity theory and information theory into a natural, yet difficult problem. In this paper, we partially answered the problem posed at the beginning: whether the communication from a verifier in an interactive proof system can always be reduced to the level of randomness used by the verifier, without increasing the verifier's randomness, the round complexity or the prover's communication significantly. We leave it as an open question if such a result is possible without relying on complexity assumptions (or using weaker ones), and if quantitative improvements can be achieved over our result.

En-route to our main result, we encounter several interesting problems. While our focus is on proof systems, the compression results here extend to any 2-party protocol where one party is computationally unbounded, and the other party is randomized but has no private inputs. Further, the special case of the single-round compression problem is of significance in its own right. The notion of efficiently conditional distributions that we introduced, being natural, could be of independent interest.

References

1 Benny Applebaum, Sergei Artemenko, Ronen Shaltiel, and Guang Yang. Incompressible functions, relative-error extractors, and the power of nondeterministic reductions. *Comput. Complex.*, 25(2):349–418, 2016. `doi:10.1007/S00037-016-0128-9`.

2 Benny Applebaum and Eyal Golombek. On the Randomness Complexity of Interactive Proofs and Statistical Zero-Knowledge Proofs. In Stefano Tessaro, editor, *2nd Conference on Information-Theoretic Cryptography (ITC 2021)*, volume 199 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 4:1–4:23, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.ITC.2021.4`.

3 S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2006. URL: `https://theory.cs.princeton.edu/complexity/book.pdf`.

4 László Babai and Shlomo Moran. Arthur-merlin games: A randomized proof system, and a hierarchy of complexity class. *J. Comput. Syst. Sci.*, 36(2):254–276, April 1988. `doi:10.1016/0022-0000(88)90028-1`.

5 Marshall Ball, Ronen Shaltiel, and Jad Silbak. Non-malleable codes with optimal rate for poly-size circuits. *Electron. Colloquium Comput. Complex.*, TR23-167, 2023. `arXiv:TR23-167`.

6 Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. Derandomization in cryptography. *SIAM J. Comput.*, 37(2):380–400, 2007.

7 J.Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979. `doi:10.1016/0022-0000(79)90044-8`.

8 Andrew Drucker. Nondeterministic direct product reductions and the success probability of SAT solvers. In *FOCS*, pages 736–745. IEEE Computer Society, 2013.

9 Uriel Feige and Carsten Lund. On the hardness of computing the permanent of random matrices. *Computational Complexity*, 6(2):101–132, 1997.

10 Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a sparse table with O(1) worst case access time. *J. ACM*, 31(3):538–544, June 1984. `doi:10.1145/828.1884`.

11 Oded Goldreich. On promise problems: A survey. In Oded Goldreich, Arnold L. Rosenberg, and Alan L. Selman, editors, *Theoretical Computer Science, Essays in Memory of Shimon Even*, volume 3895 of *Lecture Notes in Computer Science*, pages 254–290. Springer, 2006. `doi:10.1007/11685654_12`.

12 Oded Goldreich. *Computational complexity - a conceptual perspective*. Cambridge University Press, 2008. `doi:10.1017/CBO9780511804106`.

13 Oded Goldreich and Maya Leshkowitz. *On Emulating Interactive Proofs with Public Coins*, pages 178–198. Springer International Publishing, Cham, 2020. `doi:10.1007/978-3-030-43662-9_12`.

14 Oded Goldreich and Avi Wigderson. Derandomization that is rarely wrong from short advice that is typically good. In *RANDOM*, volume 2483 of *Lecture Notes in Computer Science*, pages 209–223. Springer, 2002.

15 S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, pages 291–304, New York, NY, USA, 1985. Association for Computing Machinery. `doi:10.1145/22145.22178`.

16 S Goldwasser and M Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, pages 59–68, New York, NY, USA, 1986. Association for Computing Machinery. `doi:10.1145/12130.12137`.

17 Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. Uniform hardness versus randomness tradeoffs for arthur-merlin games. *Computational Complexity*, 12(3-4):85–130, 2003.

18 Russell Impagliazzo and Avi Wigderson. P = bpp if e requires exponential circuits: Derandomizing the xor lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 220–229, New York, NY, USA, 1997. Association for Computing Machinery. doi:10.1145/258533.258590.

19 Mark Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theor. Comput. Sci.*, 43:169–188, 1986.

20 Adam Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Comput.*, 31(5):1501–1526, 2002.

21 Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM Journal on Computing*, 31(5):1501–1526, 2002. doi:10.1137/S0097539700389652.

22 Maya Leshkowitz. Round complexity versus randomness complexity in interactive proofs. *Theory Comput.*, 18:1–65, 2022. URL: https://theoryofcomputing.org/articles/v018a013/.

23 Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, October 1992. doi:10.1145/146585.146605.

24 Peter Bro Miltersen and N. V. Vinodchandran. Derandomizing arthur-merlin games using hitting sets. *Computational Complexity*, 14(3):256–279, 2005.

25 Moni Naor, Leonard J. Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 182–191. IEEE Computer Society, 1995. doi:10.1109/SFCS.1995.492475.

26 A. Orlitsky. Worst-case interactive communication. i. two messages are almost optimal. *IEEE Transactions on Information Theory*, 36(5):1111–1126, 1990. doi:10.1109/18.57210.

27 Rasmus Pagh. Hash and displace: Efficient evaluation of minimal perfect hash functions. In Frank K. H. A. Dehne, Arvind Gupta, Jörg-Rüdiger Sack, and Roberto Tamassia, editors, *Algorithms and Data Structures, 6th International Workshop, WADS '99, Vancouver, British Columbia, Canada, August 11-14, 1999, Proceedings*, volume 1663 of *Lecture Notes in Computer Science*, pages 49–54. Springer, 1999. doi:10.1007/3-540-48447-7_5.

28 Ronen Shaltiel. Weak derandomization of weak algorithms: Explicit versions of yao's lemma. *Comput. Complex.*, 20(1):87–143, 2011. doi:10.1007/S00037-011-0006-4.

29 Ronen Shaltiel and Jad Silbak. Explicit codes for poly-size circuits and functions that are hard to sample on low entropy distributions. *Electron. Colloquium Comput. Complex.*, TR23-149, 2023. arXiv:TR23-149.

30 Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *J. ACM*, 52(2):172–216, 2005.

31 Ronen Shaltiel and Christopher Umans. Pseudorandomness for approximate counting and sampling. *Computational Complexity*, 15(4):298–341, 2006.

32 Ronen Shaltiel and Christopher Umans. Low-end uniform hardness versus randomness tradeoffs for AM. *SIAM J. Comput.*, 39(3):1006–1037, 2009.

33 Adi Shamir. Ip = pspace. *J. ACM*, 39(4):869–877, October 1992. doi:10.1145/146585.146609.

34 Claude E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(3):379–423, 1948. doi:10.1002/J.1538-7305.1948.TB01338.X.

35 Michael Sipser. A complexity theoretic approach to randomness. In *STOC*, pages 330–335. ACM, 1983.

36 Larry J. Stockmeyer. The complexity of approximate counting (preliminary version). In *STOC*, pages 118–126. ACM, 1983.

37 Luca Trevisan and Salil P. Vadhan. Extracting randomness from samplable distributions. In *FOCS*, pages 32–42. IEEE Computer Society, 2000.

38 Luca Trevisan, Salil P. Vadhan, and David Zuckerman. Compression of samplable sources. *Comput. Complex.*, 14(3):186–227, 2005. doi:10.1007/S00037-005-0198-6.

Are Your Keys Protected? Time Will Tell

Yoav Ben Dov
Weizmann Institute of Science, Rehovot, Israel

Liron David
Weizmann Institute of Science, Rehovot, Israel

Moni Naor
Weizmann Institute of Science, Rehovot, Israel

Elad Tzalik
Weizmann Institute of Science, Rehovot, Israel

Abstract

Side channel attacks, and in particular timing attacks, are a fundamental obstacle to obtaining secure implementation of algorithms and cryptographic protocols, and have been widely researched for decades. While cryptographic definitions for the security of cryptographic systems have been well established for decades, none of these accepted definitions take into account the running time information leaked from executing the system. In this work, we give the foundation of new cryptographic definitions for cryptographic systems that take into account information about their leaked running time, focusing mainly on keyed functions such as signature and encryption schemes. Specifically,

(1) We define several cryptographic properties to express the claim that the timing information does not help an adversary to extract sensitive information, e.g. the key or the queries made. We highlight the definition of *key-obliviousness*, which means that an adversary cannot tell whether it received the timing of the queries with the actual key or the timing of the same queries with a random key.

(2) We present a construction of *key-oblivious pseudorandom permutations* on a small or medium-sized domain. This construction is not "fixed-time," and at the same time is secure against any number of queries even in case the adversary knows the running time exactly. Our construction, which we call *Janus Sometimes Recurse*, is a variant of the "Sometimes Recurse" shuffle by Morris and Rogaway.

(3) We suggest a new security notion for keyed functions, called *noticeable security*, and prove that cryptographic schemes that have noticeable security remain secure even when the exact timings are leaked, provided the implementation is key-oblivious. We show that our notion applies to cryptographic signatures, private key encryption and PRPs.

2012 ACM Subject Classification Theory of computation → Cryptographic primitives; Security and privacy → Key management; Security and privacy → Mathematical foundations of cryptography; Security and privacy → Side-channel analysis and countermeasures

Keywords and phrases Side channel attacks, Timing attacks, Keyed functions, Key oblivious, Noticeable security

Digital Object Identifier 10.4230/LIPIcs.ITC.2024.3

Related Version *Full Version*: `https://ia.cr/2024/906`

Funding Research supported in part by grants from the Israel Science Foundation (no.2686/20), by the Simons Foundation Collaboration on the Theory of Algorithmic Fairness and by the Israeli Council for Higher Education (CHE) via the Weizmann Data Science Research Center.

1 Introduction

In any implementation of a cryptographic scheme there is a disparity between the mathematical specification of its functionality and the actual implementation in a physical device and environment. By nature, a physical implementation leaks more information than was intended and this leakage is known as a side-channel. In this work we concentrate on the running time, the side channel that is perhaps the hardest to block (and the easiest to exploit), since the time it took to perform a certain service is often known.

Consider the security of encryption and signature schemes (which we refer to as keyed cryptographic functions). After much work, definitions of the security of such schemes have been well-established for decades, and may be considered one of the crown achievements of the foundations of cryptography. But none of these accepted definitions take into account the running time information leaked from executing the system.

In this work, we give a foundation for defining the security of cryptographic systems that take into account that their running time is *leaked*, focusing mainly on keyed functions such as signature and encryption schemes. More specifically, we suggest several cryptographic definitions for scenarios where the leaked information does not help an adversary to expose sensitive information, e.g. the key or the queries made. The most interesting definition we propose is *key-oblivious*. For this notion we prove that for cryptographic schemes such as digital signatures, private key encryption and pseudorandom permutations (PRPs), if their implementation satisfies key-obliviousness, then they preserve their security even when the *exact* timing is leaked. Finally, we construct a PRP called "Janus Sometimes Recurse (JSR)" that is not fixed-time, yet provably secure against timing attacks (key-oblivious and other properties).

For a motivating example of the JSR construction, consider the following question, taken more or less verbatim from Stack Overflow[1]: "*I am looking to enumerate a random permutation of the numbers* $1 \ldots N$ *in fixed space. I cannot store all numbers in a list,* N *can be very large, more than available memory. I still want to be able to walk through such a permutation of numbers one at a time, visiting each number exactly once.*"

As we shall see, there are good cryptographically based solutions to the question, but they are susceptible when the creator of the permutation leaks how long it took to choose the next value. In light of the work on JSR in Section 4.1 we have a good solution that is immune even when the timing information is leaked.

1.1 A Brief History of Timing Attacks

There is a significant body of research about side-channel attacks and more specifically timing attacks, and how to exploit them in order to break cryptographic protocols. An early work by Lipton and Naughton [24] showed a way to exploit timing information to compromise the performance of dictionaries that employ universal hash functions.

Kocher [23] showed how the running time of certain implementations of RSA and Diffie Hellman schemes leaks information that can be used to recover the prime factors or find the discrete log, hence breaking the systems. In more detail, Kocher showed that the running time of some implementations depends on the exponent chosen in the protocol, and by carefully timing the running time on multiple outputs one can extract information about

[1] `https://stackoverflow.com/questions/10054732/create-a-random-permutation-of-1-n-in-constant-space`.

the exponent, which can then be used to break the security of the protocol. Kocher's work brought widespread attention to the crucial importance of implementations of cryptographic protocols, such as public-key encryption and signatures, and led to a considerable body of research on vulnerabilities to side-channel attacks and spurring studies and advancements aimed at strengthening their security against such attacks.

Brumley and Boneh [11] showed that timing attacks are practical on large systems and over the web. In large systems, the response time suffers significantly from noise coming from latency, multi-threading, communication bottlenecks and more. Brumley and Boneh showed that even under these conditions it is possible to retrieve the private key with timing attacks on OpenSSL servers. More specifically, they showed how to reconstruct the prime factors used in an RSA protocol by making about one million queries and carefully inspecting the response times.

One of the most efficient lattice-based digital signature schemes is BLISS, suggested by Ducas, Durmus, Lepoint and Lyubashevsky [14]. This scheme uses a bimodal Gaussian sampler and was shown to be vulnerable to timing attacks, and, in particular, the sampling component that is not independent of the secret-key [15, 10], as well as other attacks.

All of these examples are but a drop in the ocean of the vast and rich research area of side-channel attacks. New and more sophisticated and subtle attacks and vulnerabilities are found every once in a while, a solution is suggested and implemented, and then another attack is found, in what feels like a never-ending game of cat and mouse. For a more thorough (but still far from full) overview of the history and background of side-channel attacks and timing attacks in particular see Crosby, Wallach and Riedi [13] and Biswas, Ghosal and Nagaraja [8].

1.2 Prevention Techniques

The main approach to prevent timing attacks is to use *fixed-time algorithms*, *often called in the literature "constant time algorithms,"* meaning algorithms that run the same *amount of* time on all inputs.

There are two main drawbacks to this solution. First, in order for the algorithm to run in fixed-time on all the inputs, we need to know the worst-case running time, a task that is often challenging on its own. The second one is that even if we do know the running time, in many cases there is a very large gap between *best case and worst case running time*, or even *average case and worst case running time*, and by making the algorithm run in the worst case time on all inputs, we create huge overheads. It is also worth mentioning that the second caveat can make many protocols and algorithms impractical and not usable when efficiency is critical.

In addition, the survey in Section 1.1 demonstrates that the task of making an algorithm run in fixed-time is more subtle and challenging than meets the eye. Timing information might leak from the response times of the server, from I/O calls, from reading RAM memory or cache memory, and many more possibilities. For the implemented algorithm to be truly and fully fixed-time one must make sure to make everything fixed-time, which is often very challenging and goes against hardware and software optimizations.

A common technique to thwart timing attacks in the public-key context is *"blinding,"* first suggested by Chaum [12] in the context of signatures, where a value v is mapped into a random-looking one u prior to the encryption or signature, in a manner that allows retrieving the desired signature or encryption from the encryption or signature on u. Kocher [23] suggested using blinding to make RSA implementations secure against timing attacks. The blinding works by multiplying the input x by a fresh random element r of the group $\mathbb{Z}_N{}^*$, i.e.

a random element which is co-prime to N. To decode, multiplication by the group inverse r^{-1} is done at the end of the computation. Note that simply using the same r for many inputs will not work, as the attack suggested by Kocher can recover r over time, and even recover the exponent without knowing r. Hence, fresh r needs to be chosen in each round. This example goes to show that using blinding as a technique to protect against timing attacks is often a subtle task, and that if implemented naively or incorrectly can lead to a false sense of security.

A general approach to preventing leakage is to employ techniques from secure multi-party computation, and split the input into various parts where leaking *almost* all of the parts does not reveal the actual values. It was first suggested in Ishai, Sahai and Waters [20] for thwarting probing attacks (see Kalai and Reyzin [21] for a survey). This can be thought of as the "moral equivalent" of blinding for a general function. However, in the case of timing, at the very least the *sum* of their running times is leaked (since they are all executed on the same machine, what is leaked in the *total* time of all the emulated processors) and this is a function of *all* parties. Hence this does not solve the problem, unless an argument is made that the sum of all the execution times is not meaningful.

Another set of techniques is known as the bounded retrieval model (see [2] for a survey). In this model, the adversary learns some arbitrary function of the secret key that is shorter than the key. An alternative is the noisy leakage model, where the leakage is not of bounded length but it is guaranteed that the secret key is still unpredictable given the leaked value [30]. But this is not the case in our setting, with a repeatedly used keyed function: the adversary learns the timing of the keyed function on *many* inputs. Altogether this leaked information may be much greater than the key size. There is also the continual-leakage model, which is more appropriate for this case. The work of Goldwasser and Rothblum [18] considered leakage with an unbounded number of executions, in the presence of an adversary who observes partial information on the internal state of the computation during the executions. They showed that it is possible to obtain secure computation in the sense that the adversary learns only input-output behavior if the leakage in any round is bounded (following the "only computation leaks information" maxim of Micali and Reyzin [27]). However, this is not a silver bullet for timing attacks, as in this work the first step is to turn the program to be computed into a circuit, i.e. into a fixed-time computation - and this carries over the various downsides of this approach, for instance, that typical case becomes the worst case. Nevertheless, such an approach may be useful for various critical sections when one wants to get fixed time. *Our goal in this work is to relax fixed time and allow information related to the key to leak, but specify what it means to say that it is not harmful.*

Extending the Notion of Constant Time

There have been a number of proposals to extend the notion of constant time implementations in order to argue that no meaningful information is leaked from the timing. For instance, Benegas et al. [4] talk about the distribution of the running time being the same for any key and any input. Similarly, Almeida et al. [1] define a program to be secure if all equivalent programs in terms of inputs and outputs are indistinguishable given the leaked information, i.e. it "means that any two executions whose input and output values differ only with respect to secret information must leak exactly the same observation." These extensions are not flexible enough to talk about protecting keyed cryptographic functions, since the protection there is computational, and the inputs and outputs are not going to be identical. For instance consider the case of signature schemes.

Note that we use the term "fixed-time" since in the literature constant time sometimes does not refer to operations that take the same amount of time no matter what the input is.

1.3 Comparing Our Work With the Existing Ones

At this point, the reader may be wondering whether enough theoretical work was already done in the area of leakage and there is not much to add. The novel aspect of our work is proposing criteria for arguing that the leakage is benign, that is, the presence of this benign leakage, although not being fixed, does not compromise the original guaranteed security in many cases. Maintaining this criteria is, therefore, sufficient to argue security even with the presence of the leakage in many cases. An illustrative example is the famous GGM construction of pseudorandom functions (PRF) F from length-doubling pseudorandom generators (PRG) G. *What properties should we require of the PRG G in order to argue that F is secure?* Recall that the construction is defined by imagining a full binary tree of depth n where each node gets an n bit label. The root is labeled with the key k and each parent induces a labeling of its two children by applying G to its label; the left half of the result becomes the label of the left child and the right half the label of the right child. Clearly, requiring that G be fixed time and making the rest of operations (deciding whether to branch left or right based on the bit) fixed-time is sufficient. But can we get a weaker requirement from G and how to express it? What happens when the construction is not applied a fixed number of times, but one that can vary with the input? Could such a construction be secure?

In this work we define a formal condition that is sufficient to argue security in the presence of leakage in many cases. We call this condition “key-oblivious.” Namely, in order to prove that a construction is secure in the presence of leakage, one only needs to prove that the construction is key-oblivious. The key-obliviousness then implies security in the presence of leakage. We argue that this notion is easier to reason about than directly proving that the leakage does not hurt security.

A possible comparison is to the definition of security of encryption. The “moral equivalent” of this condition is the notion of indistinguishability of encryptions, which is, generally speaking, easier to prove than semantic security. But we know that the two notions are equivalent in that context.

Note that the notion of key-oblivious is relevant to any type of leakage, not necessarily timing, but in case of time we have various properties that make it particularly useful, e.g, the leakage of applying a function f and then g is, under reasonable assumptions, the sum of the two leakages.

1.4 Our Contributions and Technical Overview

Our goal in this work is to investigate the landscape of algorithms and systems that can be implemented in a manner resistant to timing attacks, but we wish to expand the “Procrustean bed” of fixed-time algorithms. We provide foundational treatment to the subject as well as many algorithms and separation results.

We propose several criteria for expressing the property that the timing information of an implementation of an algorithm does not expose sensitive information in the context of keyed functions. The most interesting one is *key-oblivious* (Definition. 1), which means that a polynomially bounded adversary cannot tell whether it received the timing of the actual key or of a random unrelated key. Namely, suppose that $\mathcal{F}_k$ is a keyed function with a key k and $\mathcal{T}(\mathcal{F}_k(q))$ is the time takes to execute $\mathcal{F}_k$ on the query q, then the key oblivious definition means that a PPT adversary cannot distinguish the following two cases: whether the time it gets is the real running time on the actual key k, or whether the running time is on an unrelated key k':

Definition 1. We say a keyed function $\mathcal{F}$ is **key-oblivious** secure against timing attacks if any probabilistic polynomial-time (PPT) adversary Adv has a winning probability at most $\frac{1}{2} + \mathsf{negl}(n)$ in the following game:

1. Two keys are sampled k_0, k_1.
2. A random bit $b \in \{0, 1\}$ is sampled.
3. The adversary Adv makes $\ell = poly(n)$ adaptive queries $q_1, \ldots, q_\ell$ to $\mathcal{F}_{k_0}$, and gets $\mathcal{F}_{k_0}(q_1), \ldots, \mathcal{F}_{k_0}(q_\ell)$, as well as $\mathcal{T}(\mathcal{F}_{k_b}(q_1)), \ldots, \mathcal{T}(\mathcal{F}_{k_b}(q_\ell))$.
4. The adversary outputs b', the guess of b, and wins if $b' = b$.

How useful is this criterion? What does it imply? The notion of key obliviousness is most useful in cases where the period where the adversary has access to the timing information is separated from when it actually attacks; for instance, in the case of signatures schemes, where the adversary may know how long it takes to produce a signature on a message, but where the adversary does not have access to the timing information of the signing of the actual message it wants to forge. We then prove that if we have signature scheme $\mathcal{F}_k$ that is existentially unforgeable secure against an adaptive chosen message attack and the signature function $\mathcal{F}_k$ is key-oblivious, then even if the adversary in the forgery game has access to the running time it takes to generate the signatures, then this adversary will not manage to forge a valid-looking signature on any message it was not given a signature explicitly.

As mentioned above, the key-oblivious criteria is most relevant when the attack occurs after timing information is not available anymore, e.g. as in the case of signature schemes. But there are scenarios where this is not the case and the adversary does get timing information during a "challenge phase." Consider, for instance, the case of encryption, where the final goal of the adversary is to distinguish between the encryption of two messages. The game has a "challenge phase" in which the adversary sends two messages and receives an encryption of one of them and its goal is to guess which one it is. The encryption may not be time-secure, even though the encryption implementation is key-oblivious.

To see this, consider the following example: suppose that the running time depends only on the least significant bit (lsb) of the message and does not depend on the key or other bits of the message. Then given two messages with different lsbs, the adversary who gets the running time of the actual message that was chosen, can easily distinguish whether the encryption was of one message or the other.

A case where this may be significant is in voting machines where votes are encrypted and then shuffled. If the timing of an encryption of a particular vote is known, then if the encryption is not *query-oblivious* in the above sense then this yields information about the actual vote.

To guarantee time-security also in cryptographic games that have a "challenge phase" (as in the indistinguishability game) we propose another security criteria called *query-oblivious* (Definition 3) whose aim is to capture the property that the time to evaluate a query does not leak information about the query itself. We then prove specifically for indistinguishability of encryptions (Theorem 26) that if the implementation is query-oblivious, then it is time-secure.

A fundamental issue concerning any new security definition is what happens when a primitive satisfying it is part of a larger structure and whether the new criterion is preserved under different constructions. To this end, we investigate different constructions and explore whether they preserve key-obliviousness and whether being query-oblivious as well is necessary for them to preserve key-obliviousness. We focus on the following constructions:

- The famed Goldreich-Goldwasser-Micali (GGM) construction: we show that if G is a PRG implemented in a key-oblivious manner, then applying GGM with G yields a PRF that is key oblivious.

- The cycle walking technique for format-preserving encryption: we show that if the permutation π is key oblivious, then the result π' is key oblivious.
- Domain extensions of PRFs: we show that even if the underlying PRFs are key-oblivious, then the classical results do not necessarily imply that the result is key-oblivious. But we show that the *cascading construction of PRF extension preserves key obliviousness.*

Main Application

In Section 4 we turn our attention to pseudorandom permutations (PRP) on small domains (related to format-preserving encryption). The most efficient construction for small domain PRP is the "Sometimes Recurse" (SR) shuffle by Morris and Rogaway [28], which runs in *expected* time of $O(\log N)$ and is secure even when the adversary queries the whole domain (a more detailed exposition appears in Section 4). The downside of SR is that its running time is fully determined by the number of leading 1's in the output. This makes the SR construction not secure against timing attacks, namely, SR is neither key-oblivious (Claim 12) nor query-oblivious (Claim 14).

We suggest a new construction of a PRP on small domains which we call the *Janus Sometimes Recurse* (JSR) *that is not constant time, yet provably secure against timing attacks. Our construction is faster than all previously known constructions that are secure against timing attacks.* Specifically we prove that: (1) JSR is key-oblivious (Claim 16), i.e., a PPT adversary cannot distinguish between the key that was used and a random key even when the adversary gets the exact running time of the PRP; and (2) JSR is also *query-oblivious* (Claim 18), i.e., a PPT adversary cannot infer from the computation time of the PRP on a query q what q is.

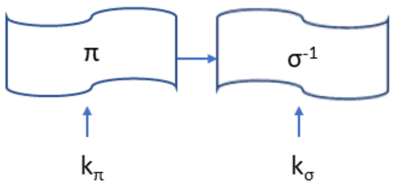


Figure 1 JSR construction on two PRPs.

Generally speaking, JSR takes two independent (i.e. with two independent keys) copies of SR on the same domain $[N]$, where the permutations are denoted by π and σ and the keys by k^{π} and k^{σ}, and composes π with σ^{-1}, see Figure 1. This is similar to the approach that Maurer and Pietrzak [26] used to move from non-adaptive to adaptive PRPs. The term "Janus" in the name of our construction "Janus Sometimes Recurse" (JSR) comes from the Roman god who was depicted as having two faces, since both the directions (encryption and decryption) are forward-looking.

The intuition for this construction is that while the running time of the forward direction leaks information about the output, the running time of the inverse is *determined by the input*, and so by composing the two we get that the running time of the algorithm both in the forward direction and in the inverse, is determined by *the inner value* which is *almost* independent of the input and output since π, σ are PRPs.

Main Security Claim

To formally define the criteria in general, we consider a *cryptographic game* (Definition 20) for a keyed function, which captures the security of many primitives including indistinguishability of encryptions, digital signatures, and pseudo-random permutations (we denote for function

$\mathcal{F}$ the security game with $G_{\mathcal{F}}$). A game $G_{\mathcal{F}}$ has *noticeable security* (Definition 20), if it is defined between an adversary and a principal, and determining who wins the game can be done without direct access to the key, but simply based on the queries and the state of the principal. We show that digital signatures, pseudo-random permutations, and encryption (but for indistinguishability of encryption see caveat below and Section 5.2) have games with noticeable security.

The main result (Theorem 1) shows that: For *a keyed function $\mathcal{F}_k$ that is secure w.r.t. a game $G_{\mathcal{F}}$ that has "noticeable security," if the implementation of $\mathcal{F}_k$ is key-oblivious then $\mathcal{F}_k$ is time-secure, that is, $\mathcal{F}_k$ is secure w.r.t. $G_{\mathcal{F}}$ even when the exact running timing of executing the oracle on the queries is leaked to the adversary.*

2 Keyed Functions Secure Against Timing Attacks

We now aim to formalize security against timing attacks for keyed cryptographic functions. Since the security of keyed cryptographic functions is usually measured by the success of a PPT adversary in some game, we would like the security notion to provide the keyed function security against such adversaries.

Let $\mathcal{F}$ be a keyed function, with key space $\{\mathcal{K}_n\}_n$, where keys are sampled from $\mathcal{K}_n$ have length $poly(n)$. Denote by $\mathcal{F}_k$ the keyed function with a chosen key k. To ease the notation we will use $k \sim \mathcal{K}$ to denote sampling k from $\mathcal{K}_n$ when n is understood from context.

We denote by q a query to the function. We do not state what type of query it is, since different types of functions will have different queries. For example, if $\mathcal{F}$ is an encryption scheme, then it makes sense to allow encryption queries as well as decryption queries. Denote by $\mathcal{F}_k(q)$ the answer to the query and by $\mathcal{T}(\mathcal{F}_k(q))$ *the running time* it took for the answer to return.

Assumption on Running Time Running time is implementation dependent, hence we stress that $\mathcal{T}(\mathcal{F}_k(q))$ depends on $\mathcal{F}, q, k$ as well as the implementation of $\mathcal{F}$ in the computational model. In many cases, once a key is fixed, the running time on a query will be deterministic, but there are cases in which the running time might be a distribution even with the same key and query. We therefore think of $\mathcal{T}$ as a distribution (which may be a distribution supported on one element).

The crucial assumption which is *running time specific* that we assume is *linearity of composition*, meaning that $\mathcal{T}(\mathcal{F} \circ \mathcal{G}(x)) = \mathcal{T}(\mathcal{F}(\mathcal{G}(x))) + \mathcal{T}(\mathcal{G}(x))$ where by $\mathcal{F} \circ \mathcal{G}$ means running $\mathcal{G}$ on x, and sequentially $\mathcal{F}$ on $G(x)$. This assumption is not used in this section, as well as Section 5, since in these sections we study a single function. On the other hand linearity of composition is a key for designing complex cryptographic primitives from basic ones, e.g. in the JSR construction appearing in Section 4.

We, therefore, require that for some of the statements in the paper *at certain points of the computation*, the model is inherently sequential and that many optimizations incorporated by modern computers to speed running time (e.g. pipeline, multi-processing, branch prediction, etc.) are not allowed at those points (hence the linearity assumption). As is well known, such optimizations can be exploited, with Spectre being one of the notable examples.

We use the notation negl for any function $\mathsf{negl}\colon \mathbb{N} \to \mathbb{R}^+$ satisfying that for every positive polynomial $p(\cdot)$ there is an N such that for all integers $n > N$ it holds that $\mathsf{negl}(n) < \frac{1}{p(n)}$. Such functions are called *negligible*. We will also call a random variable with distribution $Bernoulli(\frac{1}{2})$ a random bit.

We state three definitions of security of keyed cryptographic function. The first two definitions concern securing the key from a timing attack, while the third is designed to secure the result of queries.

▶ **Definition 1.** *We say a keyed function $\mathcal{F}$ is* **key-oblivious** *secure against timing attacks if any probabilistic polynomial-time (PPT) adversary* Adv *has a winning probability at most* $\frac{1}{2} + \mathsf{negl}(n)$ *in the following game:*

1. *Two keys are sampled $k_0, k_1 \sim \mathcal{K}$.*
2. *A random bit $b \in \{0, 1\}$ is sampled.*
3. *The adversary* Adv *makes $\ell = poly(n)$ adaptive queries $q_1, \ldots, q_\ell$ to $\mathcal{F}_{k_0}$, and gets $\mathcal{F}_{k_0}(q_1), \ldots, \mathcal{F}_{k_0}(q_\ell)$, as well as $\mathcal{T}(\mathcal{F}_{k_b}(q_1)), \ldots, \mathcal{T}(\mathcal{F}_{k_b}(q_\ell))$.*
4. *The adversary outputs b', the guess of b, and wins if $b' = b$.*

Definition 1 means that the joint distribution of running times on a polynomial number of queries for two keys $\mathcal{T}(\mathcal{F}_{k_b}(q_1)), \ldots, \mathcal{T}(\mathcal{F}_{k_b}(q_\ell))$ are indistinguishable by a PPT adversary even when it sees the results of the query on a specific key. We now strengthen Definition 1:

▶ **Definition 2.** *We say a keyed function $\mathcal{F}$ is* **key-switch** *secure against timing attacks if any PPT adversary* Adv *has a winning probability at most* $\frac{1}{2} + \mathsf{negl}(n)$ *in the following game:*

1. *Two keys are sampled $k_0, k_1 \sim \mathcal{K}$.*
2. *The adversary* Adv *makes $\ell = poly(n)$ many queries $q_1, \ldots, q_\ell$ and gets $\mathcal{F}_{k_0}(q_1), \ldots, \mathcal{F}_{k_0}(q_\ell)$ as well as $\mathcal{T}(\mathcal{F}_{k_0}(q_1)), \ldots, \mathcal{T}(\mathcal{F}_{k_0}(q_\ell))$.*
3. *A random bit $b \in \{0, 1\}$ is sampled.*
4. *The adversary* Adv *makes another $\ell' = poly(n)$ many queries $p_1, \ldots, p_{\ell'}$ and gets $\mathcal{F}_{k_0}(p_1), \ldots, \mathcal{F}_{k_0}(p_{\ell'})$, as well as $\mathcal{T}(\mathcal{F}_{k_b}(p_1)), \ldots, \mathcal{T}(\mathcal{F}_{k_b}(p_{\ell'}))$.*
5. *The adversary outputs b', the guess of b, and wins if $b' = b$.*

Notice that if we skip step 2 we get back to Definition 1. The difference here is that the adversary gets the running time of the function with the same key as the answers, until a bit is chosen, and only then the adversary does not know if the timing comes from the same key or a different key. This essentially means that the distribution of the running times $\mathcal{T}(\mathcal{F}_{k_b}(p_1)), \ldots, \mathcal{T}(\mathcal{F}_{k_b}(p_{\ell'}))$ conditioning on the answers to the queries and the running time on the original key are indistinguishable by a PPT adversary.

This definition could be useful to prevent denial-of-service attacks: if the adversary can find expensive (time-wise) queries, then it can bunch expensive queries together and ask them so as to overload the system. If, in addition, the system has the property that a priory it is not clear how long a query would take, then it is not possible to find expensive queries (since then it would be possible to figure out whether a key-switch occurred or not).

The third definition is of *query-obliviousness* and involves only a single key and aims to express that the actual queries made are secure from a timing attack. This is desirable, for instance, in voting systems. Consider a voting system that uses a keyed function (e.g. a PRP) as a subroutine to the actual vote cast. The sensitive information that needs to be protected is the votes themselves and not necessarily the key.

▶ **Definition 3.** *We say a keyed function $\mathcal{F}$ is* **query-oblivious** *secure against timing attacks if any PPT adversary* Adv *has a winning probability at most* $\frac{1}{2} + \mathsf{negl}(n)$ *in the following game:*

1. *A single key $k \sim \mathcal{K}$ is sampled.*
2. *The adversary* Adv *makes $\ell = poly(n)$ adaptive queries $q_1, \ldots, q_\ell$ and gets $\mathcal{F}_k(q_1), \ldots, \mathcal{F}_k(q_\ell)$ and $\mathcal{T}(\mathcal{F}_k(q_1)), \ldots, \mathcal{T}(\mathcal{F}_k(q_\ell))$ of all the queries.*
3. *The adversary* Adv *chooses two new distinct queries $q'_0 \neq q'_1$ such that $q'_0, q'_1 \notin \{q_1, \ldots, q_\ell\}$.*

4. *A bit $b \in \{0,1\}$ is chosen at random.*
5. *The adversary* Adv *gets $\mathcal{T}(\mathcal{F}_k(q'_b))$.*
6. *The adversary outputs b', the guess of b, and wins if $b' = b$.*

In the game above, the adversary makes queries and gets their running time. Then the adversary chooses two different queries and gets the running time of one of them. The challenge is to decide what is the query whose running time was returned. There are two (non-mutually exclusive) variations on this:

Weakly vs. Strongly: If the function is also secure for general queries q'_0, q'_1 that are *not necessarily new*, then we say that function is *strongly query-oblivious*, while the definition above is *weakly query-oblivious.*

With vs. Without Results We define a variant of query-oblivious, we call **query-with-results-oblivious** which is the same as the above query-oblivious game but with one change: in step (5) the adversary Adv gets both $\mathcal{F}_k(q'_b)$ and $\mathcal{T}(\mathcal{F}_k(q'_b))$ rather than only $\mathcal{T}(\mathcal{F}_k(q'_b))$. As we shall see, this query-with-result-oblivious will be useful, for example, for time-security of indistinguishability of encryption, while the original query-oblivious definition will be useful, for example, for domain extension.

▶ Remark 4. The definitions above are not equivalent to one another. The argument appears in the extended version of this paper.

3 Constructions Preserving Key-Obliviousness

A natural question about the notion of key-oblivious is whether the key-oblivious is preserved when applying it to several functions. In this section we investigate several well known constructions and check whether the key-obliviousness is preserved under these constructions, given that the underlying building blocks are key-oblivious. We start with the basic constructions of composition and concatenation. We then consider the fundamental cryptographic constructions:

1. The GGM construction of pseudorandom functions, where we show that if the basic building block, the PRG G, is key oblivious, then the result is key oblivious (Claim 5). However it is not necessarily query-oblivious (Claim 6).
2. We consider the Cycle walking technique for constructing format-preserving encryption, which is not fixed time by nature, yet we show that if the underlying permutation π is key oblivious, then the result is key oblivious.
3. Finally we consider various *domain extension techniques* and show that while some of them do not preserve key-obliviousness (e.g. using the Levin trick) it is possible to get key-obliviousness using either an additional primitive such as UOWHF or using the cascading consturction.

Composition and concatenation

A basic issue when considering security definition is how they interact as part of a larger system. The good news is that wrt concatenation key-obliviousness is preserved: Suppose that we have two keyed functions f and g and suppose that their keys are independent. Then the natural implementation of producing $f(x) \circ g(x)$, first compute $f(x)$ and then $g(x)$ is also key oblivious. Also suppose that at each step either f or g are called, then the whole process is still key-oblivious. Furthermore, let h be any function that is implemented in constant time (e.g. addition or Xor). Then the natural implementation of $h(f(x), g(x))$ where f and g are key-oblivious and are computed in a sequential manner and where h is fixed time is itself key oblivious.

On the other hand, as we shall see, for composition the case is different: even if f and g are key-oblivious it is not necessarily true that $f(g(x))$ is key oblivious! This is shown in Claim 8.

3.1 The GGM Construction of PRFs

Consider the Goldreich-Goldwasser-Micali (GGM) construction of pseudorandom functions (PRF) from pseudorandom generators [16]. The construction starts with a pseudorandom generator $G\colon \{0,1\}^n \mapsto \{0,1\}^{2n}$ and the PRF $\mathcal{F}_k\colon \{0,1\}^n \mapsto \{0,1\}^n$ with key $k \in \{0,1\}^n$ is defined by imagining a full binary tree of depth n where each node gets an n bit label. The root is labeled with the key k and each parent induces a labeling of its two children by applying G to its label; the left half of the result becomes the label of the left child and the right half the label of the right child. The value of $\mathcal{F}_k(x)$ for $x \in \{0,1\}^n$ is the label of the leaf at the end of the path defined by x.

Suppose that G is implemented in a key oblivious manner, meaning in this case, simply that given $G(k)$ for a uniform $k \in \{0,1\}^n$ and $\mathcal{T}(G(k))$ or $\mathcal{T}(G(k'))$ for a uniform $k' \in \{0,1\}^n$, it is hard for a poly-time adversary to distinguish between the two cases. Now consider the straightforward implementation of the GGM PRF $\mathcal{F}_k(x)$ from G, which consists of n applications of G given k and x (developing the required labels). Assume that taking the left or right half of the output of G once it is computed is fixed-time and that each application of G starts from scratch. Is the result key oblivious? Is it query-oblivious?

▷ Claim 5. The key-obliviousness of G together with the requirement that it is a PRG imply that the GGM construction is key-oblivious.

Proof. One possible way to prove the claim is to follow the same lines as the classical proof of pseudo-randomness of the GGM construction. This proof is based on a hybrid argument. If it is possible to distinguish between the construction and a truly random function using m queries, then there is a sequence of $m' \leq m \cdot n$ distributions, the first being the pseudorandom one, as described above and the last one being the truly random one[2]. An alternative approach is to use induction on the depth on the tree. For the base case $n = 1$, the property that G is assumed to have is sufficient to guarantee key obliviousness. To increase the number of levels, we will think of the two branches from the root as two independent functions. In this case, the whole process should still be key oblivious, as the discussion at the beginning of Section 3 shows. If the actual implementation is no key-oblivious, then again, we have an attack of the key obliviousness of G. ◁

▷ Claim 6. The GGM construction is *not* query-oblivious, at least not if G is not fixed-time computable. This is true even if we consider weakly query-without-result-oblivious.

Proof. Note that the time to compute $\mathcal{F}_k(x)$ is the sum of n applications of G on random-looking inputs. Furthermore, the timings of $\mathcal{F}_k(x)$ and $\mathcal{F}_k(x')$ for x and x' that differ only in the last bit are closely correlated (since the sums are over the same summands, except the last one), compared to the timing of x and x'' where x'' is, say, a random input. In the latter, there will be little correlation. So Definition 3 is not satisfied. ◁

[2] In our case, for the ith distribution, the first $i-1$ timings are the application of G on one key (the real one, for which the output values are also given) and the last $m-i+1$ ones are the timings of another key, unrelated to the real one. If it is possible to distinguish between the first and last distributions, then it is possible to distinguish between some two neighboring distributions.

3.2 Format Preserving Encryption - the Cycle Walking Technique

A good example of a construction that is inherently non fixed-time, yet key oblivious and under some conditions query-oblivious, is the cycle walking technique of Format Preserving Encryption (See Bellare et al. [6]). Imagine that we have a construction of a pseudo-random permutation (PRP) on some domain, say of size 2^ℓ, and we want to build from it a PRP on a smaller domain $S \subset [2^\ell]$. A simple example is when S is the set of numbers 0 through $Q-1$ (where Q is not a power of 2), but the question is relevant for any format of S.

What Black and Rogaway [9] analyzed is a simple cycle walking technique for constructing permutations on smaller domains. Given a PRP π on the larger domain of size 2^ℓ, to get a PRP on S define π' by starting with $x \in S$ and repeatedly apply π to it until hitting a value in S. This is defined to be $\pi'(x)$. The expected number of applications of π is $2^\ell/|S|$. It is clear that this construction is *not* fixed-time (if $|S| < 2^\ell$), even if the original PRP is fixed-time. So which of the definitions of our framework does this technique satisfy? We note that Bellare et al. wrote that it "Doesn't Give Rise to Timing Attacks,"[3] but we want to check in what sense this is true and how it fits our definitions.

We claim that the exact properties of π' depend on whether the original PRP is fixed-time or "merely" key oblivious. For the former we get both key and query-obliviousness and for the latter just key obliviousness. We assume that the implementation is such that the timing of the various calls to π are independent of each other, in the sense the time it takes to run $\pi(x)$ does not depend on any sequence of operations done before and in particular on whether we have just executed $\pi(x_1), \pi(x_2), \ldots, \pi(x_m)$.

▷ Claim 7. (i) If the permutation π is fixed-time, then the result π' is both key oblivious and query-oblivious in the weakly with results sense. (ii) If the permutation π is key oblivious, then the result π' is key oblivious.

In order to see that (ii) is correct, think of simulating π' through access to π. Now if there is an attack on π' that distinguishes between true timing and timing of some random unrelated key, then we can apply it to π itself and get the same distinguishing probability, thus violating the assumption that π is key oblivious.

In order to see that (i) is true, think of composing the permutation π with a random permutation σ. By the definition of a PRP, this is indistinguishable from the plain π (since this is the case for a truly random permutation). Now instead of a query x, the query is effectively $\sigma(x)$, which makes it into a random unknown value. Given that the implementation is of π is fixed-time, the only information gained from the timing is the number of applications of π needed to evaluate $\pi'(x)$. But given the randomness of σ this is useless information to distinguish between two queries q'_0 and q'_1 that have not appeared so far and therefore the construction is query-oblivious.

Note that the construction does not satisfy the key switch requirement of Definition 2, since once π is fixed, then each element x has a very distinctive number of evaluations of π needed to compute $\pi'(x)$. Therefore it is possible to see if the time to compute $\pi'(x)$ changes at the potential key switch time.

[3] What they proved is that leaking the number of applications of π does not hurt the pseudorandomness of π'

3.3 Key-oblivious Domain Extension

Pseudo-random functions are a major cryptographic primitive that can be used to efficiently obtain many other primitives and is very useful in many protocols. One question that comes up is: given a function family F where each function $\mathcal{F}_k \in F$ is, say, length preserving, i.e. $\mathcal{F}_k \colon \{0,1\}^n \mapsto \{0,1\}^n$, how does one come up with constructions which are on larger domains, e.g. $\{0,1\}^{2n} \mapsto \{0,1\}^n$.

The Levin Trick

A common and simple way of obtaining domain extension is to apply a universal hash function from the larger domain to the smaller one and then apply the PRF. Namely, let Γ be family of pair-wise independent hash functions s.t. $g \colon \{0,1\}^{2n} \mapsto \{0,1\}^n$ for $g \in \Gamma$ and $\mathcal{F}_k$ be a PRF. Then the extended function is defined as $\mathcal{F}'_{k,g}(x_1, x_2) = \mathcal{F}_k(g(x_1, x_2))$.

▷ Claim 8. Even if $\mathcal{F}_k$ is key-oblivious and even if g is fixed time, the resulting construction may not be key-oblivious.

Proof. Consider the case that $\mathcal{F}_k$ is very much *not* query oblivious. That is, the time to compute $\mathcal{F}_k(x)$ is x. Now suppose that h is defined by two values $a_1, a_2 \in GF[2^n]$ (chosen uniformly at random) and the function is $g(x_1, x_2) = a_1x_1 + a_2x_2$ where the computation is over $GF[2^n]$. Given a few examples of pairs (x_1^i, x_2^i) and the corresponding values $\mathcal{F}'_{k,g}(x_1^i, x_2^i)$ it is possible to reconstruct a_1 and a_2. Once this is done it is possible to find collisions with g, i.e. two pairs (x_1, x_2) and (x'_1, x'_2) s.t. $g(x_1, x_2) = g(x'_1, x'_2)$. For this pair we will have that $\mathcal{F}'_{k,g}(x_1, x_2) = \mathcal{F}'_{k,g}(x'_1, x'_2)$. Now if the timing of a random key is given, then this will not cause a collision in $\mathcal{F}'_{k,g}$ and therefore there is a way to distinguish correct and incorrect timings. ◁

Note that this claim shows that key-obliviousness is not necessarily preserved under composition. In this example both $\mathcal{F}_k$ and g are key-oblivious, yet the composition is not.

▶ **Corollary 9.** *There are keyed functions f and g such that given key-oblivious implementations of f and g the resulting composition $f(g(x))$ is not key oblvious.*

Using CRH and UOWHF

A way to remedy this problem is to use a Collision Resistant Hash (CRH) function, instead of a combinatorial one used in the original proposal by Levin. Recall that a function h is a CRH if it is hard to find two different values x and x' that collide under h, that is, any collision is considered a violation of the hardness assumption. But this approach (i) Requires another cryptographic assumption or primitive. Recall that in terms of assumptions, PRFs can be built from one-way functions in a black-box (BB) manner, whereas CRHs are BB separated from one-way functions. (ii) May not work with all range of the parameters we are interested in, since a CRH requires a minimum range size. For instance, the range of the CRH cannot be 80 bits.

The perhaps surprising observation is that we show that the CRH in the above construction can be replaced with a Universal One-way Hash Function (UOWHF) [31]. UOWHF (or second pre-image resistant hash function), are ones where the target x is chosen before the function is known and the collision should be with the target x. UOWHF can be based on one-way functions [33, 22]. The idea is to replace each x_i with its PRF value. Let $h \in H$ where H is a UOWHF family. Consider the construction

$$\mathcal{F}'_{k,k',h}(x_1, x_2) = \mathcal{F}_k(h(\mathcal{F}_{k'}(x_1), \mathcal{F}_{k'}(x_2))).$$

▷ Claim 10. If the implementation of $\mathcal{F}$ is key-oblivious, then for *any* implementation of the hash function h chosen from a family of UOWHFs, the implementation of $\mathcal{F}'$ as defined above is key-oblivious.

The only case we need to worry is if the adversary finds collisions under h. The hardness properties of h do not guarantee hardness of finding *any arbitrary* collision, but rather with one target specified in advance. The idea is that an adversary may ask various queries, and since we do *not* assume that $\mathcal{F}_k$ is query oblivious and we make no timing assumption regarding h, then the values of $\mathcal{F}_{k'}(x_1)$ are known to the adversary (i.e. they may leak through the computation of h).

Furthermore, the adversary can mix and match an x_1^i and x_2^j from different queries. Nevertheless, the values $\mathcal{F}_{k'}(x_b^i)$ look random to the adversary. Suppose an adversary makes m queries to the function $\mathcal{F}_{k'}$. At the end of the attack it finds a collision with some pair (x_1^i, x_2^j) as one of the inputs with a collision in h (if the colliding pair was never queried this is even better). Then we can use this adversary as a second pre-image finder: We select a random value $(y_1, y_2) \in \{0,1\}^{2n}$ as the target and then guess i and j and when x_1^i is given, we plug in y_1 as the value of $\mathcal{F}_{k'}(x_1^i)$ and similarly for x_1^j we give y_2 as the value of $\mathcal{F}_{k'}(x_2^i)$. From the adversary's point of view this looks like a "normal" instance. Therefore the probability of selecting i and j correctly is $\Omega(1/m^2)$ times the probability that the adversary finds a collision.

Cascading

We show that the *cascading domain extension*, as analyzed by Bellare, Canetti and Krawczyk [5] actually preserves key-obliviousness. The length-doubling construction is

$$F'_k(x_1, x_2) = F_{F_k(x_1)}(x_2).$$

The straightforward implementation of F' is to first compute $F_k(x_1)$ then take the resulting value as the key k_2 to $F_{k_2}(x_2)$ where the time of the second is independent of the time is took compute the first one.

▷ Claim 11. If the implementation of F is key-oblivious, then the "straightforward" implementation of F' is key-oblivious.

Proof (Sketch). The main issue is that in this construction it is always clear what the query is. The keys, that keep changing, on the other hand, are not known. Therefore the Bellare et al. proof can be translated to this setting. Consider the experiment where instead of using the value $F_k(x_1^j)$ and random value is v_j is used for the next step (while making sure to use it consistently) but the timing produced is that of $\mathcal{T}(F_k(x_1^j))$ and $\mathcal{T}(F_{v_j}(x_2^j))$. If it is possible to distinguish between these two cases, then there is an attack on the key obliviousness of F. If not, then note that we can view the new construction as a concatenation of many functions, as argued at the beginning of this section. ◁

▶ **Question.** *A major issue we did not resolve is how* not to lose the birthday bound*, as was done in Berman et al. [7] for domain extension without timing. Is it possible to get a similar result when timing is leaked?*

4 Key-Oblivious PRPs on Small Domains

The question of how to generate Pseudo Random Permutations (PRP) has been extensively investigated for a few decades. Luby and Rackoff [25] (who defined the notion of PRP) showed how to get a PRP from a PRF. Their construction uses a Feistel network and there are many variants of it. In their work the security of the construction works provided the adversary makes at most $O(N^{\frac{1}{4}})$ queries, where N is the size of the domain. When the domain is not very big, such a security guarantee might not be enough, as $N^{\frac{1}{4}}$ can be a feasible amount of queries made. This is also true to refinements of the method, such as Naor and Reingold [29], who get to $N^{\frac{1}{2}}$. In such cases we might want the security to hold even if the adversary queries a constant fraction of the domain or even all of the domain but a constant number of elements.

When the domain is very small, it is possible to generate a fully random permutation, rather than a PRP. Yet this is undesirable in many cases, since the memory required to represent a random permutation is $\Omega(N \log N)$ bits. This memory requirement may be feasible for small enough N but is infeasible for medium-sized N (e.g. all credit card numbers or SSNs). We are left with the intermediate case of small, but not too small, domains, so that explicitly saving a permutation is infeasible, yet the security guarantees of Feistel network constructions might not suffice. As a concrete example think of the domain of credit card numbers (16 decimal digits). We refer to PRPs of this intermediate case as small-domain PRPs.

Small-domain PRPs are useful in a variety of application scenarios, e.g. cryptographic constructions, as Oblivious RAMs [17, 35], for randomly reordering (permuting) a list of items. They can also be used to generate pseudorandom *unique* tokens (e.g., product serial numbers) in a specific format and to encrypt data in a small domain, such as encrypting a 9-digit social security number into another 9-digit number. Because of this, a small-domain PRP is also commonly referred to as a small-domain cipher or format-preserving encryption (FPE).[4] FPE has been a useful tool in encrypting financial and personal identification information, and transparently encrypting information in legacy databases.

In this section we focus on the small domain and show an interesting construction of PRPs on which is not fixed-time, yet is also secure under some of our definitions. A line of three works addressed this issue and showed efficient constructions for PRPs on small domains N with strong security guarantees, based on PRP or PRFs on large domains. The key observation in these works is one made by Moni Naor (see [34]), that if a card shuffling algorithm is *oblivious*, meaning that one can trace the trajectory of a card without attending to a lot of other cards in the deck, then it gives rise to a computationally feasible PRP. Therefore we can think of $[N]$ as a deck of cards of size N. All three works we describe start from this viewpoint on PRPs. The dominant computational resource in these works is the calls to a PRF on a large domain.

The first work, called “Swap-or-Not Shuffle” (SN) by Hoang, Morris and Rogaway [19] consists of a sequence of rounds that gradually shuffle the deck. In each round they consider a random matching of the cards in the deck that matches card X with card $X \oplus K$ (the randomness is over the choice of K). Additionally, for each matched pair $X, X \oplus K$ there is a random and independent bit b that decides whether to swap the matched pair of cards or not (these bits are also derived from the key). Hoang et al. [19] proved that applying the swap-or-not procedure $O(\log N)$ times and picking the matching index K for each round at random and independently suffices as long as at most $(1 - \varepsilon)N$ queries were made for any fixed $\varepsilon > 0$ (notice that this ε affects multiplicatively the number of rounds of swap-or-not needed to achieve a PRP).

[4] Note that the cycle walking technique mentioned in Section 3 does not solve the problem of constructing a small or medium size PRPs, since it needs a PRP of not much larger size to begin with.

To implement this as a PRP the swap bits along the shuffle b, as well as the K of the matching should be produced by a PRF (which can be derived from a PRP on *large* domain such as AES). This gives a PRP that runs in a fixed time and remains secure as long as at most $(1-\varepsilon)N$ queries were made for any fixed $\varepsilon > 0$ (notice that this ε affects by a multiplicative the number of rounds of swap-or-not needed to achieve a PRP). This procedure is fixed-time provided that the PRFs and XOR operations are implemented in running time independent on the inputs and keys.

The second work, called "Mix and Cut" by Ristenpart and Yilek [32], aimed to improve on the number of queries that can be made while keeping the PRP secure. Ristenpart and Yilek introduced a construction of PRP which runs in a fixed-time of $O(\log^2 N)$ and achieves *full security*, meaning it remains secure even if the entire domain is queried. The Mix and Cut shuffle works by mixing the deck, cutting it to two equal parts, and mixing each of them recursively using the Mix and Cut shuffle. In the paper, they also proved that if the shuffle before each cut mixes the cards well enough (which means the top half and bottom half is approximately a random partition of the cards), then this procedure achieves full security. They explicitly showed that the Swap-or-Not shuffle can be used with the ε from last paragraph being $\frac{1}{2}$ to give a fully secure PRP with the Mix and Cut construction.

The third work, called "Sometimes Recurse" (SR) shuffle by Morris and Rogaway [28], constructed a PRP which runs in *expected* time of $O(\log N)$ and achieves *full security.* Morris and Rogaway observed that when the Mix and Cut procedure is called, there is no need to mix both the top half and the bottom half of the deck. Since the top half looks almost uniform, it is enough to mix only the bottom half, therefore they suggested to only recurse on the bottom half of the deck, hence the name sometimes recurse. This construction allows an improvement on the $O(\log^2 N)$ of the Mix and Cut shuffle to an expected number of rounds which is $O(\log N)$.

The downside of SR is that it is no longer fixed-time, and in fact the running time is fully determined by the number of leading 1s in the output. Morris and Rogaway address this issue by stating that in a very common use of SR an adversary sees the outputs anyway, and so the running time doesn't give more information. However it is also the case that keyed functions, and in particular PRPs, are employed as a *subroutine of a larger system* (see for example [3]). In such cases the adversary no longer sees the output, and so the running time might leak valuable information that can harm the security of the system.

Consider for example if the PRP is used for storing or transmitting some piece of sensitive information like a vote or a credit card number in a way that should not reveal the correspondence between the customer or voter and the ciphertext. In this case the adversary can only get the runtime of the transaction (purchase, vote) which corresponds to the runtime of the PRP but does not have direct access to the results of the queries. Suppose now that the adversary does get to see after some time a *batch of such ciphertexts* and perhaps some other information about them (say their opening in case of votes, or whether the transaction was declined for credit cards). If the adversary knows how long each transaction took, then it can connect the ciphertext and the voter or customer and learn something it should not have learned. Moreover, the SR scheme is vulnerable to a *denial-of-service attack* where the attacker can easily assemble *without any query* many different ciphertexts that take a long time to decrypt (by picking those that have long prefixes of '1's).

Before analyzing the security of SR we need to specify what we mean by runtime of SR. We assume that SR construction uses the Swap-or-Not (SN) PRF for each shuffle and that the Swap-or-Not is implemented in a fixed-time manner. I.e. for any $x \in \{0,1\}^{\log N}$ we have that the running time of $SN_k(x)$ is independent of x and k but *is dependent* on the input length $\log N$. We denote this fixed running time of the the Swap-or-Not on $\log N$ bit inputs by $\mathcal{T}_{\log N}(SN) := \mathcal{T}(SN_k(x))$ for any x, k. Consequently, by linearity of composition, the running time of SR on the input x is $\mathcal{T}(SR_k(x)) = \sum_{i=0}^{j} \mathcal{T}_{(\log N)-i}(SN)$ where j is the number of leading 1's in the output, i.e. we assume commands are executed one after the other on a single unit without branch predictions. We show that the SR construction is not secure with respect to our definitions in Section 2.

▷ Claim 12. The SR construction is *not* key-oblivious, i.e. does not satisfy Definition 1 and hence also is *not* key-switch oblivious, i.e. does not satisfy Definition 2.

Proof. Recall that by definition of SR, an adversary can determine the number of leading 1s in the output, from the running time and vice versa. This property of SR gives the following simple strategy:

1. Choose a single element x and query it to get: $SR_{k_0}(x)$ and $\mathcal{T}(SR_{k_b}(x))$.
2. Check the number of leading 1s in $SR_{k_0}(x)$, which is $\mathcal{T}(SR_{k_0}(x))$ and compare it to the running time $\mathcal{T}(SR_{k_b}(x))$ (number of calls to SN by SR on x).
3. If $\mathcal{T}(SR_{k_0}(x))$ is equal to $\mathcal{T}(SR_{k_b}(x))$, return 0, else return 1.

Observe that the distribution of the number of leading 1s is a $Geo(\frac{1}{2})$ distribution truncated at n. Also since the SR shuffle is a PRP, then with constant probability $\mathcal{T}(SR_{k_0}(x)) \neq \mathcal{T}(SR_{k_1}(x))$. This gives the adversary a constant advantage in the key-oblivious game (Definition 1), and therefore the SR PRP is not key-oblivious. Note that by choosing a polynomial number of inputs $x_1, \ldots, x_\ell$, the adversary can get a winning advantage that is exponentially close to 1. ◁

▶ **Corollary 13.** *The SR construction is not* $(G_{\mathsf{PRP}}, \frac{1}{2})$*-time-secure.*

Proof. This follows from Remark 25. ◀

▷ Claim 14. The SR construction is *not* secure with respect to query-obliviousness (Definition 3) even in the weakly without results sense.

Proof. Observe that in the definition there is no restriction on making queries in one direction, we assume both forward and inverse queries. By the construction of SR, in the inverse direction the running time is determined by the number of leading $1's$ *in the input.* This gives an adversary a very simple attack. First, make an inverse query on the string x which we define to be half $1's$ in the beginning and then half $0's$. Now for the challenge pick x_0 to be the all $0's$ string, and x_1 the all $1's$ string and make inverse queries. If the running time is faster than that on x, return $b = 0$; else return $b = 1$. This gives the adversary a winning probability of 1 in the game defined in Definition 3. ◁

4.1 JSR: Constructing key-oblivious and query-oblivious PRPs

So far we have seen fixed-time constructions and a non fixed-time construction which is not secure under our definitions. It begs the question: are there (interesting) constructions that are not fixed-time, yet are also secure under some of our definitions? We now show a construction of "SR with a twist", which: (1) achieves the same expected run time of $O(\log N)$ up to a multiplicative factor of 2, (2) is not fixed-time, and (3) is secure with respect to Definition 1 and with respect to Definition 3 (assuming we have a fixed-time implementation of a PRF).

The Proposed Construction: Janus Sometimes Recurse

The construction takes two independent (i.e. with two independent keys) copies of SR on the same domain $[N]$, where the permutations are denoted by π and σ and the keys by k^π and k^σ, and composes π with σ^{-1}, see Figure 2. This is similar to the approach that Maurer and Pietrzak [26] used to move from non-adaptive to adaptive PRPs. We call this construction "Janus Sometimes Recurse" (JSR):

Algorithm 1 $JSR(x)$ with keys k^π and k^σ.

```
1: return $\sigma^{-1}(\pi(x))$
```

The term "Janus" comes from the Roman god who was depicted as having two faces, since both the directions (encryption and decryption) are forward looking.

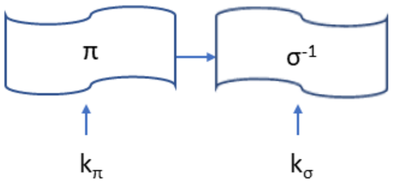


Figure 2 Janus construction on two PRPs.

The intuition for this construction is that while the running time of the forward direction leaks information about the output, the running time of the inverse is *determined by the input*, and so by composing the two we get that the running time of the algorithm both in the forward direction and in the inverse, is determined by *the inner value* which is *almost* independent of the input and output since π, σ are PRPs.

We now turn to show that (1) JSR is secure with respect to Definition 1 as well as Definition 3 when the two elements chosen are fresh (were not queried before), i.e., in the weakly sense, and (2) that it is not secure under Definition 2 even in the relaxed version when queries cannot repeat after the switch.

We start with a lemma that will help us prove the security of the construction.

▶ **Lemma 15.** *Let π and σ be two PRPs on the same domain D, secure under q queries, and let (k_0^π, k_1^π) and (k_0^σ, k_1^σ) be two pairs of keys for π and σ respectively, then any PPT adversary* Adv *has a negligible advantage over $1/2$ in the following game:*

1. *A random bit $b \in_R \{0,1\}$ is chosen.*
2. *The adversary can make at most $j \leq q$ queries $x_1, \ldots, x_j$ to the composition of π with σ^{-1}, either in the forward direction $\sigma^{-1}_{k_0^\sigma} \circ \pi_{k_0^\pi}(x_i)$ or in the inverse direction: $\pi^{-1}_{k_0^\pi} \circ \sigma_{k_0^\sigma}(x_i)$.*
3. *The adversary gets in addition to the result of each query, the inner result with respect to b. Namely in the forward direction* Adv *gets $\pi_{k_b^\pi}(x_i)$ and in the inverse direction $\sigma_{k_b^\sigma}(x_i)$.*
4. *The adversary should output b' such that $b' = b$.*

Proof. Consider the distribution where a truly random value that hasn't appeared in previous queries is given instead of the inner value either in the case $b = 0$ or $b = 1$. We claim that the truly random distribution is indistinguishable from the distribution of inner values in both cases of b. Observe that the distribution of a truly random value that hasn't appeared in previous queries is the same distribution of the inner value of a composition of two truly random permutations that agree on the values already seen. Since π and σ are both PRPs, it follows that this distribution is indistinguishable from the inner value distribution both in $b = 0$ and $b = 1$. By a hybrid argument we get that the inner value distribution with $b = 0$ is indistinguishable from the inner value distribution with $b = 1$ which completes the proof. ◀

▷ Claim 16. Assuming we have a fixed-time implementation of PRFs and the implementation of the JSR uses it, then the JSR construction is key-oblivious, i.e. secure with respect to Definition 1.

Proof. The proof follows from Lemma 15 by observing that twice the number of leading 1's in $\pi_k(x)$ is exactly the running time for computing $\sigma_k^{-1} \circ \pi_k(x)$. Therefore the adversary can determine the running time of $\mathcal{F}_k(x)$ from $\pi_k(x)$. This implies that the advantage of the adversary in the key-oblivious game is at most the advantage of an adversary in the game defined in Lemma 15. Since we know that the advantage of an adversary in Lemma 15 is negligible, we get that the construction is secure with respect to Definition 1. ◁

▷ Claim 17. The JSR construction is *not* key-switch secure. That is, it does not satisfy Definition 2 even in the relaxed version.

Proof. Since we are interested in PRPs on small domains, this means that the adversary can query a large fraction or even *entire domain* apart from a few elements. The main observation is that the $\sum_{x \in D} \mathcal{T}(SR_{k_0}(x))$ is a constant independent of k_0, where the sum is over the running time of the JSR algorithm.

The strategy for the adversary is to query the entire domain up to a single element l (last element). By the observation above, this gives the adversary full information about the timing distribution of the last element. Now the switch happens and the adversary makes one last query on the remaining element. If the running time the adversary receives is the running time it expected with respect to k_0, then it returns $b = 0$; otherwise, it returns $b = 1$. Since the running time of $\mathcal{T}(SR_{k_1}(l))$ is not constant (with non-negligible probability), then with non-negligible probability $\mathcal{T}(SR_{k_0}(l)) \neq \mathcal{T}(SR_{k_1}(l))$ and so this strategy gives the adversary a non-negligible probability of winning. ◁

Finally, we have:

▷ Claim 18. The JSR construction *is* query-oblivious secure, in the weakly and with results sense.

Proof. The proof is similar to the proof of Claim 16. By a similar argument as the one in the proof of Lemma 15, we know that given the inner result of either q'_0 or q'_1, the adversary cannot tell which query it came from. Since given the inner result, the adversary can get the running time of the query, we conclude that given the running time $\mathcal{T}(SR_k(q'_b))$, the adversary cannot guess b. ◁

We note that all claims above regarding the JSR construction hold to more general constructions. The crucial property of SR, from which the claims follow, is that *the running time of the algorithm was fully determined by the output.* We summarize:

▶ **Theorem 19.** *Let π_k be a PRP for which the running time to compute π_k on x is fully determined by x's image, i.e. $\mathcal{T}(\pi_k(x)) = f(\pi_k(x))$ for some function f. If the permutation π is secure under q queries, then the Janus PRP $\pi_{k_0}^{-1} \circ \pi_{k_1}(x)$ is a key-oblivious and query-oblivious PRP secure under q queries.*

5 Main Security Theorem, Noticeable Security and Games Definitions

5.1 Defining Noticeable Security and Cryptographic Games

Our goal is to prove that if the implementation of a keyed function is key-oblivious then the keyed function is time-secure, that is, the keyed function is secure w.r.t. the cryptographic game in which the adversary has access not only to the relevant oracles but also to the time it takes the oracle to execute. We show this for cryptographic keyed functions that have what we call **Noticeable Security.** Many well-known and useful keyed functions primitives have noticeable security, such as: (i) digital signatures, (ii) pseudo-random permutations (PRP), and (iii) indistinguishability of encryptions (either symmetric or not).

We start with defining a cryptographic game. We formulate a cryptographic game as an interactive game between a principal and an adversary. The principal maintains a state. In each round of an attack the adversary sends a pair (a_i, q_i) where q_i represents a query to the keyed function and a_i some additional information (for instance, a_i could be a bit indicating whether the adversary wishes to receive the upper half or the lower half of $\mathcal{F}_k(q_i)$). The principal responds with some function of its current state and the query. At the end, the adversary issues a guess and a tester decides whether to accept or not based on the state of the principal and the queries made (this determines who won the game).

▶ **Definition 20.** *A **cryptographic game** $G_{\mathcal{F}}$ for a keyed function $\mathcal{F}_k(x)$ between a principal and an adversary is a game defined by three PPT algorithms*

$$(\mathsf{StateTransition}, \mathsf{Answer}, \mathsf{Tester}),$$

where the first two define the actions of the principal and Tester *produces outputs in* $\{0,1\}$ *and determines who won the game. Specifically, at round i:*

- *Function* $\mathsf{StateTransition}$ *gets as input a state* S_{i-1} *and the current query pair issued by the adversary* (a_i, q_i) *and the value of* $\mathcal{F}_k$ *of* q_i. *The new state is* S_i. *In other words* S_i *is defined by:*

$$\mathsf{S}_i := \mathsf{StateTransition}(a_i, q_i, \mathsf{S}_{i-1}, \mathcal{F}_k(q_i)).$$

- Answer *takes as input the current state* (S_i) *and returns a response* $\mathsf{CA_i}$ *to the adversary (e.g. this may simply be* $\mathcal{F}_k(q_i)$ *or some function of it). That is*

$$\mathsf{CA_i} := \mathsf{Answer}(\mathsf{S}_i).$$

- Tester *gets as input all the queries pairs* $(q_i, a_i)_{i=1}^{\ell}$ *and states* $(\mathsf{S}_i)_{i=1}^{\ell}$ *as well the adversary's response* guess *and outputs either 0 or 1. If the output is 1 the adversary wins the game, otherwise it loses the game.*

For an adversary Adv, *the game* $G_{\mathcal{F}}$ *is the following:*

1. *A key $k \sim \mathcal{K}$ is chosen randomly.*
2. *Learning Phase: for $\ell = poly(n)$ rounds where at round i:*
   a. Adv *chooses a query q_i that depends on $q_1, \dots, q_{i-1}$ as well as* $\mathsf{CA}_1, \dots, \mathsf{CA}_{i-1}$.
   b. *The principal generates* $\mathsf{S}_i := \mathsf{StateTransition}(a_i, q_i, \mathsf{S}_{i-1}, \mathcal{F}_k(q_i))$.
   c. *The principal sends* Adv *an answer* CA_i *which is a function of* S_i *and i.*
3. *Guessing Phase: The adversary generates* guess.
4. *Testing Phase: If* $\mathsf{Tester}((a_1, q_1), \dots, (a_\ell, q_\ell), \mathsf{S}_0, \mathsf{S}_1, \dots, \mathsf{S}_\ell, \mathsf{guess}) = 1$, *then* Adv *wins the game and otherwise it loses the game.*

Note that Tester does not know the key k, nor does it have query access to $\mathcal{F}_k$, hence the term **noticeable security**. Let $\tau < 1$ be a “benign” success probability. If the adversary cannot win the cryptographic game of a keyed function with probability much better than τ, then we say that the keyed function is noticeable secure. Specifically:

▶ **Definition 21.** *Let $\tau < 1$, the “benign” success probability. A keyed function $\mathcal{F}_k(x)$ is* ***$(G_{\mathcal{F}}, \tau)$-noticeable-secure*** *if any PPT adversary* Adv *has probability at most $\tau + \mathsf{negl}(n)$ to win the game $G_{\mathcal{F}}$.*

As we shall see, classical notions of security can be expressed within the framework of cryptographic games presented above. In Appendix A we demonstrate that indistinguishability of encryptions, digital signatures, and pseudorandom permutations can each be formulated in terms of a cryptographic game.

Adding Timing to the Game

Next, we define $(G_{\mathcal{F}}, \tau)$-time-secure which intuitively says that the winning probability in the game $G_{\mathcal{F}}$ remains $\tau + \mathsf{negl}(n)$ even when the adversary gets, not only the answers for the oracles queries, but also the time it takes for the oracle to execute them. More formally:

▶ **Definition 22.** *Let $\tau < 1$, the “benign” success probability. The implementation of a keyed function $\mathcal{F}_k(x)$ is* ***$(G_{\mathcal{F}}, \tau)$-time-secure*** *if any PPT adversary* Adv *has probability at most $\tau + \mathsf{negl}(n)$ to win the game $G_{\mathcal{F}}$ with the following modification: in step 2.c of the game $G_{\mathcal{F}}$ as in Definition 20, the principal sends both CA_i and the time it takes for the principal to execute $\mathcal{F}_k$, that is $\mathcal{T}(\mathcal{F}_k(q_i))$.*

Main Theorem: Key Oblivious Implies Time-Security

While key obliviousness suggests that the running time doesn’t disclose significant information about the key, the security of a keyed function $\mathcal{F}_k$ is assessed through $G_{\mathcal{F}}$. Therefore, it’s imperative that the running times of $\mathcal{F}$ don’t significantly aid an attacker in winning $G_{\mathcal{F}}$. The following theorem establishes that key-obliviousness of a keyed function implies it’s resistance against timing attacks (w.r.t $G_{\mathcal{F}}$), whenever $G_{\mathcal{F}}$ has noticeable security.

▶ **Theorem 23.** *Let $\mathcal{F}_k$ be a keyed function that is $(G_{\mathcal{F}}, \tau)$-noticeable-secure for $\tau < 1$. If the implementation of $\mathcal{F}_k$ is key-oblivious, then the implementation of $\mathcal{F}_k$ is $(G_{\mathcal{F}}, \tau)$-time-secure.*

We can use the theorem above, in addition to Propositions 28, 30 and 32, in order to obtain the following corollary:

▶ **Corollary 24.** *The cryptographic schemes: digital signature, pseudo-random permutation and encryption are $(G_{\mathcal{F}}, \tau)$-time-secure with the corresponding τ for each game, provided the implementation of each scheme is key-oblivious.*

▶ Remark 25. In some cases the converse of Theorem 23 holds, meaning that if $\mathcal{F}$ is not $(G_{\mathcal{F}}, \tau)$-time-secure, and $G_{\mathcal{F}}$ is noticeable, then the implementation of $\mathcal{F}_k$ is not key-oblivious. In particular, in the extended version we prove this fact for PRPs and PRFs.

5.2 Caveat on Key-Oblivious and an Application of Query-Oblivious

It is essential not to abuse Theorem 23 and Corollary 24 and apply them correctly. The point is that the setting considered assumes that the timing information gained by the adversary follows the pattern of the timing game (Definition 20). But this may not necessarily be the case and it could be that the adversary does not know which query q is evaluated by the implementation of $\mathcal{F}_k$ at a given point and the time $\mathcal{T}(\mathcal{F}_k(q))$ it takes to compute $\mathcal{F}_k(q)$ may reveal information about q itself and compromise the security of the system.

A case in point is that of indistinguishably of encryption, where following a learning session an adversary selects a pair of messages (m_0, m_1) and receives either $\mathcal{F}_k(m_0)$ or $\mathcal{F}_k(m_1)$ and has to make a guess about the value of b where the ciphertext is of m_b. We saw in Proposition 32 that it falls into the framework of noticeable security. But this assumed that the principal computes both $\mathcal{F}_k(m_0)$ and $\mathcal{F}_k(m_1)$ before selecting which one to send. But a natural version of it is to first select whether to encrypt m_0 or m_1 and evaluate $\mathcal{F}$ only on the selected point. Thus the time to compute $\mathcal{F}_k(m_b)$ may reveal which one was selected. Note that here we have two games that are equivalent in the non-timed version (Definition 20), i.e. the adversary has exactly the same probability of winning in these two games, but when considering the timed version (Definition 22) these two games are not equivalent any more, since there function $\mathcal{F}_k$ is called after each query.

So given an implementation of a system, to argue that it is secure in terms of timing, one must argue that the setting of Definition 20 is the relevant one to the given system. Requiring the implementation of the encryption to be query-with-result-obliviousness solves the above situation in the indistinguishability of encryption game. If the implementation is query-with-result-oblivious, then the two games are equivalent also in the timed version:

▶ **Theorem 26.** *Suppose $\mathcal{F}_k$ is a keyed function which is (G_{IND}, τ)-noticeable-secure. If the implementation of $\mathcal{F}_k$ is query-with-results-oblivious, then $\mathcal{F}_k$ is also (G_{IND}, τ)-time-secure.*

Proof. This is straightforward since the query-with-results-oblivious game is the exact same game as indistinguishability of encryptions with timing. ◀

To conclude, arguing that the properties of key and query obliviousness are relevant to a given system is a delicate matter. Key obliviousness is most relevant to systems where the timing information is exposed for a certain period of time, but then when the attack is made, there is no leakage. Think of the definition of semantic security of encryption. If the attacker chooses a distribution on which it is to be tested, the encryption of the challenge message should be done after the leakage is over (note that it does not fall into the game framework, since there at any point it is clear what call is made to the keyed function).

On the other hand query obliviousness is relevant when the timing information is leaked about the message or challenge that is protected.

References

1 José Bacelar Almeida, Manuel Barbosa, Gilles Barthe, François Dupressoir, and Michael Emmi. Verifying constant-time implementations. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, pages 53–70. USENIX Association, 2016. URL: `https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/almeida`.

2 Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Survey: Leakage resilience and the bounded retrieval model. In Kaoru Kurosawa, editor, *Information Theoretic Security, 4th International Conference, ICITS 2009, Shizuoka, Japan, December 3-6, 2009. Revised Selected Papers*, volume 5973 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2009. `doi:10.1007/978-3-642-14496-7_1`.

3 Yuriy Arbitman, Moni Naor, and Gil Segev. Backyard cuckoo hashing: Constant worst-case operations with a succinct representation. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 787–796. IEEE Computer Society, 2010. `doi:10.1109/FOCS.2010.80`.

4 Gustavo Banegas, Daniel J. Bernstein, Fabio Campos, Tung Chou, Tanja Lange, Michael Meyer, Benjamin Smith, and Jana Sotáková. CTIDH: faster constant-time CSIDH. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(4):351–387, 2021. doi:10.46586/TCHES.V2021.I4.351-387.

5 Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Pseudorandom functions revisited: The cascade construction and its concrete security. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 514–523. IEEE Computer Society, 1996. doi:10.1109/SFCS.1996.548510.

6 Mihir Bellare, Thomas Ristenpart, Phillip Rogaway, and Till Stegers. Format-preserving encryption. In Michael J. Jacobson Jr., Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *Selected Areas in Cryptography, 16th Annual International Workshop, SAC 2009, Calgary, Alberta, Canada, August 13-14, 2009, Revised Selected Papers*, volume 5867 of *Lecture Notes in Computer Science*, pages 295–312. Springer, 2009. doi:10.1007/978-3-642-05445-7_19.

7 Itay Berman, Iftach Haitner, Ilan Komargodski, and Moni Naor. Hardness-preserving reductions via cuckoo hashing. *J. Cryptol.*, 32(2):361–392, 2019. doi:10.1007/s00145-018-9293-0.

8 Arnab Kumar Biswas, Dipak Ghosal, and Shishir Nagaraja. A survey of timing channels and countermeasures. *ACM Comput. Surv.*, 50(1):6:1–6:39, 2017. doi:10.1145/3023872.

9 John Black and Phillip Rogaway. Ciphers with arbitrary finite domains. In Bart Preneel, editor, *Topics in Cryptology - CT-RSA 2002, The Cryptographer's Track at the RSA Conference, 2002, San Jose, CA, USA, February 18-22, 2002, Proceedings*, volume 2271 of *Lecture Notes in Computer Science*, pages 114–130. Springer, 2002. doi:10.1007/3-540-45760-7_9.

10 Jonathan Bootle, Claire Delaplace, Thomas Espitau, Pierre-Alain Fouque, and Mehdi Tibouchi. LWE without modular reduction and improved side-channel attacks against BLISS. In *Advances in Cryptology - ASIACRYPT 2018, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part I*, volume 11272 of *Lecture Notes in Computer Science*, pages 494–524. Springer, 2018. doi:10.1007/978-3-030-03326-2_17.

11 David Brumley and Dan Boneh. Remote timing attacks are practical. In *Proceedings of the 12th USENIX Security Symposium, Washington, D.C., USA, August 4-8, 2003.* USENIX Association, 2003. URL: https://www.usenix.org/conference/12th-usenix-security-symposium/remote-timing-attacks-are-practical.

12 David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982*, pages 199–203. Plenum Press, New York, 1982. doi:10.1007/978-1-4757-0602-4_18.

13 Scott A. Crosby, Dan S. Wallach, and Rudolf H. Riedi. Opportunities and limits of remote timing attacks. *ACM Trans. Inf. Syst. Secur.*, 12(3):17:1–17:29, 2009. doi:10.1145/1455526.1455530.

14 Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 40–56. Springer, 2013. doi:10.1007/978-3-642-40041-4_3.

15 Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongswan and electromagnetic emanations in microcontrollers. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1857–1874. ACM, 2017. doi:10.1145/3133956.3134028.

16 Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM (JACM)*, 33(4):792–807, 1986.

17 Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *Journal of the ACM (JACM)*, 43(3):431–473, 1996.

18 Shafi Goldwasser and Guy N. Rothblum. How to compute in the presence of leakage. *SIAM J. Comput.*, 44(5):1480–1549, 2015. doi:10.1137/130931461.

19 Viet Tung Hoang, Ben Morris, and Phillip Rogaway. An enciphering scheme based on a card shuffle. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2012. doi:10.1007/978-3-642-32009-5_1.

20 Yuval Ishai, Amit Sahai, and David A. Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003 Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003. doi:10.1007/978-3-540-45146-4_27.

21 Yael Tauman Kalai and Leonid Reyzin. A survey of leakage-resilient cryptography. In Oded Goldreich, editor, *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 727–794. ACM, 2019. doi:10.1145/3335741.3335768.

22 Jonathan Katz and Chiu-Yuen Koo. On constructing universal one-way hash functions from arbitrary one-way functions. *IACR Cryptol. ePrint Arch.*, page 328, 2005. URL: http://eprint.iacr.org/2005/328.

23 Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Advances in Cryptology - CRYPTO '96, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996. doi:10.1007/3-540-68697-5_9.

24 Richard J. Lipton and Jeffrey F. Naughton. Clocked adversaries for hashing. *Algorithmica*, 9(3):239–252, 1993. doi:10.1007/BF01190898.

25 Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.*, 17(2):373–386, 1988. doi:10.1137/0217022.

26 Ueli M. Maurer and Krzysztof Pietrzak. Composition of random systems: When two weak make one strong. In Moni Naor, editor, *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 410–427. Springer, 2004. doi:10.1007/978-3-540-24638-1_23.

27 Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In Moni Naor, editor, *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004. doi:10.1007/978-3-540-24638-1_16.

28 Ben Morris and Phillip Rogaway. Sometimes-recurse shuffle - almost-random permutations in logarithmic expected time. In *Advances in Cryptology - EUROCRYPT 2014, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 311–326. Springer, 2014. doi:10.1007/978-3-642-55220-5_18.

29 Moni Naor and Omer Reingold. On the construction of pseudorandom permutations: Luby-rackoff revisited. *J. Cryptol.*, 12(1):29–66, 1999. doi:10.1007/PL00003817.

30 Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. *SIAM J. Comput.*, 41(4):772–814, 2012. doi:10.1137/100813464.

31 Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 33–43. ACM, 1989. doi:10.1145/73007.73011.

32 Thomas Ristenpart and Scott Yilek. The mix-and-cut shuffle: Small-domain encryption secure against N queries. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 392–409. Springer, 2013. doi:10.1007/978-3-642-40041-4_22.

33 John Rompel. One-way functions are necessary and sufficient for secure signatures. In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 387–394. ACM, 1990. doi:10.1145/100216.100269.

34 Steven Rudich. Limits on the provable consequences of one-way functions. *PhD Thesis, University of California*, 1988.

35 Emil Stefanov, Elaine Shi, and Dawn Song. Towards practical oblivious ram. *arXiv preprint arXiv:1106.3652*, 2011.

A Classical Security Definitions are Noticeable Secure

We now give a few examples of classical security definitions of keyed functions and show how they fit Definition 20, namely we show they are noticeable secure. Specifically, we show for digital signatures, pseudo-random permutations, and indistinguishability of encryptions.

▶ **Definition 27.** *(Digital signatures) Let* Adv *be an adversary in the following game:*

1. *The principal generates public and secret keys* (pk, sk) *and shares* pk *with adversary* Adv.
2. *Adversary* Adv *chooses adaptively* $\ell = poly(n)$ *messages* $m_1, \dots, m_\ell$ *and gives them to the principal, receiving their signatures*

$$\sigma_1 = \mathsf{Sign}_{sk}(m_1), \dots, \sigma_\ell = \mathsf{Sign}_{sk}(m_\ell).$$

3. *Adversary* Adv *chooses a new message* $m' \notin \{m_1, \dots, m_\ell\}$. *Adversary* Adv succeeds *if and only if it can generate* $\sigma' = \mathsf{Sign}_{sk}(m')$ *s.t.* $\mathsf{Vrfy}_{pk}(m', \sigma') = 1$.

A signature scheme $\Pi = (\mathsf{Sign}, \mathsf{Vrfy})$ *is existentially unforgeable under an adaptive chosen-message attack, or just secure, if for all probabilistic polynomial-time adversaries* Adv, *there is a negligible function* negl *such that:*

$$\Pr[\mathsf{Adv}\ \textit{success}] \le \mathsf{negl}(n).$$

It is relatively straightforward to phrase the security of digital signatures in the game framework of Section 5.1. Let G_{DS} be the following cryptographic game:

1. The principal generates public and secret keys (pk, sk) and shares pk with adversary Adv.

$$\mathsf{S}_0 = pk$$

2. Learning phase: For $i \in [1, \ell]$,
   a. Adv chooses adaptively a message m_i and gives it to the principal.
   b. The principal is essentially stateless, i.e. its current state is simply the signature $\mathsf{Sign}_{sk}(m_i)$ on m_i.
   c. The answer function is simply to send the full state to the adversary, i.e.

$$\mathsf{CA}_i = \mathsf{S}_i = \mathsf{Sign}_{sk}(m_i).$$

3. Guessing phase: Adversary Adv chooses a new message $m' \notin \{m_1, \dots, m_\ell\}$ and calculates

$$\mathsf{guess} = (m', \mathsf{Sign}_{sk}(m')).$$

4. Testing phase: Tester returns 1 iff m' is not in $\{m_1, m_2, \dots, m_\ell\}$ and $\mathsf{Vrfy}_{pk}(\mathsf{guess}) = 1$.

▶ **Proposition 28.** *A digital signature scheme is secure according to Definition 27 iff it is* $(G_{\mathsf{DS}}, 0)$*-noticeable-secure.*

Notice that when discussing key-obliviousness for digital signatures the adversary also knows the corresponding public-key to k_0 (but not the one corresponding to k_1).

▶ **Definition 29.** *(Strong Pseudo-random Permutations) Let* Adv *be a probabilistic polynomial-time adversary in the following game:*

1. *A key $k \sim \mathcal{K}$ is sampled.*
2. *A random bit b is sampled.*
3. Adv *chooses adaptively $\ell = poly(n)$ queries $(x_1, s_1), \ldots, (x_\ell, s_\ell)$ where x_i is a message and $s_i \in \{-1, 1\}$ is the oracle direction. If $b = 0$ then it receives the values of E_k on each query*

   $$y_1 = E_k^{s_1}(x_1), \ldots, y_\ell = E_k^{s_\ell}(x_\ell).$$

   If $b = 1$ then it receives the values of a random permutation f on each query

   $$y_1 = f^{s_1}(x_1), \ldots, y_\ell = f^{s_\ell}(x_\ell).$$

4. *Adversary* Adv *guesses b' and wins if $b' = b$.*

The function E_k is a pseudo-random permutation if for all PPT adversaries Adv *as above there is a negligible function* negl *such that, for all n,*

$$\Pr[b = b'] \leq 1/2 + \mathsf{negl}(n).$$

Note that a random permutation f can be simulated perfectly by always picking a random value that has not appeared so far (while being consistent with the values that have appeared).

As before, it is relatively straightforward to phrase the security of a PRP in terms of a game of Definition 20. Let the G_{PRP} be:

1. A random key $k \sim \mathcal{K}$ is chosen and the principal chooses a random bit b

   $$\mathsf{S}_0 = b.$$

   The bit b remains in the state of the principal throughout the game.
2. Learning Phase: For $i \in [1, \ell]$:
   a. Adversary Adv chooses adaptively a query (x_i, s_i) where x_i is a message and $s_i \in \{-1, 1\}$ is the direction of the permutation (forward or background).
   b. The principal state includes the bit b and all the queries $(x_1, s_1, y_1), \ldots, (x_{i-1}, s_{i-1}, y_{i-1})$ values it sent and received. The state transition gets in addition to the current state the value of $E_k^{s_i}(x_i)$. If $b = 0$ then $y_i = E_k^{s_i}(x_i)$ and if $b = 1$ then, if x_i is equal to any of previous values x_j or y_j (with the appropriate s_j), then set y_i as was the previous response. Else choose y_i to be a random value that has not appeared so far.
   c. The answer

      $$\mathsf{CA}_i = y_i.$$

3. Guessing Phase: Adversary Adv guesses guess $:= b'$.
4. Testing phase: returns 1 and wins if guess $= b$.

▶ **Proposition 30.** *A strong pseudo-random permutation is secure iff it is $(G_{\mathsf{PRP}}, \frac{1}{2})$-noticeable-secure.*

Our final example is indistinguishability of a private-key encryption scheme against chosen plaintext attacks (the same also works in public-key setting):

▶ **Definition 31.** *(Indistinguishability of encryptions) Let* Adv *be a PPT adversary in the following game:*

1. *A random key $k \sim \mathcal{K}$ is chosen.*
2. *Adversary* Adv *chooses adaptively $\ell = poly(n)$ messages $m_1, \dots, m_\ell$ and gives them to the principal, receiving their encryption in the form of ciphertext*
$$C_1 = E_k(m_1), C_2 = E_k(m_2), \dots, C_\ell = E_k(m_\ell).$$
3. *Adversary* Adv *then chooses two more messages $m'_0 \neq m'_1$ such that $|m'_0| = |m'_1|$ (not necessarily distinct from the previous messages) and sends them to the principal.*
4. *The principal chooses randomly $b \in_R \{0,1\}$ and sends $E_k(m'_b)$ back to adversary* Adv*.*
5. Adv *chooses adaptively another collection of $\ell = poly(n)$ messages $m_{\ell+1}, \dots, m_{\ell+\ell}$ and receives their encryption in the form of ciphertexts*
$$C_{\ell+1} = E_k(m_{\ell+1}), \dots, C_{\ell+\ell} = E_k(m_{\ell+\ell}).$$
6. *Adversary* Adv *guesses b' and wins if $b' = b$.*

A private-key encryption scheme is said to have the property of indistinguishablity of encryptions under a chosen-plaintext attack, if for all PPT adversaries Adv *as above there is a negligible function* negl *such that for all n*
$$\Pr[b = b'] \leq 1/2 + \mathsf{negl}(n).$$

It is relatively straightforward to phrase the security of indistinguishability of encryptions in the game framework of Section 5.1. Let G_{IND} be the following cryptographic game:

1. A random key $k \sim \mathcal{K}$ is chosen and the principal chooses a random bit b.
$$\mathsf{S}_0 = b.$$
2. Learning Phase:
   - For each $i \in [1, \ell - 1]$, adversary Adv chooses adaptively a message m_i and gives it to the principal, receiving its encryption in the form of ciphertext
   $$\mathsf{CA}_i = \mathsf{S}_i = E_k(m_i).$$
   - Adversary Adv chooses two more messages $m_{\ell-1} \neq m_\ell$ such that $|m_{\ell-1}| = |m_\ell|$ and sends them to the principal. The principal calculates
   $$\mathsf{S}_{\ell-1} = E_k(m_{\ell-1}) \text{ and } \mathsf{S}_\ell = (E_k(m_{\ell-1}), E_k(m_\ell), b).$$
   $$\mathsf{CA}_{\ell-1} = \mathsf{NULL} \text{ and } \mathsf{CA}_\ell = (E_k(m_{\ell-1+b}),$$
   - For each $i \in [\ell + 1, 2\ell]$, adversary Adv chooses adaptively a message m_i and gives it to the principal, receiving its encryption in the form of ciphertext
   $$\mathsf{CA}_i = \mathsf{S}_i = E_k(m_i).$$
3. Guessing phase: Adversary Adv guesses $\mathsf{guess} = b'$
4. Testing phase: returns 1 if $\mathsf{guess} = b$.

▶ **Proposition 32.** *The indistinguishability of encryptions game G_{IND} is secure iff it is $(G_{\mathsf{IND}}, 0.5)$-noticeable-secure.*

B Proof of Theorem 23

Proof. Recall that $G_{\mathcal{F}}$ is the cryptographic game of $\mathcal{F}_k$. We prove the theorem in two steps. We first consider a random-time game of $G_{\mathcal{F}}$, which we denote by $G^R_{\mathcal{F}}$. The random-time game $G^R_{\mathcal{F}}$ is similar to $G_{\mathcal{F}}$, with the following changes: (1) at the beginning of the game a random key k' is generated (unrelated to the original key k), and (2) besides receiving CA_i from the principal, the adversary receives the timing information $\mathcal{T}(\mathcal{F}_{k'}(q_i))$, which is the running time it takes for $\mathcal{F}_{k'}$ (namely, using the random key k') to be executed on q_i. It is easy to see that the winning probability of $G^R_{\mathcal{F}}$ is the same as the winning probability of $G_{\mathcal{F}}$. To see that, assume adversary $\mathsf{Adv}_{\mathsf{GR}}$ attacks $G^R_{\mathcal{F}}$. We build adversary $\mathsf{Adv}_{\mathsf{G}}$ attacking $G_{\mathcal{F}}$. Adversary $\mathsf{Adv}_{\mathsf{G}}$ works as follows: it first chooses a random key k', then for every query q_i from adversary $\mathsf{Adv}_{\mathsf{GR}}$, it calculates $\mathcal{T}(\mathcal{F}_{k'}(q_i))$ and sends $\mathsf{CA_i}$ together with $\mathcal{T}(\mathcal{F}_{k'}(q_i))$ to adversary $\mathsf{Adv}_{\mathsf{GR}}$. Finally, adversary $\mathsf{Adv}_{\mathsf{G}}$ returns the guess of adversary $\mathsf{Adv}_{\mathsf{GR}}$. From the description it holds that

$$\Pr[\mathsf{Adv}_{\mathsf{G}} \text{ wins } G_{\mathcal{F}}] = \Pr[\mathsf{Adv}_{\mathsf{GR}} \text{ wins } G^R_{\mathcal{F}}] \leq \tau + \mathsf{negl}.$$

Now, let $G^T_{\mathcal{F}}$ be the game defining the time-security of $\mathcal{F}_k$, namely it is similar to $G_{\mathcal{F}}$ but in addition to CA_i, the adversary also receives $\mathcal{T}(\mathcal{F}_k(q_i))$ from the principal. Let $\mathsf{Adv}_{\mathsf{GT}}$ be adversary attacking the game $G^T_{\mathcal{F}}$. We show that if the implementation of $\mathcal{F}_k$ is key-oblivious then

$$\Pr[\mathsf{Adv}_{\mathsf{GT}} \text{ wins } G^T_{\mathcal{F}}] - \Pr[\mathsf{Adv}_{\mathsf{GR}} \text{ wins } G^R_{\mathcal{F}}] \leq \mathsf{negl} \tag{B.1}$$

and this will imply

$$\Pr[\mathsf{Adv}_{\mathsf{GT}} \text{ wins } G^T_{\mathcal{F}}] \leq \tau + \mathsf{negl}.$$

To prove Equation B.1, we assume there exists adversary $\mathsf{Adv}_{\mathsf{GT}}$ attacking the game $G^T_{\mathcal{F}}$ and we build adversary $\mathsf{Adv}_{\mathsf{KO}}$ attacking the key-obliviousness of $\mathcal{F}_k$. Adversary $\mathsf{Adv}_{\mathsf{KO}}$ works as follows:

- Adversary $\mathsf{Adv}_{\mathsf{KO}}$ receives polynomially many queries q_i from adversary $\mathsf{Adv}_{\mathsf{GT}}$ and sends them to the principal.
- Adversary $\mathsf{Adv}_{\mathsf{KO}}$ receives from the principal $\mathcal{F}_{k_0}(q_i)$ and $\mathcal{T}(\mathcal{F}_{k_b}(q_i))$ for a random bit b. Adversary $\mathsf{Adv}_{\mathsf{KO}}$ sends these $\mathcal{F}_{k_0}(q_i)$ and $\mathcal{T}(\mathcal{F}_{k_b}(q_i))$ to adversary $\mathsf{Adv}_{\mathsf{GT}}$.
- Adversary $\mathsf{Adv}_{\mathsf{GT}}$ returns its guess. Adversary $\mathsf{Adv}_{\mathsf{KO}}$ tests if the guess is correct using Tester and receives $w = \mathsf{Tester}(\mathsf{guess})$. If $w = 1$, then $\mathsf{Adv}_{\mathsf{KO}}$ returns $b' = 0$ since the queries were executed on $\mathcal{F}_{k_0}$. If $w = 0$, it returns $b' = 1$.

$$\begin{aligned}
\Pr[\mathsf{Adv}_{\mathsf{KO}} \text{ wins key-oblivious game}] &= \Pr[b' = 0|b = 0] \cdot \Pr[b = 0] + \Pr[b' = 1|b = 1] \cdot \Pr[b = 1] \\
&= \Pr[b' = 0|b = 0] \cdot \Pr[b = 0] + (1 - \Pr[b' = 0|b = 1]) \cdot \Pr[b = 1] \\
&= \Pr[\mathsf{Adv}_{\mathsf{GT}} \text{ wins } G^T_{\mathcal{F}}] \cdot 0.5 + (1 - \Pr[\mathsf{Adv}_{\mathsf{GR}} \text{ wins } G^R_{\mathcal{F}}) \cdot 0.5
\end{aligned}$$

Since $\Pr[\mathsf{Adv}_{\mathsf{KO}}$ wins key-oblivious game$] \leq 0.5 + \mathsf{negl}$, we get Equation B.1. ◀

Pure-DP Aggregation in the Shuffle Model: Error-Optimal and Communication-Efficient

Badih Ghazi
Google Research, Mountain View, CA, USA

Ravi Kumar
Google Research, Mountain View, CA, USA

Pasin Manurangsi
Google Research, Bangkok, Thailand

Abstract

We obtain a new protocol for binary counting in the ϵ-DP$_{\text{shuffle}}$ model with error $O(1/\epsilon)$ and expected communication $\widetilde{O}\left(\frac{\log n}{\epsilon}\right)$ messages per user. Previous protocols incur either an error of $O(1/\epsilon^{1.5})$ with $O_\epsilon(\log n)$ messages per user (Ghazi et al., ITC 2020) or an error of $O(1/\epsilon)$ with $O_\epsilon(n^2)$ messages per user (Cheu and Yan, TPDP 2022). Using the new protocol, we obtained improved ϵ-DP$_{\text{shuffle}}$ protocols for real summation and histograms.

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

Keywords and phrases Differential Privacy, Shuffle Model, Aggregation, Pure Differential Privacy

Digital Object Identifier 10.4230/LIPIcs.ITC.2024.4

1 Introduction

Differential privacy (DP) [11] is a widely accepted notion used for bounding and quantifying an algorithm's leakage of personal information. Its most basic form, known as *pure*-DP, is governed by a single parameter $\epsilon > 0$, which bounds the leakage of the algorithm. Specifically, a randomized algorithm $A(\cdot)$ is said to be *ϵ-DP* if for any subset S of output values, and for any two datasets D and D' differing on a single user's data, it holds that $\Pr[A(D) \in S] \le e^\epsilon \cdot \Pr[A(D') \in S]$. In settings where pure-DP is not (known to be) possible, a common relaxation is the so-called *approximate*-DP [10], which has an additional parameter $\delta \in [0, 1]$. In this case, the condition becomes: $\Pr[A(D) \in S] \le e^\epsilon \cdot \Pr[A(D') \in S] + \delta$. Understanding the gap between pure- and approximate-DP algorithms is a natural and fundamental question that has been studied for a variety of analytics tasks (e.g., [23, 6]). Besides this, pure-DP protocols might be preferable in practice since an approximate-DP protocol may allow a (very small) non-zero probability of a catastrophic event, e.g., that the entire database is leaked[1].

Depending on the trust assumptions, three models of DP are commonly studied. The first is the *central* model, where a trusted curator is assumed to hold the raw data and is required to release a private output; this goes back to the first work of Dwork et al. [11] on DP. The second is the *local* model [13, 11, 22], where each user's message is required to be private. The third is the *shuffle* model [5, 8, 12], where the users' messages are routed through a trusted shuffler, which is assumed to be non-colluding with the curator, and which is expected to randomly permute the messages incoming from the different users (DP$_{\text{shuffle}}$). Formally, a protocol $P = (R, S, A)$ in the shuffle model consists of three procedures: (i)

[1] Indeed, such a catastrophic event can happen in some approximate-DP$_{\text{shuffle}}$ protocols proposed in previous works [18, 19].

a local randomizer $R(\cdot)$ that takes as input the data of a single user and outputs one or more messages, (ii) a shuffler $S(\cdot)$ that randomly permutes the messages from all the local randomizers, and (iii) an analyst $A(\cdot)$ that consumes the permuted output of the shuffler; the output of the protocol P is the output of the analyst $A(\cdot)$. Privacy in the shuffle model is defined as follows:

▶ **Definition 1** ([8, 12]). *A protocol $P = (R, S, A)$ is said to be (ϵ, δ)-$\mathrm{DP}_{\text{shuffle}}$ if for any input dataset $D = (x_1, \dots, x_n)$ where n is the number of users, it holds that $S(R(x_1), \dots, R(x_n))$ is (ϵ, δ)-DP. In the case where $\delta = 0$, the protocol P is said to be ϵ-$\mathrm{DP}_{\text{shuffle}}$.*

For several analytics tasks, low-error algorithms are known in the central model, whereas such algorithms are known to be impossible in the local model. For these analytic tasks, low-error algorithms are commonly sought in the shuffle model, since it is more preferable to trust a shuffler than a central curator. We note that while in this paper we treat the shuffler as a black box, multiple possible implementations have been considered in the literature including via secure hardware, mixnets and lightweight cryptographic protocols; see, e.g., the discussion in [5].

Interestingly, almost all algorithms studied in the shuffle model are for the approximate-DP setting. The only exceptions, to the best of our knowledge, are the pure-DP algorithms of Ghazi et al. [15] and Cheu and Yan [9] for binary summation; we discuss these next.

1.1 Our Contributions

In the *binary summation* (aka *counting*) problem, each user i receives an input $x_i \in \{0, 1\}$ and the goal is to estimate $\sum_{i \in [n]} x_i$. For this well-studied task, the discrete Laplace mechanism is known to achieve the optimal (expected absolute) error of $O(1/\epsilon)$ for ϵ-DP summation in the central model [21, 14]. Note that this error is independent of the number n of users, and is an absolute constant for the common parameter regime where ϵ is a constant. In contrast, the error of any aggregation protocol in the local model is known to be at least on the order of $\sqrt{n}$ [4, 7]. There have been many works that studied aggregation in the $\mathrm{DP}_{\text{shuffle}}$ setting including [2, 3, 20, 15, 18, 19, 17, 1]. For pure-DP aggregation, it is known that any single-message protocol (where each user sends a single message to the shuffler) should incur error $\Omega_\epsilon(\sqrt{n})$ [1]. For multi-message protocols, where each user can send multiple messages to the shuffler, the best known protocols incur either an error of $O(1/\epsilon^{1.5})$ with $O(\log n)$ messages per user [15] or an error of $O(1/\epsilon)$ with $O(n^2)$ messages per user [9]. No protocol simultaneously achieved error $O(1/\epsilon)$ and communication $O(\log n)$.

In this paper, we obtain an ϵ-$\mathrm{DP}_{\text{shuffle}}$ algorithm for binary summation, where each user, in expectation, sends $O\left(\frac{\log n}{\epsilon}\right)$ one-bit messages; this answers the main open question for this basic aggregation task.

▶ **Theorem 2.** *For every positive real number $\epsilon \leq O(1)$, there is a (non-interactive) ϵ-$\mathrm{DP}_{\text{shuffle}}$ protocol for binary summation with root mean square error $O(1/\epsilon)$, where each user sends $O\left(\frac{\log n}{\epsilon}\right)$ messages in expectation and each message consists of a single bit.*

In fact, similar to the protocol of Cheu and Yan [9], our protocol can get an error that is arbitrarily close to that of the discrete Laplace mechanism, which is known to be optimal in the central model for any $\epsilon > 0$; see [21, 14]. We defer the formal statement to Theorem 6.

Before we continue, we note that while the expected number of messages in Theorem 2 is small (and with an exponential tail), the *worst* case number of messages is unbounded. This should be contrasted with an $\Omega_\epsilon(\sqrt{\log n})$ lower bound in [15] that only applies to the worst case number of bits sent by a user. We discuss this further in Section 6.

Protocols for Real Summation and Histogram

Using known techniques (e.g., [8, 15]), we immediately get the following consequences for real summation and histogram.

In the *real summation* problem, each x_i is a real value in $[0, 1]$; the goal is again to estimate the sum $\sum_{i \in [n]} x_i$. The protocol in [15] achieves an expected root mean square error (RMSE) of $\widetilde{O}(1/\epsilon^{1.5})$; here, each user sends $O_\epsilon(\log^3 n)$ messages each of length $O(\log \log n)$ bits. By running their protocol bit-by-bit with an appropriate privacy budget split, we get an algorithm with an improved, and asymptotically optimal, error of $O(1/\epsilon)$ while with expected communication similar to theirs.

▶ **Corollary 3.** *For every positive real number $\epsilon \leq O(1)$, there is a (non-interactive) ϵ-$\mathrm{DP}_{\mathsf{shuffle}}$ protocol for real summation with RMSE $O(1/\epsilon)$, where each user sends $O(\frac{\log^3 n}{\epsilon})$ messages in expectation and each message consists of $O(\log \log n)$ bits.*

A widely used primitive related, though not identical, to aggregation is histogram computation. In the *histogram* problem, each x_i is a number in $[B]$; the goal is to estimate the histogram of the dataset, where the histogram $\mathbf{h} \in \mathbb{Z}^B_{\geq 0}$ is defined by $h_b = |\{i \in [n] \mid x_i = b\}|$. The error of an estimated histogram $\tilde{\mathbf{h}}$ is usually measured in the ℓ_∞-sense, i.e., $\|\tilde{\mathbf{h}} - \mathbf{h}\|_\infty = \max_{b \in [B]} |h_b - \tilde{h}_b|$.

For this task, which has been studied in several papers including [16, 1], the best known pure-$\mathrm{DP}_{\mathsf{shuffle}}$ protocol achieved ℓ_∞-error $O\left(\frac{\log B \log n}{\epsilon^{1.5}}\right)$ and communication $O\left(\frac{B \log n \log B}{\epsilon}\right)$ bits. By running our $(\epsilon/2)$-$\mathrm{DP}_{\mathsf{shuffle}}$ protocol separately for each bucket [15, Appendix A], we immediately arrive at the following:

▶ **Corollary 4.** *For every positive real number $\epsilon \leq O(1)$, there is a (non-interactive) ϵ-$\mathrm{DP}_{\mathsf{shuffle}}$ protocol that computes histograms on domains of size B with an expected ℓ_∞-error of at most $O\left(\frac{\log B \log n}{\epsilon}\right)$, where each user sends $O\left(\frac{B \log n}{\epsilon}\right)$ messages in expectation and each message consists of $O(\log B)$ bits.*

1.2 Technical Overview

We will now briefly discuss the proof of Theorem 2. Surprisingly, we show that a simple modification of the algorithm from [18] satisfies pure-DP! To understand the modification and its necessity, it is first important to understand their algorithm. In their protocol, the messages are either $+1$ or -1, and the analyzer's output is simply the sum of all messages. There are three type of messages each user sends:

- *Input-Dependent Messages*: If the input x_i is 1, the user sends a $+1$ message. Otherwise, the user does not send anything.
- *Flooding Messages*: These are messages that do *not* affect the final estimation error. In particular, a random variable $z_i^{\pm 1}$ is drawn from an appropriate distribution and the user sends $z_i^{\pm 1}$ additional copies of -1 and $z_i^{\pm 1}$ additional copies of $+1$. These messages get canceled when the analyzer computes it output.
- *Noise Messages*: These are the messages that affect the error in the end. Specifically, z_i^{+1}, z_i^{-1} are drawn independently from an appropriate distribution, and z_i^{-1} additional copies of -1 and z_i^{+1} additional copies of $+1$ are then sent.

We note here that the view of the analyzer is simply the number of $+1$ messages and the number of -1 messages, which we will denote by V_{+1} and V_{-1} respectively.

While [18] show that this protocol is (ϵ, δ)-DP, it is easy to show that this is *not* ϵ-DP for any finite ϵ. Indeed, consider two neighboring datasets where X consists of all zeros and X'

consists of a single one and $n-1$ zeros. There is a non-zero probability that $V_{+1}(X) = 0$, while $V_{+1}(X')$ is always non-zero (because of the input-dependent message from the user holding the single one).

To fix this, we randomize this "input-dependent" part. With probability q, the user sends nothing. With the remaining probability $1-q$, (instead of sending a single $+1$ for $x_i = 1$ as in [18],) the user sends $s+1$ copies of $+1$ and s copies of -1; similarly, for $x_i = 0$, the user sends s copies of $+1$ and s copies of -1. By setting q to be sufficiently small (e.g., $q = O(1/\epsilon n)$), it can be shown that the error remains roughly the same as before. Furthermore, when s is sufficiently large (i.e., $O_\epsilon(\log n)$), we manage to show that this algorithm satisfies ϵ-DP$_{\text{shuffle}}$. While the exact reason for this pure-DP guarantee is rather technical, the general idea is similar to [15]: by making the "border" part of the support equal in probabilities in the two cases, we avoid the issues presented above. Furthermore, by making s sufficiently large, the input-dependent probability is "sufficiently inside" of the support that it usually does not completely dominate the contribution from the outer part.

Finally, note that V_{+1}, V_{-1} involves summation of many i.i.d. random variables $\sum_{i\in[n]} z_i^{\pm 1}$, $\sum_{i\in[n]} z_i^{+1}$, and $\sum_{i\in[n]} z_i^{-1}$. As observed in [18], it is convenient to use *infinitely divisible* distributions so that these sums have distributions that are independent of n, allowing for simpler calculations. We inherit this feature from their analysis.

2 Preliminaries

For a discrete distribution $\mathcal{D}$, let $f_\mathcal{D}$ denote its probability mass function (PMF). The *max-divergence* between distributions $\mathcal{D}_1, \mathcal{D}_2$ is defined as $d_\infty(\mathcal{D}_1 \| \mathcal{D}_2) := \max_{x \in \operatorname{supp}(\mathcal{D}_1)} \ln \frac{f_{\mathcal{D}_1}(x)}{f_{\mathcal{D}_2}(x)}$.

For two distributions $\mathcal{D}_1, \mathcal{D}_2$ over $\mathbb{Z}^d$, we write $\mathcal{D}_1 * \mathcal{D}_2$ to denote its *convolution*, i.e., the distribution of $z_1 + z_2$ where $z_1 \sim \mathcal{D}_1, z_2 \sim \mathcal{D}_2$ are independent. Moreover, let $(\mathcal{D})^{*n}$ denote the n-fold convolution of $\mathcal{D}$, i.e., the distribution of $z_1 + \cdots + z_n$ where $z_1, \dots, z_n \sim \mathcal{D}$ are independent. We write $\mathcal{D} \otimes \mathcal{D}'$ to denote the *product* distribution of $\mathcal{D}_1, \mathcal{D}_2$. Furthermore, we may write a value to denote the distribution all of whose probability mass is at that value (e.g., 0 stands for the probability distribution that is always equal to zero).

A distribution $\mathcal{D}$ is *infinitely divisible* iff, for every positive integer n, there exists a distribution $\mathcal{D}_{/n}$ such that $\mathcal{D} = (\mathcal{D}_{/n})^{*n}$. Two distributions we will use here (both supported on $\mathbb{Z}_{\geq 0}$) are:

- *Poisson Distribution* $\operatorname{Poi}(\lambda)$: This is the distribution whose PMF is $f_{\operatorname{Poi}(\lambda)}(k) = \lambda^k e^{-\lambda}/k!$. It satisfies $\operatorname{Poi}(\lambda)_{/n} = \operatorname{Poi}(\lambda/n)$.
- *Negative Binomial Distribution* $\operatorname{NB}(r,p)$: Its PMF is $f_{\operatorname{NB}(r,p)}(k) = \binom{k+r-1}{k} p^r (1-p)^k$. It satisfies $\operatorname{NB}(r,p)_{/n} = \operatorname{NB}(r/n, p)$.
  - *Geometric Distribution* $\operatorname{Geo}(p)$: A special case of the NB distribution is the geometric distribution $\operatorname{Geo}(p) = \operatorname{NB}(1,p)$, i.e., one with $f_{\operatorname{Geo}(p)}(k) = p(1-p)^k$.

Finally, we recall that the *discrete Laplace distribution* $\operatorname{DLap}(a)$ is a distribution supported on $\mathbb{Z}$ with PMF $f_{\operatorname{DLap}(a)}(x) \propto \exp(-a|x|)$. It is well-known that $\operatorname{DLap}(a)$ is the distribution of $z_1 - z_2$ where $z_1, z_2 \sim \operatorname{Geo}(1-\exp(-a))$ are independent. Furthermore, the variance of the discrete Laplace distribution is $\operatorname{Var}(\operatorname{DLap}(a)) = \frac{2e^{-a}}{(1-e^{-a})^2}$.

We will also use the following well-known lemma[2]:

▶ **Lemma 5.** *For any distributions $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$ over $\mathbb{Z}^d$, $d_\infty(\mathcal{D}_1 * \mathcal{D}_3 \| \mathcal{D}_2 * \mathcal{D}_3) \leq d_\infty(\mathcal{D}_1 \| \mathcal{D}_2)$.*

[2] This can be viewed as a special case of the post-processing property of DP where the post-processing function is adding a random variable drawn from $\mathcal{D}_3$. Another way to see that this holds is to simply observe that, for any $y \in \operatorname{supp}(\mathcal{D}_1 * \mathcal{D}_2)$, we have $f_{\mathcal{D}_1 * \mathcal{D}_3}(y) = \sum_{z \in \operatorname{supp}(\mathcal{D}_3)} f_{\mathcal{D}_3}(z) \cdot f_{\mathcal{D}_1}(y-z) \leq \sum_{z \in \operatorname{supp}(\mathcal{D}_3)} f_{\mathcal{D}_3}(z) \cdot \left(e^{d_\infty(\mathcal{D}_1 \| \mathcal{D}_2)} \cdot f_{\mathcal{D}_2}(y-z)\right) = e^{d_\infty(\mathcal{D}_1 \| \mathcal{D}_2)} f_{\mathcal{D}_2 * \mathcal{D}_3}(y)$.

3 Counting Protocol

In this section, we will describe a pure-DP$_{\text{shuffle}}$ algorithm for counting, which is our main result.

▶ **Theorem 6.** *For any positive real numbers $\epsilon \leq O(1)$ and $\rho \in (0, 1/2]$, there is a (non-interactive) ϵ-DP$_{\text{shuffle}}$ protocol for binary summation that has MSE at most $(1 + \rho) \cdot \mathrm{Var}(\mathrm{DLap}(\epsilon))$ where each user sends $O\left(\frac{\log(n/\rho)}{\epsilon\rho}\right)$ messages in expectation and each message consists of a single bit.*

By setting ρ arbitrarily close to zero, we can get the mean-square error (MSE) to be arbitrarily close to that of the discrete Laplace mechanism, which is known to be (asymptotically) optimal in the central model [21, 14]. We can get this guarantee for other type of errors, e.g., ℓ_1-error (aka expected absolute error) as well, but for ease of presentation, we only focus on the MSE.

Note that Theorem 6 implies Theorem 2 by simply setting ρ to be a positive constant (say, 0.5).

3.1 Algorithm

In this section we present and analyze our main algorithm for counting (aka binary summation). To begin, we will set our parameters as follows.

▶ **Condition 7.** *Let $\lambda, \epsilon', \epsilon, q \in \mathbb{R}_{>0}$ and $s \in \mathbb{Z}_{>0}$. Suppose that the following conditions hold:*
- $\epsilon' < \epsilon$,
- $s \geq 2 \ln \left(\frac{1}{(e^{\epsilon}-1)q}\right) / (\epsilon - \epsilon')$,
- $\lambda \geq \frac{e^{\epsilon-\epsilon'}}{e^{(\epsilon-\epsilon')/2}-1} \cdot s$.

We now define the following distributions:
- $\mathcal{D}^{\text{noise}} = \mathrm{Geo}(1 - e^{-\epsilon'})$.
- $\mathcal{D}^{\text{flood}} = \mathrm{Poi}(\lambda)$.
- For $x \in \{0, 1\}$, $\mathcal{D}^{\text{input},x}$ supported on $\mathbb{Z}^2_{\geq 0}$ is defined as

$$\begin{aligned}\mathcal{D}^{\text{input},x}((s + x, s)) &= 1 - q,\\ \mathcal{D}^{\text{input},x}((0, 0)) &= q.\end{aligned}$$

Algorithm 1 contains the formal description of the randomizer and Algorithm 2 contains the description of the analyzer. As mentioned earlier, our algorithm is the same as that of [18], except in the first step (Line 2). In their work, the protocol always sends a single +1 if $x_i = 1$ and nothing otherwise in this step. Instead, we randomize this step by always sending nothing with a certain probability. With the remaining probability, instead of sending a single +1 for $x_i = 1$, we send $s + 1$ copies of +1 and s copies of −1 (similarly, we send s copies of +1 and s copies of −1 in the case $x_i = 0$).

4 Analysis of the Protocol

In this section we analyze the privacy, utility, and communication guarantees of our counting protocol. Throughout the remainder of this section, we assume the distributions and parameters are set as in Condition 7; for brevity, we will not state this assumption in our privacy analysis.

Algorithm 1 Counting Randomizer.

```
1: procedure CorrNoiseRandomizer$_n(x_i)$
2:     Sample $(y_i^{+1}, y_i^{-1}) \sim \mathcal{D}^{\mathrm{input},x_i}$
3:     Sample $z_i^{+1}, z_i^{-1} \sim \mathcal{D}^{\mathrm{noise}}_{/n}$
4:     Sample $z_i^{\pm 1} \sim \mathcal{D}^{\mathrm{flood}}_{/n}$
5:     Send $y_i^{+1} + z_i^{+1} + z_i^{\pm 1}$ copies of $+1$, and $y_i^{-1} + z_i^{-1} + z_i^{\pm 1}$ copies of $-1$
```

Algorithm 2 Counting Analyzer.

```
1: procedure CorrNoiseAnalyzer$_q$
2:     $R \leftarrow$ multiset of messages received
3:     return $\frac{1}{1-q}\left(\sum_{y \in R} y\right)$
```

4.1 Privacy Analysis

▶ **Lemma 8** (Main Privacy Guarantee). *CorrNoiseRandomizer satisfies* ϵ-DP$_{\mathrm{shuffle}}$.

To prove the above, we need the following technical lemmas regarding $\mathcal{D}^{\mathrm{noise}}, \mathcal{D}^{\mathrm{flood}}$.

▶ **Lemma 9.** *For every* $i \in \mathbb{Z}$, $f_{\mathcal{D}^{\mathrm{noise}}}(i-1) \le e^{\epsilon'} f_{\mathcal{D}^{\mathrm{noise}}}(i)$

Proof. This immediately follows from the PMF definition of $\mathcal{D}^{\mathrm{noise}} = \mathrm{Geo}(1-e^{\epsilon'})$. ◀

▶ **Lemma 10.** *For every* $i \in \mathbb{Z}$, $(e^{\epsilon}-1)q \cdot f_{\mathcal{D}^{\mathrm{flood}}}(i+s) + e^{\epsilon-\epsilon'} f_{\mathcal{D}^{\mathrm{flood}}}(i-1) \ge f_{\mathcal{D}^{\mathrm{flood}}}(i)$.

Proof. If $e^{\epsilon-\epsilon'} f_{\mathcal{D}^{\mathrm{flood}}}(i-1) \ge f_{\mathcal{D}^{\mathrm{flood}}}(i)$, then the statement is clearly true. Otherwise, we have $f_{\mathcal{D}^{\mathrm{flood}}}(i) > 0$ (i.e., $i \ge 0$) and $e^{\epsilon'-\epsilon} > \frac{f_{\mathcal{D}^{\mathrm{flood}}}(i-1)}{f_{\mathcal{D}^{\mathrm{flood}}}(i)} = \frac{i}{\lambda}$, which implies

$$0 \le i \le e^{\epsilon'-\epsilon}\lambda. \tag{1}$$

We can then bound $\frac{f_{\mathcal{D}^{\mathrm{flood}}}(i+s)}{f_{\mathcal{D}^{\mathrm{flood}}}(i)}$ as

$$\begin{aligned}\frac{f_{\mathcal{D}^{\mathrm{flood}}}(i+s)}{f_{\mathcal{D}^{\mathrm{flood}}}(i)} &= \frac{\lambda^s}{(i+1)\cdots(i+s)} \ge \frac{\lambda^s}{(i+s)^s} \\ &\overset{(1)}{\ge} \left(\frac{\lambda}{e^{\epsilon'-\epsilon}\lambda + s}\right)^s \ge \left(\frac{\lambda}{e^{(\epsilon'-\epsilon)/2}\lambda}\right)^s \\ &\ge \frac{1}{(e^{\epsilon}-1)q},\end{aligned}$$

where the last two inequalities follow from our assumptions on λ and s respectively (Condition 7). Thus, in this case, we also have $(e^{\epsilon}-1)q \cdot f_{\mathcal{D}^{\mathrm{flood}}}(i+s) + e^{\epsilon-\epsilon'} f_{\mathcal{D}^{\mathrm{flood}}}(i-1) \ge f_{\mathcal{D}^{\mathrm{flood}}}(i)$ as desired. ◀

We are now ready to prove the privacy guarantee (Lemma 8).

Proof of Lemma 8. For any input dataset X. Let $V(X) = (V_{+1}, V_{-1})$ denote the distribution of the view of shuffler, where V_{+1} and V_{-1} denotes the number of $+1$ messages and the number of -1 messages respectively.

Consider two neighboring datasets $X = (x_1, \dots, x_n)$ and $X' = (x'_1, \dots, x'_n)$. Assume w.l.o.g. that they differ in the first coordinate and $x_1 = 1, x'_1 = 0$ and $x'_2 = x_2, \dots, x'_n = x_n$. To prove that CorrNoiseRandomizer satisfies ϵ-DP$_{\mathrm{shuffle}}$, we need to prove that $d_\infty(V(X) \| V(X')) \le \epsilon$ and $d_\infty(V(X') \| V(X)) \le \epsilon$.

Let $\mathcal{F}$ denote the distribution on $\mathbb{Z}^2$ of (X, X) where $X \sim \mathcal{D}^{\text{flood}}$. Observe that

$$\begin{aligned} V(X) = \mathcal{D}^{\text{input},1} &* \mathcal{D}^{\text{input},x_2} * \cdots * \mathcal{D}^{\text{input},x_n} \\ &* \mathcal{F} * (\mathcal{D}^{\text{noise}} \otimes 0) * (0 \otimes \mathcal{D}^{\text{noise}}), \end{aligned}$$

and

$$\begin{aligned} V(X') = \mathcal{D}^{\text{input},0} &* \mathcal{D}^{\text{input},x_2} * \cdots * \mathcal{D}^{\text{input},x_n} \\ &* \mathcal{F} * (\mathcal{D}^{\text{noise}} \otimes 0) * (0 \otimes \mathcal{D}^{\text{noise}}). \end{aligned}$$

Bounding $d_\infty(V(X) \| V(X'))$

From Lemma 5, we have

$$\begin{aligned} &d_\infty(V(X)\|V(X')) \\ &\leq d_\infty(\mathcal{D}^{\text{input},1} * (\mathcal{D}^{\text{noise}} \otimes 0) \| \mathcal{D}^{\text{input},0} * (\mathcal{D}^{\text{noise}} \otimes 0)). \end{aligned}$$

For any $i, j \in \mathbb{Z}$, we have

$$\begin{aligned} &f_{\mathcal{D}^{\text{input},1} * \mathcal{D}^{\text{noise}} \otimes 0}(i,j) \\ &= q \cdot f_{\mathcal{D}^{\text{noise}}}(i)\mathbf{1}[j=0] + (1-q) \cdot f_{\mathcal{D}^{\text{noise}}}(i-s-1)\mathbf{1}[j=s] \\ &\leq q \cdot f_{\mathcal{D}^{\text{noise}}}(i)\mathbf{1}[j=0] + (1-q) \cdot e^{\epsilon'} f_{\mathcal{D}^{\text{noise}}}(i-s)\mathbf{1}[j=s] \\ &\leq e^{\epsilon}\left(q \cdot f_{\mathcal{D}^{\text{noise}}}(i)\mathbf{1}[j=0] + (1-q) \cdot f_{\mathcal{D}^{\text{noise}}}(i-s)\mathbf{1}[j=s]\right) \\ &= e^{\epsilon} \cdot f_{\mathcal{D}^{\text{input},0} * \mathcal{D}^{\text{noise}} \otimes 0}(i,j), \end{aligned}$$

where the first inequality follows from Lemma 9 and the second inequality follows from Condition 7. Combining the above inequalities, we have $d_\infty(V(X)\|V(X')) \leq \epsilon$ as desired.

Bounding $d_\infty(V(X') \| V(X))$

Again, from Lemma 5, we have

$$\begin{aligned} &d_\infty(V(X')\|V(X)) \\ &\leq d_\infty\left(\mathcal{D}^{\text{input},0} * \mathcal{F} * (0 \times \mathcal{D}^{\text{noise}})\right. \\ &\qquad\qquad \left.\| \mathcal{D}^{\text{input},1} * \mathcal{F} * (0 \times \mathcal{D}^{\text{noise}})\right). \end{aligned}$$

For any $i, j \in \mathbb{Z}$, we have

$$\begin{aligned} &f_{\mathcal{D}^{\text{input},0} * \mathcal{F} * (0 \times \mathcal{D}^{\text{noise}})}(i,j) \\ &= f_{\mathcal{D}^{\text{input},0} * \mathcal{F}}(i,i) \cdot f_{\mathcal{D}^{\text{noise}}}(j-i) \\ &= (q \cdot f_{\mathcal{D}^{\text{flood}}}(i) + (1-q) \cdot f_{\mathcal{D}^{\text{flood}}}(i-s)) \cdot f_{\mathcal{D}^{\text{noise}}}(j-i) \\ &\leq e^{\epsilon}\left(q \cdot f_{\mathcal{D}^{\text{flood}}}(i) + (1-q) \cdot e^{-\epsilon'} f_{\mathcal{D}^{\text{flood}}}(i-s-1)\right) \\ &\qquad\qquad \cdot f_{\mathcal{D}^{\text{noise}}}(j-i) \\ &\leq e^{\epsilon}\left(q \cdot f_{\mathcal{D}^{\text{flood}}}(i) \cdot f_{\mathcal{D}^{\text{noise}}}(j-i)\right. \\ &\qquad\qquad \left.+(1-q) \cdot f_{\mathcal{D}^{\text{flood}}}(i-s-1) \cdot f_{\mathcal{D}^{\text{noise}}}(j-i+1)\right) \\ &= e^{\epsilon} f_{\mathcal{D}^{\text{input},1} * \mathcal{F} * (0 \times \mathcal{D}^{\text{noise}})}(i,j), \end{aligned}$$

where the first inequality follows from Lemma 10 and the second inequality follows from Lemma 9. Combining the above two inequalities, we have $d_\infty(V(X')\|V(X)) \leq \epsilon$, concluding our proof. ◀

4.2 Utility Analysis

We next analyze the MSE of the output estimate.

▶ **Lemma 11.** *The estimator from Algorithm 2 is unbiased and its MSE is at most*

$$\left(\frac{1}{1-q}\right)^2 \cdot (qn + \mathrm{Var}(\mathrm{DLap}(\epsilon'))) .$$

Proof. Notice that the output estimate is equal to

$$\frac{1}{1-q}\left(\sum_{i\in[n]}(y_i^{+1} - y_i^{-1} + z_i^{+1} - z_i^{-1})\right) = \frac{1}{1-q}\left(\sum_{i\in[n]}(y_i^{+1} - y_i^{-1}) + Z\right),$$

where $Z \sim \mathrm{DLap}(\epsilon')$. It is also simple to verify that $\mathbb{E}[y_i^{+1} - y_i^{-1}] = (1-q)x_i$. Thus, the estimator is unbiased as desired. Its MSE is equal to

$$\begin{aligned}&\mathrm{Var}\left(\frac{1}{1-q}\left(\sum_{i\in[n]}(y_i^{+1} - y_i^{-1}) + Z\right)\right)\\ &= \left(\frac{1}{1-q}\right)^2\left(\sum_{i\in[n]}\mathrm{Var}(y_i^{+1} - y_i^{-1}) + \mathrm{Var}(\mathrm{DLap}(\epsilon'))\right).\end{aligned}$$

Next, notice that, if $x_i = 0$, then $y_i^{+1} - y_i^{-1} - x_i = 0$ always. Otherwise, if $x_i = 1$, then $y_i^{+1} - y_i^{-1} - x_i = 0$ with probability $1-q$ and $y_i^{+1} - y_i^{-1} - x_i = 1$ with probability q. As a result, we have $\mathrm{Var}(y_i^{+1} - y_i^{-1}) \leq q$. Plugging this into the above inequality yields the claimed bound on the MSE. ◀

4.3 Communication Analysis

The expected number of bits send by the users can be easily computed as follows.

▶ **Lemma 12.** *The expected number of messages sent by each user is at most* $2s + 1 + \frac{\lambda}{n} + O\left(\frac{1}{\epsilon' n}\right)$.

Proof. The expected number of bits sent per user is

$$\begin{aligned}&\mathbb{E}[y_i^{+1} + y_i^{-1}] + \mathbb{E}[z_i^{+1} + z_i^{-1}] + 2\mathbb{E}[z_i^{\pm 1}]\\ &\leq (2s+1) + \frac{2\mathbb{E}[\mathcal{D}^{\mathrm{noise}}]}{n} + \frac{\mathbb{E}[\mathcal{D}^{\mathrm{flood}}]}{n}\\ &= 2s + 1 + O\left(\frac{1}{\epsilon' n}\right) + \frac{\lambda}{n}.\end{aligned}$$ ◀

4.4 Putting Things Together: Proof of Theorem 6

Finally, we are ready to prove Theorem 6 by plugging in appropriate parameters and invoke the previous lemmas.

Proof of Theorem 6. We start by picking $\epsilon' = \epsilon - 0.01\rho \cdot \min\{\epsilon, 1\}$. For this choice of ϵ', we have

$$\frac{\mathrm{Var}(\mathrm{DLap}(\epsilon'))}{\mathrm{Var}(\mathrm{DLap}(\epsilon))} = \frac{\frac{2e^{-\epsilon'}}{(1-e^{-\epsilon'})^2}}{\frac{2e^{-\epsilon}}{(1-e^{-\epsilon})^2}}$$

$$\begin{aligned}&\leq 1+\frac{(e^{\epsilon-\epsilon'}-1)(1+e^{-\epsilon'})}{1-e^{-\epsilon'}}\\&\leq 1+\frac{3(\epsilon-\epsilon')\cdot 2}{\epsilon'}\leq 1+0.1\rho.\end{aligned}$$

Then, picking

$$\begin{aligned}&q=0.1\rho\cdot\min\left\{\frac{\mathrm{Var}(\mathrm{DLap}(\epsilon))}{n},1\right\}=O\left(\frac{\rho}{\epsilon^2 n}\right),\\&s\geq 2\ln\left(\frac{1}{(e^{\epsilon}-1)q}\right)/(\epsilon-\epsilon')=O\left(\frac{\log(n/\rho)}{\epsilon\rho}\right),\\&\lambda\geq\frac{e^{\epsilon-\epsilon'}}{e^{(\epsilon-\epsilon')/2}-1}\cdot s=O\left(\frac{\log(n/\rho)}{\epsilon^2\rho}\right),\end{aligned}$$

and applying Lemma 8, Lemma 11, and Lemma 12 immediately yields Theorem 6. (Note that we may assume that $\epsilon\geq 1/n$; otherwise we can just output zero. Under this assumption, we have $\lambda/n\leq O\left(\frac{\log(n/\rho)}{\epsilon\rho}\right)$ as desired for the communication complexity claim.) ◀

5 Non-Asymptotic Comparisons with Previous Work

In this section, we provide concrete non-asymptotic comparisons between our binary summation protocol and those from previous work [15, 9] for various population sizes n and privacy parameters ϵ. As we explain in more detail below, our results demonstrate that our protocol is much more practical than those of previous works.

First, we find that the parameters in the protocol of [15] are impractical; in fact, for $n\leq 800,000$, their protocol is *undefined* unless $\epsilon<0.01$.[3] Furthermore, even in the regime that it is well-defined, their expected communication complexity is provably at least 1000x ours and their root-mean-square error (RMSE) is probably at least 100x ours. Hence, we only focus on the comparison between our algorithm and that of [9].

Parameter Setting

For both our algorithm and that of [9], one can achieve RMSE arbitrarily close to that of the ϵ-DP discrete Laplace mechanism in the central model. (See the parameter ρ in Theorem 6.) To reduce the parameter space for comparison, we set the parameters so that the RMSE of these protocols is within 10% of the discrete Laplace mechanism. Given this error target, we simply use the formulae from Lemma 11 and Condition 7 to optimize for ϵ', q, λ that minimizes the expected communication (according to Lemma 12); we use `scipy` package for this optimization. For [9]'s algorithm, we set the parameter in an optimistic manner so that we underestimate the communication required in their protocol[4].

[3] This is due to the fact that they require their parameter $p=\frac{100e^{100\epsilon}\log(1/(1-e^{0.1\epsilon}))}{n(1-e^{0.1\epsilon})}$ to be less than one. Of course, one can run their protocol at a smaller ϵ but this increases the communication and error even further.

[4] Namely, we only set p in their protocol to $0.5/n$ and do not account for the error from the p-probability event that the input is randomized. (In their analysis, p should actually be set to $\hat{q}/n$ where $\hat{q}\leq O(1/n)$ is yet another small parameter.)

Expected Communication Comparison

We provide a comparison of the expected number of messages sent when fixing $\epsilon = 1$ and varying n from 1 to 1000 in Figure 1(ii). To summarize, the number of messages of their protocols grows very quickly and exceed 10 million even when $n = 65$! This agrees with theory, which suggests that their communication complexity grows with $\widetilde{O}(n^2)$. Meanwhile, our protocol has expected number of messages sent less than 600 for the entire range of $10 < n \leq 100$, again agreeing with the theory that our communication grows only with $O\left(\frac{\log n}{\epsilon}\right)$. Moreover, the expected number of messages of our protocol is less than that of theirs except when $n = 1$. For clarity, we also provide our protocol's expected number of messages in Figure 2(i) for the small n case $(1 \leq n \leq 10^3)$ and in Figure 2(ii) for the large n case $(10^3 \leq n \leq 10^6)$. These plots show that the expected number of messages is large for very small $n \leq 5$, in which regime the expected number of messages decrease as n increases. This regime corresponds to the regime where the communication due to the Poisson noise dominates. Once the expected number of messages bottoms out, it increases slowly, as suggested by our theoretical analysis. Finally, we suspect that the curve is not completely smooth since `scipy.optimize.minimize_scalar` does not always find the optimum[5].

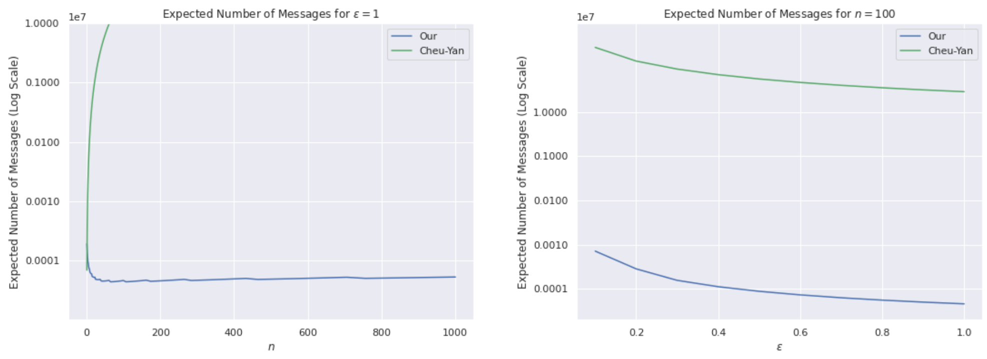


Figure 1 Comparison between the expected number of messages sent in our protocol and in Cheu–Yan protocol when (i) $\epsilon = 1$ and varying n, (ii) $n = 100$ and varying ϵ. (Note that the y-axis is in log-scale.)

Next, we fix $n = 100$ and vary ϵ. The resulting expected communication is presented in Figure 1(i). Again, there is a very large (> 10000x) gap between our expected communication and theirs. Furthermore, these increase roughly as $1/\epsilon$, as predicted by theory.

Finally, we note that, while we perform comparisons for binary summation, the comparisons would be similar for histogram as well. This is because all protocols are adapted to the histogram problem by simply running the binary protocol separately for each bucket; thus, the expected communication simply increases by a factor of B.

6 Conclusions and Open Questions

In this work, we have provided pure-DP$_{\text{shuffle}}$ algorithms that achieve nearly optimal errors for bit summation, real summation, and histogram while significantly improving on the communication complexity compared to the state-of-the-art. Despite this, there are still a number of interesting open questions, some of which we highlight below.

[5] In particular, the value of s is discrete in our optimization problem, making it harder to optimize for

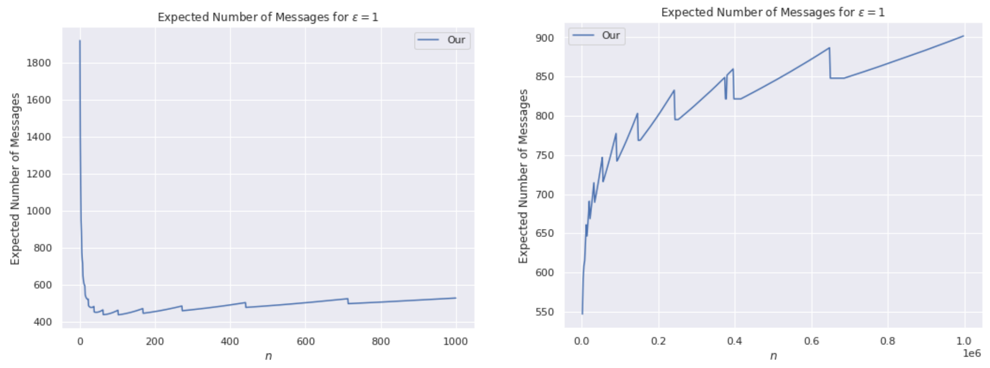


Figure 2 The expected number of messages sent in our protocol when $\epsilon = 1$ for (i) $1 \leq n \leq 10^3$, (ii) $10^3 \leq n \leq 10^6$.

- **Protocol with a bounded number of messages.** As mentioned briefly in Section 1.1, our protocol can result in an arbitrarily large number of messages per user, although the expected number is quite small. (In fact, the distribution of the number of messages enjoys a strong exponential tail bound.) Is it possible to design a pure-DP$_{\text{shuffle}}$ protocol where the maximum number of messages is $O\left(\frac{\log n}{\epsilon}\right)$ for binary summation?
  For this question, we note that a rather natural approach is to modify our protocol to make its number of messages bounded. Namely, we replace $\mathcal{D}^{\text{noise}}_{/n}$ and $\mathcal{D}^{\text{flood}}_{/n}$ by a truncated version of their respective distributions. It turns out that the latter is relatively simple (e.g., even replacing it with a Bernoulli distribution also works) because we only require a mild condition in Lemma 10 to hold. On the other hand, for the former, we are using Lemma 9, which only holds for unbounded distributions. We would like to stress that we do not know whether replacing $\mathcal{D}^{\text{noise}}_{/n}$ with a truncated version of the negative binomial distribution with a "symmetrized" the input dependent part[6] violates pure-DP; however, we do not know how to prove that it satisfies pure-DP either, as the probability mass function of their convolutions become somewhat unwieldy.
- **Lower bounds on the expected number of messages.** Recall that the communication lower bound from [15] only applies to the maximum number of messages sent. Is it possible to prove a communication lower bound on the *expected* number of messages (even if the maximum number of messages is unbounded)? We note that the techniques from [15] does not apply.
- **More practical protocols.** In Section 5, we demonstrated that, while the parameters from previous work [15, 9] are completely impractical, our result is moderately practical. However, our pure-DP protocol still requires (expected) communication overhead of 500–1000x compared to the non-private protocol. Meanwhile, the approximate-DP protocol of [18] achieves communication overhead of only $1 + o(1)$ (assuming that δ is not too small). Due to this, there is still a large gap between the practicality of pure-DP and approximate-DP protocols. While the lower bound from [15] mentioned in the previous bullet point strongly suggests that it might not be possible to reduce the communication required for pure-DP all the way to that of approximate-DP, it remains an important

[6] This means that w.p. q we output s copies of both $+1$ and -1 messages, for both $x_i = 0$ and $x_i = 1$ cases. Without this change, the supports of the two cases are not the same and thus it obviously violates pure-DP.

question to make pure-DP protocol more practical. For example, can we reduce the communication by a factor of 10 while achieving similar utility and privacy guarantees as in this work?

- **Histogram protocol for large B.** Our protocol has communication complexity that grows linearly with B, which is impractical when B is large. Can we get protocol for histogram whose communication is $O_\epsilon\left((\log n)^{O(1)}\right)$ for $B = O(n)$ (while achieving nearly optimal errors)? For approximate-DP$_{\text{shuffle}}$, a histogram protocol with expected communication of $1 + O_\epsilon\left(\frac{B(\log(n/\delta)^{O(1)})}{n}\right)$ is known [18]. It would be interesting to understand if such a protocol exists in the pure-DP$_{\text{shuffle}}$ setting.
- **Generic DP$_{\text{local}}$ $\Rightarrow$ DP$_{\text{shuffle}}$ transformation for pure-DP?** More generally, despite the rich literature on the shuffle model, most work has focused attention on approximate-DP$_{\text{shuffle}}$. It would be interesting to expand the existing study to pure-DP$_{\text{shuffle}}$ as well. In our opinion, a main barrier in doing so is that the so-called *amplification-by-shuffling* phenomenon does not apply to pure-DP. Recall that the amplification-by-shuffling theorem [12] roughly states that, if we take any ϵ-DP$_{\text{local}}$ algorithm and runs it in the shuffle model, then it immediately becomes (ϵ', δ')-DP$_{\text{shuffle}}$ where $\epsilon' \ll \epsilon$ for any non-too-small $\delta > 0$. This means that any DP$_{\text{local}}$ algorithm translates to approximate-DP$_{\text{shuffle}}$ algorithm with improved privacy; this allows the design of approximate-DP$_{\text{shuffle}}$ algorithms to tap into the vast literature of DP$_{\text{local}}$. Unfortunately, it is known that the amplification-by-shuffling theorem does not hold when we want pure-DP$_{\text{shuffle}}$; see [15] for an explanation. A natural question here is thus whether we can take any ϵ-DP$_{\text{local}}$ algorithm, modify it slightly (while preserving utility) and make it ϵ'-DP$_{\text{shuffle}}$ algorithm for $\epsilon' \ll \epsilon$. Such a transformation would enable a sort of "amplification-by-shuffling" in the pure-DP$_{\text{shuffle}}$ regime as well.

References

1 Victor Balcer and Albert Cheu. Separating local & shuffled differential privacy via histograms. In *ITC*, pages 1:1–1:14, 2020.

2 Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. The privacy blanket of the shuffle model. In *CRYPTO*, pages 638–667, 2019.

3 Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. Private summation in the multi-message shuffle model. In *CCS*, pages 657–676, 2020.

4 Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In *CRYPTO*, pages 451–468, 2008.

5 Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnés, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *SOSP*, pages 441–459, 2017.

6 Mark Bun, Jelani Nelson, and Uri Stemmer. Heavy hitters and the structure of local privacy. *TALG*, 15(4):1–40, 2019.

7 T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Optimal lower bound for differentially private multi-party aggregation. In *ESA*, pages 277–288, 2012.

8 Albert Cheu, Adam D. Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In *EUROCRYPT*, pages 375–403, 2019.

9 Albert Cheu and Chao Yan. Pure differential privacy from secure intermediaries. In *TPDP*, 2022.

10 Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, pages 486–503, 2006.

11 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

12 Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *SODA*, pages 2468–2479, 2019.

13 Alexandre Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In *PODS*, pages 211–222, 2003.

14 Quan Geng and Pramod Viswanath. The optimal noise-adding mechanism in differential privacy. *IEEE Trans. Inf. Theory*, 62(2):925–951, 2016.

15 Badih Ghazi, Noah Golowich, Ravi Kumar, Pasin Manurangsi, Rasmus Pagh, and Ameya Velingker. Pure differentially private summation from anonymous messages. In *ITC*, pages 15:1–15:23, 2020.

16 Badih Ghazi, Noah Golowich, Ravi Kumar, Rasmus Pagh, and Ameya Velingker. On the power of multiple anonymous messages: Frequency estimation and selection in the shuffle model of differential privacy. In *EUROCRYPT*, pages 463–488, 2021.

17 Badih Ghazi, Ravi Kumar, and Pasin Manurangsi. User-level differentially private learning via correlated sampling. In *NeurIPS*, pages 20172–20184, 2021.

18 Badih Ghazi, Ravi Kumar, Pasin Manurangsi, and Rasmus Pagh. Private counting from anonymous messages: Near-optimal accuracy with vanishing communication overhead. In *ICML*, pages 3505–3514, 2020.

19 Badih Ghazi, Ravi Kumar, Pasin Manurangsi, Rasmus Pagh, and Amer Sinha. Differentially private aggregation in the shuffle model: Almost central accuracy in almost a single message. In *ICML*, pages 3692–3701, 2021.

20 Badih Ghazi, Pasin Manurangsi, Rasmus Pagh, and Ameya Velingker. Private aggregation from fewer anonymous messages. In *EUROCRYPT*, pages 798–827, 2020.

21 Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. In *STOC*, pages 351–360, 2009.

22 Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Rashkodnikova, and Adam Smith. What can we learn privately? In *FOCS*, pages 531–540, 2008.

23 Thomas Steinke and Jonathan Ullman. Between pure and approximate differential privacy. *Journal of Privacy and Confidentiality*, 7(2):3–22, 2016.

On the Power of Adaptivity for Function Inversion

Karthik Gajulapalli
Georgetown University, Washington, DC, USA

Alexander Golovnev
Georgetown University, Washington, DC, USA

Samuel King
Georgetown University, Washington, DC, USA

Abstract

We study the problem of *function inversion with preprocessing* where, given a function $f : [N] \to [N]$ and a point y in its image, the goal is to find an x such that $f(x) = y$ using at most T oracle queries to f and S bits of preprocessed advice that depend on f.

The seminal work of Corrigan-Gibbs and Kogan [TCC 2019] initiated a line of research that shows many exciting connections between the non-adaptive setting of this problem and other areas of theoretical computer science. Specifically, they introduced a very weak class of algorithms (strongly non-adaptive) where the points queried by the oracle depend only on the inversion point y, and are independent of the answers to the previous queries and the S bits of advice. They showed that proving even mild lower bounds on strongly non-adaptive algorithms for function inversion would imply a breakthrough result in circuit complexity.

We prove that every strongly non-adaptive algorithm for function inversion (and even for its special case of permutation inversion) must have $ST = \Omega(N \log(N) \log(T))$. This gives the first improvement to the long-standing lower bound of $ST = \Omega(N \log N)$ due to Yao [STOC 90]. As a corollary, we conclude the first separation between strongly non-adaptive and adaptive algorithms for permutation inversion, where the adaptive algorithm by Hellman [TOIT 80] achieves the trade-off $ST = O(N \log N)$.

Additionally, we show equivalence between lower bounds for strongly non-adaptive data structures and the one-way communication complexity of certain partial functions. As an example, we recover our lower bound on function inversion in the communication complexity framework.

2012 ACM Subject Classification Theory of computation → Cryptographic primitives

Keywords and phrases Function Inversion, Non-Adaptive lower bounds, Communication Complexity

Digital Object Identifier 10.4230/LIPIcs.ITC.2024.5

Related Version *ECCC Version*: `https://eccc.weizmann.ac.il/report/2024/054/`

Funding This research is supported by the National Science Foundation CAREER award (grant CCF-2338730).

Acknowledgements We would like to thank Spencer Peters for fruitful discussions on this topic. We are also grateful to the anonymous reviewers for their helpful comments.

1 Introduction

We study the fundamental problem of *function inversion* where, given oracle access to a function $f : [N] \to [N]$ and a point y in the image of f, the goal is to find some x such that $f(x) = y$.

Clearly, to work for all functions, this would require any algorithm to make at least $N - 1$ oracle calls. However, to make the problem more interesting, we consider a pair of algorithms $(\mathcal{P}, \mathcal{A})$ that work in two phases. In the first phase, using unlimited computational power, the *pre-processing* algorithm $\mathcal{P}$ is allowed to analyze the function f and write down S bits of *advice* $\sigma \in \{0,1\}^S$. Then in the second phase, called the *online* phase, the algorithm

$\mathcal{A}$[1], given inputs y and σ and at most T oracle queries to f, is required to output x such that $f(x) = y$. We informally refer to S and T as *space* and *time*, and the goal is to find algorithms $(\mathcal{P}, \mathcal{A})$ for function inversion that minimize S and T. Note that the problem is trivial when $S = N \log N$ or $T = N$. We are interested in the trade-offs between time and space when in between these two cases.

This model has received a lot of attention, especially for its applications to cryptanalysis [2, 3, 27, 25], cryptography [18, 14, 15, 31, 28, 11, 12, 8, 9, 17], circuit and data structure lower bounds [32, 10, 13], algorithms [22, 16], information theory [13], and most recently even meta-complexity [23, 20].

Function inversion and *permutation inversion*, a special case of function inversion where f is a permutation, were initially studied by Hellman [18]. Hellman constructed an elegant algorithm that inverts any permutation when $ST = \Omega(N \log N)$. Later Yao [32] showed that this algorithm was optimal by proving a tight lower bound of $ST = \Omega(N \log N)$ for permutation inversion (assuming $S = \Omega(\log N)$). For function inversion, Hellman gave an algorithm that inverts a random function when $S^2 T = \widetilde{\Omega}(N^2)$.[2] Fiat and Naor [14] extended Hellman's construction, giving an algorithm that inverts any function when $S^3 T = \widetilde{\Omega}(N^3)$.

One key facet of all the upper bounds mentioned above is that the queries made to f are highly adaptive; i.e., deciding which point $\mathcal{A}$ is going to query next depends on the inversion point y, the advice string σ, and the values of the points queried before. A long-standing open question has been to see if any of the upper bounds could be made non-adaptive. This question was extensively studied in [10], and they introduced the notion of strongly non-adaptive algorithms where the points queried by $\mathcal{A}$ are a fixed set depending only on the inversion point y. This makes the model much weaker compared to even the standard non-adaptive (weakly non-adaptive) setting where the fixed set of points queried by $\mathcal{A}$ is allowed to depend on the inversion point y and the advice string σ. Upper bounds for non-adaptive algorithms would be really useful, as they would lead to efficient parallelisation. Perhaps even more interestingly, lower bounds even in this very weak model would already imply circuit and communication lower bounds [10] and data structure lower bounds [10, 16, 13].

Indeed, as shown in [10], a lower bound of $S = \omega(N \log N / \log \log N)$ when $T = N^{\varepsilon}$ would imply a circuit lower bound against Boolean circuits of linear size and logarithmic depth, and thus resolve a long-standing open question due to Valiant [29]. A similar argument shows that even a lower bound of $S = \omega(N \log N / (\log \log (T / \log N)))$ for any $T = \Omega(\log N)$ would imply a super-linear circuit lower bound for series-parallel circuits [29, 4, 30].

The only known strongly non-adaptive algorithm is the trivial one where the pre-processing algorithm stores the value of f at $S/\log N$ points as advice, and the online algorithm queries the remaining $N - S/\log N$ points, giving $S/\log N + T = N$. On the other hand, the best known lower bound for the non-adaptive setting is still $ST \geq \Omega(N \log N)$ obtained by Yao's compression argument [32] that works even for adaptive algorithms. Hence, it might still be conceivable that the algorithm by Hellman can be made non-adaptive, which leads us to the natural question:

Are non-adaptive algorithms for permutation inversion as efficient as adaptive algorithms?

[1] We will refer to $\mathcal{A}$ as the "online" algorithm, referring to its phase, even though it does not process its input in a serial fashion as is typical for what are called "online" algorithms.

[2] The notation $\widetilde{\Omega}(\cdot)$ and $\widetilde{O}(\cdot)$ suppresses factors polynomial in $\log N$.

1.1 Our Results

We answer this question in the negative by showing a lower bound of $ST = \Omega(N \log(N) \log(T))$ for any strongly non-adaptive algorithm for permutation inversion (and, thus, for the more general problem of function inversion).

▶ **Theorem 1.** *Every strongly non-adaptive algorithm that solves permutation inversion with S bits of preprocessing and $T \leq N/5$ queries must have*

$$S = \Omega\left(\frac{N \log(N) \log(T)}{T}\right) .$$

Since permutation inversion can be solved adaptively when $ST = O(N \log N)$ [18], Theorem 1 gives us the first separation between adaptive and strongly non-adaptive algorithms for permutation inversion for every super-constant T. (No separation is possible for constant T as in this case the problem is maximally hard, $S = \Omega(N \log N)$, even in the adaptive setting.)

We remark that the result of Theorem 1 begins to bridge the gap between Yao's bound and a bound sufficient for a super-linear lower bound for series-parallel circuits. For example, in the case of $T = \Theta(\log(N) \log\log(N))$, Theorem 1 gives us $S = \Omega(N)$, whereas Yao's bound gives $S = \Omega(N/\log\log(N))$. A bound of $S = \omega(N \log(N)/\log\log\log\log(N))$ would already imply a breakthrough in circuit complexity [10, 29, 4].

The proof of Theorem 1 goes in two steps. First, we show that a compression argument can be used to get a lower bound on the amount of space required when, for a large enough set of inversion points, the union of all points queried by the online algorithm is small. In the following, we abuse notation when X is a set and define $\varphi(X) = \bigcup\limits_{x \in X} \varphi(x)$.

▶ **Theorem 2.** *For every $T \in \mathbb{N}$, $\varphi : [N] \to \binom{[N]}{T}$, and $Y \subseteq [N]$ such that $|Y| < N - |\varphi(Y)|$, every strongly non-adaptive algorithm that solves permutation inversion with S bits of preprocessing and the query function φ must have*

$$S \geq |Y| \log(N - |\varphi(Y)| - |Y|) .$$

We can now already recover Yao's lower bound for strongly non-adaptive algorithms by Theorem 2. To see this, just consider the set $X = \{1, 2, ...N/(2T)\}$. Then $|\varphi(X)| \leq N/2$, and we get $S = \Omega((N \log N)/T)$.

This result also achieves optimal lower bounds for a specific subclass of query functions of interest: query functions which admit some $X \subseteq [N]$ of size $|X| = \Theta(N)$ with $|X| < N - |\varphi(X)|$. For example, take the query function φ which queries $\varphi(x) = (x, x+1, \dots, x+T-1 \mod N)$ for each $x \in [N]$. When $T \leq N/4$, $X = \{1, 2, \dots, N/4\}$ witnesses a lower bound of $S = \Omega(N \log N)$. We note, however, that such query functions make up a small fraction of all possible query functions; random φ do not have this property.

To get an improvement over Yao's bound, our second step involves picking a large enough set X of size $\Theta((N \log T)/T)$ with a small enough $\varphi(X)$. We show the existence of such a set via the probabilistic method. We start by viewing φ as a left T-regular bipartite graph, and prove the following graph lemma, where $N(X)$ denotes the neighborhood of the set of vertices X.

▶ **Lemma 3.** *Let $G = (L \sqcup R, E)$ be an undirected bipartite graph with $|L| = |R| = N$ and $|E| \leq NT$, where $T \leq N/5$. Then for large enough n, there exists a subset of vertices $X \subseteq L$, such that*

$$|X| \geq (N \log T)/(30T) \quad \text{and}$$

$$|\mathcal{N}(X)| \leq N - N/T^{4/5} .$$

It is not hard to see that Lemma 3 is tight for a random left T-regular bipartite graph.

1.2 Related Work

In the case of *adaptive algorithms*, the tight upper bound of $ST = O(N \log N)$ for permutation inversion is due to Hellman [18]. Hellman [18], and Fiat and Naor [14] gave upper bounds of $S^2T = \widetilde{O}(N^2)$ and $S^3T = \widetilde{O}(N^3)$ for inverting random and worst-case functions, respectively. It was recently observed [17] that the algorithm of Fiat and Naor for the worst-case function inversion can be extended to an upper bound of $TS^2 \max\{T, S\} = \widetilde{O}(N^3)$. De, Trevisan and Tulsiani [11] extended [14] and gave better trade-offs when inverting on only ε-fraction of the inputs.

The best known *strongly non-adaptive algorithm* is just the trivial one which achieves the trade-off $S/\log N + T = N$. For the case of weakly non-adaptive algorithms, where the online algorithm gets to see the advice first, there is an algorithm that slightly outperforms the trivial when $S > N$ [17]. The preprocessing algorithm stores $\log(N/T)$ first bits of a preimage for each $y \in [N]$, and the online algorithm queries all of the remaining T options, which results in $S = N \log(N/T)$.

The best *lower bound* is due to Yao [32], and it works for adaptive permutation inversion and thus also for function inversion. Moreover, since it works in the adaptive setting, it also trivially carries over to both the weakly and strongly non-adaptive settings. An alternate proof was given by Impagliazzo [21], and [15, 31, 11, 12] extend the lower bound to the setting of randomized algorithms inverting on ε-fraction of inputs.

Even in the case of strongly non-adaptive algorithms, the best known lower bound is still Yao's. While no unconditional improvement to Yao's bound is known prior to this work, for some *restricted models* there are better bounds. Barkan, Biham, and Shamir [1] give a lower bound of $S^2T = \Omega(N^2/\log N)$ for Hellman-type algorithms. Chawin, Haitner and Mazor [5] prove an adaptive lower bound of $S + T \log N = \Omega(N)$ when the preprocessing algorithm $\mathcal{P}$ computes a linear function. In the case of weakly non-adaptive algorithms they show that if the online algorithm $\mathcal{A}$ is an affine function over the query points and advice then $S = \Omega(N)$. Moreover they generalize these bounds to prove lower bounds in the case when $\mathcal{A}$ is an affine decision tree. [17] gives tight bounds for guess-and-check algorithms for weakly non-adaptive function inversion. These bounds are however incomparable to strongly non-adaptive function inversion (strongly non-adaptive algorithms can't look at the advice, but can output a point they haven't queried). Finally, Dvořák, Koucký, Král and Slívová [13] prove a conditional lower bound of $T = \Omega(\log N/\log\log N)$, when $S = \varepsilon N \log N$ under the network coding conjecture.

In the *quantum setting*, [26, 19, 7, 6] give tight bounds even with quantum advice showing that Grover's search is optimal in the setting when $S = \widetilde{O}(\sqrt{N})$. Any improvement on these bounds would imply circuit lower bounds as shown in [10].

1.3 Structure of the Paper

In Section 2, we provide the necessary definitions. In Section 3, we prove the main results of this paper: Theorem 1, Theorem 2 and Lemma 3. We conclude this paper with a discussion on the equivalence between function inversion and the communication complexity of certain partial functions in Section 4.

2 Preliminaries

All logarithms are base 2. For a non-negative integer N, by $[N]$ we denote the set $\{1, \ldots, N\}$, and by Π_N we denote the set of all permutations of $[N]$. For an undirected graph $G = (V, E)$ and a subset of its vertices $S \subseteq V$, $N(S)$ denotes its neighborhood; i.e.,

$$N(S) := \{v \in V : \exists u \in S \text{ s.t. } \{u, v\} \in E\} .$$

We will use the following Chernoff bound (see e.g., [24]):

▶ **Lemma 4.** *Let $X_1, \ldots, X_n$ be independent random variables taking values in $\{0,1\}$ and X denote their sum with $\mu = \mathbb{E}[X]$. Then for $0 \leq \varepsilon \leq 1$,*

$$\Pr[X \leq (1-\varepsilon)\mu] \leq \exp\left(\frac{-\varepsilon^2\mu}{2}\right).$$

2.1 The Permutation Inversion Problem

In the following definitions, let $(\mathcal{P}, \mathcal{A})$ be a pair of deterministic algorithms.

▶ **Definition 5.** *We say that*

1. *$(\mathcal{P}, \mathcal{A})$ uses S bits of pre-processing if for all inputs, the output of $\mathcal{P}$ has bit-length at most S.*
2. *$(\mathcal{P}, \mathcal{A})$ makes T queries if for all inputs, $\mathcal{A}^f$ makes at most T queries to f.*

In this paper, we provide lower bounds on *permutation* inversion, a subproblem of function inversion. Hence, our lower bounds extend to function inversion as well.

▶ **Definition 6.** *We say that $(\mathcal{P}, \mathcal{A})$ solves the permutation inversion problem if for all $\pi \in \Pi_N$ and $y \in [N]$,*

$$\mathcal{A}^\pi(\mathcal{P}(\pi), y) = \pi^{-1}(y) .$$

We call $\mathcal{P}$ the preprocessing algorithm and $\mathcal{A}$ the online algorithm.

We say that $(\mathcal{P}, \mathcal{A})$ is strongly non-adaptive if the T queries to π made by $\mathcal{A}^\pi$ depend only on y and not on the output of $\mathcal{P}(\pi)$ nor the results of previous queries. In such a case, we can define the query function of $\mathcal{A}^\pi$ to be $\varphi : [N] \to \binom{[N]}{T}$.

For any set $X \subseteq [N]$, we let $\varphi(X) = \bigcup_{x \in X} \varphi(x)$.

3 Non-Adaptive Function Inversion

In this section, we prove our improved lower bound on non-adaptive permutation inversion and hence function inversion. We start by showing a generic space bound (Theorem 2) that follows via a compression argument. This already allows us to recover Yao's lower bound in the strongly non-adaptive setting. We next introduce a special graph lemma (Lemma 3) on sparse bipartite graphs that guarantees the existence of a large enough subset of vertices with a small neighborhood. Finally, combining these two together, we get our improved lower bound (Theorem 1).

▶ **Theorem 2.** *For every $T \in \mathbb{N}$, $\varphi : [N] \to \binom{[N]}{T}$, and $Y \subseteq [N]$ such that $|Y| < N - |\varphi(Y)|$, every strongly non-adaptive algorithm that solves permutation inversion with S bits of preprocessing and the query function φ must have*

$$S \geq |Y| \log(N - |\varphi(Y)| - |Y|) .$$

Proof. Let $(\mathcal{P}, \mathcal{A})$ be a strongly non-adaptive algorithm for permutation inversion with query function φ that uses S bits of preprocessing. Let $X \subseteq [N]$ be such that $|X| < N - |\varphi(X)|$. For ease of notation, we define $\overline{\varphi(X)} := [N] \setminus \varphi(X)$. Because $|X| < N - |\varphi(X)| = |\overline{\varphi(X)}|$, there exist injective functions from $\varphi(X)$ to $[N] \setminus X$. Fix τ to be any such function, and let

$P = \{\pi \in \Pi_N \,:\, \pi|_{\varphi(X)} = \tau\}$; in particular, for any two $\pi_1, \pi_2 \in P$, $\pi_1|_{\varphi(X)} = \pi_2|_{\varphi(X)}$. Then by construction, we have that for each $\pi \in P$, $\pi^{-1}(X) \subseteq \overline{\varphi(X)}$. We now pick a maximal subset $Q \subseteq P$ such that for every distinct $\pi_1, \pi_2 \in Q$, $\pi_1^{-1}|_X \neq \pi_2^{-1}|_X$. Thus,

$$|Q| = \binom{|\overline{\varphi(X)}|}{|X|} \cdot |X|! = \frac{|\overline{\varphi(X)}|!}{(|\overline{\varphi(X)}| - |X|)!} \geq (|\overline{\varphi(X)}| - |X|)^{|X|} .$$

Assume for the sake of contradiction that $2^S < |Q|$. Then by the pigeon hole principle, there exist two distinct $\pi_1, \pi_2 \in Q$ such that $\mathcal{P}(\pi_1) = \mathcal{P}(\pi_2)$. This implies that for all $i \in X$, $\mathcal{A}^{\pi_1}(\mathcal{P}(\pi_1), i) = \mathcal{A}^{\pi_2}(\mathcal{P}(\pi_2), i)$, since by construction $\pi_1|_{\varphi(X)} = \pi_2|_{\varphi(X)}$. This is a contradiction, as we know there exists some $i \in X$ for which $\pi_1^{-1}(i) \neq \pi_2^{-1}(i)$. Hence, $2^S \geq |Q| \geq (|\overline{\varphi(X)}| - |X|)^{|X|}$, and $S \geq |X| \log(|\overline{\varphi(X)}| - |X|)$. ◀

From this, we can get a lower bound on the size of the preprocessed advice for any query function φ which has a large X with small $\varphi(X)$. In the following lemma, we show that all query functions (viewed as bipartite graphs) admit such a subset X.

▶ **Lemma 3.** *Let $G = (L \sqcup R, E)$ be an undirected bipartite graph with $|L| = |R| = N$ and $|E| \leq NT$, where $T \leq N/5$. Then for large enough n, there exists a subset of vertices $X \subseteq L$, such that*

$$|X| \geq (N \log T)/(30T) \quad \textit{and}$$
$$|\mathcal{N}(X)| \leq N - N/T^{4/5} .$$

Proof. When $d \leq 32$, we can take $X \subseteq L$ simply to be the subset of $(n \log d)/(30d)$ vertices on the left with the smallest degrees. Then we have $|N(X)| \leq d \cdot |X| = (n \log d)/30 < n - n/d^{4/5}$. Thus, in the following, assume $d \geq 33$.

To prove the existence of such a subset X, we will first pick a random subset X of vertices from L. We will then bound the probability of X being small or having a large neighborhood away from 1. This will imply the existence of a set of X that satisfies both conditions of our lemma.

Let $p = (\log d)/(3d) \in (0, 1)$, and let each vertex $a \in L$ be in X independently with probability p. We now compute the probability of our two bad events. First, to bound the probability of picking a small X, we can apply a Chernoff Bound (Lemma 4) to get that $\Pr\left[|X| \leq \frac{pn}{10}\right] \leq e^{-0.405pn} < e^{-pn/3}$.

Now, to get a bound on the probability that the size of the neighborhood $N(X)$ is close to n, let us first compute the expected size of $N(X)$:

$$\begin{aligned}\mathbb{E}\left[|N(X)|\right] &= \sum_{b \in R} \Pr[b \in N(X)] \\ &= n - \sum_{b \in R} \Pr[b \notin N(X)] . \qquad (1)\end{aligned}$$

The probability that $b \notin N(X)$ is the probability that none of the vertices $a \in N(b)$ were picked in X; i.e., $\Pr[b \notin N(X)] = (1-p)^{|N(b)|}$. Substituting into Equation (1) we get

$$\begin{aligned}\mathbb{E}\left[|N(X)|\right] &= n - \sum_{b \in R} (1-p)^{|N(b)|} \\ &\leq n - n\,(1-p)^{\frac{1}{n}\sum_b |N(b)|} \qquad (2) \\ &\leq n - n\,(1-p)^{d} , \qquad (3)\end{aligned}$$

where Equation (2) follows from the AM-GM inequality and Equation (3) follows from the fact that G has at most dn edges. Note that for $d > 1$ and $p = (\log d)/(3d)$, we have $0 < p < 1/4$. From this, we get for all $d \geq 33$

$$\begin{aligned}(1-p)^d &\geq e^{-d(p+p^2)} \\ &\geq e^{-d(p+\frac{p}{4})} \\ &= e^{-\frac{5pd}{4}} \\ &= d^{-\frac{5\log e}{12}} \\ &> \frac{2}{d^{4/5}} \, .\end{aligned}$$

Now we can conclude $\mathbb{E}\left[|N(X)|\right] < n - 2n/d^{4/5}$. With an upper bound on the expected size of $N(X)$, we apply Markov's inequality to get

$$\begin{aligned}\Pr\left[|N(X)| > n - \frac{n}{d^{4/5}}\right] &< \frac{n - 2n/d^{4/5}}{n - n/d^{4/5}} \\ &= 1 - \frac{1}{d^{4/5}(1 - 1/d^{4/5})} \\ &\leq 1 - d^{-4/5} \, .\end{aligned}$$

A union-bound over the probability of the two bad events happening gives

$$\Pr\left[|X| < \frac{pn}{10} \text{ or } |N(X)| > n - \frac{n}{d^{4/5}}\right] < 1 - d^{-4/5} + e^{-pn/3} \, . \qquad (4)$$

Now, because $d \leq n/5$, $d \leq (5\log e)n/36$ and hence $4/5 \leq (n\log e)/(9d)$. This gives us $d^{4/5} \leq d^{(n\log e)/(9d)} = e^{pn/3}$. From this, we can conclude that the probability in Equation (4) is strictly less than 1. This implies that there exists some $X \subseteq L$ with $|X| \geq pn/10$ and $|N(X)| \leq n - n/d^{4/5}$. ◀

Now by combining Theorem 2 and Lemma 3, we get our main result.

▶ **Theorem 1.** *Every strongly non-adaptive algorithm that solves permutation inversion with S bits of preprocessing and $T \leq N/5$ queries must have*

$$S = \Omega\left(\frac{N\log(N)\log(T)}{T}\right) \, .$$

Proof. If $T < 3$, then take $T = 3$ by making more queries, and the following lower bound still holds. So, without loss of generality assume that $3 \leq T \leq N/5$.

Consider the bipartite graph of left-degree T defined by φ on $(L \sqcup R, E)$, where $L = \{\ell_1, \ldots, \ell_N\}$, $R = \{r_1, \ldots, r_N\}$, and for every $i \in [N]$ and $j \in \varphi(i)$ we have $\{\ell_i, r_j\} \in E$. Now let $X \subseteq [N]$ be the set guaranteed to exist by Lemma 3, so $|X| = \lceil N\log(T)/(30T)\rceil$ and $|\varphi(X)| \leq N - N/T^{4/5}$. Note that for all $T > 0$, $\log T < 15T^{1/5}$, so $N\log T/(15T) < N/T^{4/5}$. Thus, $|X| < |\overline{\varphi(X)}|$. Therefore, by Theorem 2, $S \geq |X|\log(|\overline{\varphi(X)}| - |X|)$. Note that

$$\begin{aligned}|\overline{\varphi(X)}| - |X| &\geq \frac{N}{T^{4/5}} - \frac{N\log(T)}{30T} \\ &= \frac{N}{T^{4/5}}\left(1 - \frac{\log(T)}{30T^{1/5}}\right) \\ &\geq \frac{N}{2T^{4/5}}\end{aligned}$$

for $T > 0$. Thus, we have

$$S \geq \frac{N\log(T)}{30T}\log\left(\frac{N}{2T^{4/5}}\right) \geq \frac{N\log(T)}{30T}\log\left(\frac{N^{1/5}}{2}\right) = \Omega\left(\frac{N\log(N)\log(T)}{T}\right) \, .$$ ◀

4 Connections to Communication Complexity

In this section, we discuss an alternate approach to proving lower bounds for strongly non-adaptive function inversion via communication complexity. This approach generalizes to other strongly non-adaptive data structure problems.

Let $(\mathcal{P}, \mathcal{A})$ be a strongly non-adaptive algorithm for permutation inversion. We say that two permutations π, τ *conflict* under a query function φ if there exists an i such that $\pi^{-1}(i) \neq \tau^{-1}(i)$ and for every $j \in \varphi(i)$, $\pi(j) = \tau(j)$. Hence, to distinguish two conflicting permutations, we must have $\mathcal{P}(\pi) \neq \mathcal{P}(\tau)$. Now consider the following promise equality problem ($\mathrm{PromEQ}_{\varphi}$).

▶ **Definition 7.** *For a given query function* $\varphi : [N] \to \binom{[N]}{T}$*,* $\mathrm{PromEQ}_{\varphi}$ *is the following promise decision problem. Given two permutations* $\pi, \tau \in \Pi_N$ *such that either* $\pi = \tau$ *or* π *and* τ *conflict under* ϕ*, decide which one of the two conditions holds.*

For a (promise) problem f, let $\mathrm{CC}^1(f)$ denote the one-way deterministic communication complexity of f. We then observe that $\mathrm{CC}^1(\mathrm{PromEQ}_{\varphi})$ is the minimum amount of space needed for preprocessing to solve permutation inversion using the query function φ. On one hand, given a strongly non-adaptive algorithm $(\mathcal{P}, \mathcal{A})$, in the communication protocol Alice can send Bob $\mathcal{P}(\pi)$. To verify, Bob just checks if $\mathcal{P}(\pi) = \mathcal{P}(\tau)$. When $\pi = \tau$, equality is preserved. Otherwise, when π and τ conflict we are guaranteed to have $\mathcal{P}(\pi) \neq \mathcal{P}(\tau)$. On the other hand, assume that we have a one-way communication protocol for $\mathrm{PromEQ}_{\varphi}$, and let σ_π be the message Alice sends to Bob when she receives π as input. We can then construct an algorithm $(\mathcal{P}, \mathcal{A})$ for permutation inversion, where $\mathcal{P}(\pi) = \sigma_\pi$. By the correctness of our communication protocol, we are guaranteed that there are no two conflicting permutations which share the same message σ_π. Hence, $\mathcal{A}$ can identify the inverse of the given point from σ_π and the points it queries. In particular, the question of understanding the complexity of strongly non-adaptive function inversion is equivalent to the following question.

▶ **Open Problem 8.** *Find the minimum one-way deterministic communication complexity of* $\mathrm{PromEQ}_{\varphi}$ *among all* $\varphi \colon [N] \to \binom{[N]}{T}$*,*

$$\min_{\varphi\colon [N]\to\binom{[N]}{T}} \mathrm{CC}^1(\mathrm{PromEQ}_{\varphi}) \ .$$

Note that each $\mathrm{PromEQ}_{\varphi}$ problem is a "subproblem" of equality (the accept sets of $\mathrm{PromEQ}_{\varphi}$ and equality are identical, and the reject set of $\mathrm{PromEQ}_{\varphi}$ is a subset of the reject set of equality). Recall that while equality admits an efficient randomized communication protocol, it has maximum deterministic communication complexity. Thus, to prove a polynomial lower bound for $\mathrm{PromEQ}_{\varphi}$ via a reduction from some known problem, the reduction must be deterministic. Moreover, the problem we reduce from must admit an efficient randomized communication protocol, while being sufficiently hard for any deterministic protocol.

4.1 Recovering our improved bound

To illustrate this approach, we now demonstrate how our main result (Theorem 1) can be obtained in this communication complexity framework. Given the discussion above, Theorem 1 is equivalent to proving a lower bound of $\mathrm{CC}^1(\mathrm{PromEQ}_{\varphi}) = \Omega(N \log(N) \log(T)/T)$ for all $\varphi\colon [N] \to \binom{[N]}{T}$. In order to do this, we first introduce an auxiliary promise problem $\mathrm{PermEQ}_{k,\Sigma}$ which checks equality of k-permutations over an alphabet Σ.

▶ **Definition 9.** *For a given alphabet* Σ *and* $k \leq |\Sigma|$, $\mathrm{PermEQ}_{k,\Sigma}$ *is the following promise decision problem. Given two* k*-permutations of* Σ *(strings of length* k *with distinct characters), decide if they are equal or not.*

In order to get a lower bound on $\mathrm{CC}^1(\mathrm{PromEQ}_\varphi)$, we reduce $\mathrm{PermEQ}_{k,\Sigma}$ to PromEQ_φ; then known lower bounds on $\mathrm{CC}^1(\mathrm{PermEQ}_{k,\Sigma})$ extend to $\mathrm{CC}^1(\mathrm{PromEQ}_\varphi)$. The following is a sketch of this reduction: Given some φ, we use Lemma 3 to get a large $X \subseteq [N]$ with small $\varphi(X)$. Then we take $\Sigma = \overline{\varphi(X)}$ and $k = |X|$. Now given α, a k-permutation of Σ, we construct π_α, a permutation of $[N]$, where π_α maps α to X and $\varphi(X)$ to $[N] \setminus X$. In particular, $\pi_\alpha|_{\varphi(X)}$ does not depend on α. Then it is not hard to see that for distinct k-permutations α and β of Σ, π_α and π_β conflict. Thus, in the reduction from $\mathrm{PermEQ}_{k,\Sigma}$ to PromEQ_φ, Alice and Bob first construct π_α and π_β from their inputs α and β and then run the protocol for PromEQ_φ. The lower bound then follows from the known lower bound of $\mathrm{CC}^1(\mathrm{PermEQ}_{k,\Sigma}) \geq \Omega(k \log |\Sigma|)$.

References

1 Elad Barkan, Eli Biham, and Adi Shamir. Rigorous bounds on cryptanalytic time/memory tradeoffs. In *CRYPTO*, 2006.

2 Alex Biryukov and Adi Shamir. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In *ASIACRYPT*, 2000.

3 Alex Biryukov, Adi Shamir, and David Wagner. Real time cryptanalysis of A5/1 on a PC. In *FSE*, 2001.

4 Chris Calabro. A lower bound on the size of series-parallel graphs dense in long paths. In *ECCC*, 2008.

5 Dror Chawin, Iftach Haitner, and Noam Mazor. Lower bounds on the time/memory tradeoff of function inversion. In *TCC*, 2020.

6 Kai-Min Chung, Siyao Guo, Qipeng Liu, and Luowen Qian. Tight quantum time-space tradeoffs for function inversion. In *FOCS*, 2020.

7 Kai-Min Chung, Tai-Ning Liao, and Luowen Qian. Lower bounds for function inversion with quantum advice. In *ITC*, 2020.

8 Sandro Coretti, Yevgeniy Dodis, and Siyao Guo. Non-uniform bounds in the random-permutation, ideal-cipher, and generic-group models. In *CRYPTO*, 2018.

9 Sandro Coretti, Yevgeniy Dodis, Siyao Guo, and John Steinberger. Random oracles and non-uniformity. In *Eurocrypt*, 2018.

10 Henry Corrigan-Gibbs and Dmitry Kogan. The function-inversion problem: Barriers and opportunities. In *TCC*, 2019.

11 Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and PRGs. In *CRYPTO*, 2010.

12 Yevgeniy Dodis, Siyao Guo, and Jonathan Katz. Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In *EUROCRYPT*, 2017.

13 Pavel Dvořák, Michal Koucký, Karel Král, and Veronika Slívová. Data structures lower bounds and popular conjectures. In *ESA*, 2021.

14 Amos Fiat and Moni Naor. Rigorous time/space tradeoffs for inverting functions. In *STOC*, 1991.

15 Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *FOCS*, 2000.

16 Alexander Golovnev, Siyao Guo, Thibaut Horel, Sunoo Park, and Vinod Vaikuntanathan. Data structures meet cryptography: 3SUM with preprocessing. In *STOC*, 2020.

17 Alexander Golovnev, Siyao Guo, Spencer Peters, and Noah Stephens-Davidowitz. Revisiting time-space tradeoffs for function inversion. In *CRYPTO*, 2023.

18 Martin Hellman. A cryptanalytic time-memory trade-off. *IEEE Trans. Inf. Theory*, 26(4):401–406, 1980.

19 Minki Hhan, Keita Xagawa, and Takashi Yamakawa. Quantum random oracle model with auxiliary input. In *ASIACRYPT*, 2019.

20 Shuichi Hirahara, Rahul Ilango, and Ryan Williams. Beating brute force for compression problems. In *STOC*, 2024.

21 Russell Impagliazzo. Relativized separations of worst-case and average-case complexities for NP. In *CCC*, 2011.

22 Tsvi Kopelowitz and Ely Porat. The strong 3SUM-INDEXING conjecture is false. *arXiv:1907.11206*, 2019.

23 Noam Mazor and Rafael Pass. The non-uniform perebor conjecture for time-bounded Kolmogorov complexity is false. In *ITCS*, 2024.

24 Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

25 Arvind Narayanan and Vitaly Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. In *CCS*, 2005.

26 Aran Nayebi, Scott Aaronson, Aleksandrs Belovs, and Luca Trevisan. Quantum lower bound for inverting a permutation with advice. *Quantum Inf. Comput.*, 15(11-12):901–913, 2015.

27 Philippe Oechslin. Making a faster cryptanalytic time-memory trade-off. In *CRYPTO*, 2003.

28 Dominique Unruh. Random oracles and auxiliary input. In *CRYPTO*, 2007.

29 Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In *MFCS*, 1977.

30 Emanuele Viola. On the power of small-depth computation. *Found. Trends Theor. Comput. Sci.*, 5(1):1–72, 2009.

31 Hoeteck Wee. On obfuscating point functions. In *STOC*, 2005.

32 Andrew Chi-Chih Yao. Coherent functions and program checkers. In *STOC*, 1990.

Information-Theoretic Single-Server PIR in the Shuffle Model

Yuval Ishai
Technion, Haifa, Israel

Mahimna Kelkar
Cornell University, New York, NY, USA

Daniel Lee
MIT, Cambridge, MA, USA

Yiping Ma
University of Pennsylvania, Philadelphia, PA, USA

Abstract

We revisit the problem of private information retrieval (PIR) in the *shuffle model*, where queries can be made anonymously by multiple clients. We present the first single-server PIR protocol in this model that has sublinear per-client communication and *information-theoretic security*. Moreover, following one-time preprocessing on the server side, our protocol only requires sublinear per-client *computation*. Concretely, for every $\gamma > 0$, the protocol has $O(n^{\gamma})$ communication and computation costs per (stateless) client, with $1/\mathsf{poly}(n)$ statistical security, assuming that a size-n database is simultaneously accessed by $\mathsf{poly}(n)$ clients. This should be contrasted with the recent breakthrough result of Lin, Mook, and Wichs (STOC 2023) on doubly efficient PIR in the standard model, which is (inherently) limited to computational security.

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

Keywords and phrases Private information retrieval, Shuffle model

Digital Object Identifier 10.4230/LIPIcs.ITC.2024.6

Related Version *Full Version*: `https://eprint.iacr.org/2024/930`

Funding This research was supported by a Google faculty grant. Yuval Ishai was additionally supported by ERC Project NTSC (742754), BSF grants 2018393 and 2022370, ISF grant 2774/20, and ISF-NSFC grant 3127/23. Mahimna Kelkar and Yiping Ma were partially supported by a Technion research scholarship. Yiping Ma was also supported by a Microsoft Research PhD Fellowship.

1 Introduction

A private information retrieval (PIR) protocol [15, 36] allows a client to fetch an entry from a database server without revealing which entry was fetched. Specifically, the server holds a database $x = (x_1, \ldots, x_n)$ consisting of n bits (or generically, n symbols over an alphabet Σ) while the client holds an index $i \in \{1, \ldots, n\}$; the client wishes to obtain x_i while hiding i from the server.

PIR protocols have been broadly studied in two flavors: information-theoretic and computational. Information-theoretic protocols provide security against computationally unbounded adversaries and do not require "cryptographic" computations. Unfortunately, non-trivial information-theoretic PIR (with less than n bits of communication) is impossible given only one server [15]. Consequently, PIR protocols in this setting need database replication across two or more non-colluding servers. This poses challenges for deployment since the cost of managing multiple storage spots is high when databases are large (e.g., synchronization, monetary cost), and enforcing non-collusion on the database servers is hard in practice, especially when the data is owned by a single entity (e.g., a company). In contrast, computational PIR can work when only one server holds the database but only provides security against polynomial-time adversaries due to its reliance on cryptographic hardness assumptions (e.g., quadratic residuosity, learning with errors). Furthermore, the associated cost is typically high due to expensive cryptographic operations at the server – indeed, existing single-server protocols [3, 5, 6, 17, 29, 38] are significantly less efficient in practice than the multi-server information-theoretic ones [26, 27].

The shuffle model: PIR with many clients. Achieving the best of both worlds, as aforementioned, is not possible in the standard model without using n bits of communication. To circumvent this barrier, Ishai, Kushilevitz, Ostrovsky and Sahai [32] proposed a relaxed model, where *many* clients (with arbitrarily correlated indices) simultaneously query a single server, but the clients are granted the ability to make anonymous queries to the server. Abstractly, we can think of the queries as being *shuffled* before reaching the server.

Specifically, consider a client using a *multi-server* PIR query algorithm to generate sub-queries for a query index. If these sub-queries were naively sent to a *single server*, the server would immediately learn the query index of this client. However, this work and [32] show the power of shuffling: if there are many clients and their sub-queries are randomly permuted by a shuffler before being sent to the server, then it is hard for the server – even one that is computationally unbounded as we show in this work – to figure out any of the client-query indices. Therefore, this single server in the shuffle model can simply perform "cheap" operations of the multi-server PIR scheme to answer sub-queries.

Understanding *the shuffle model* in the context of PIR is well-motivated by real-world applications: databases with high-volume queries, such as stock quotes and search engines, naturally enjoy the feature that thousands of users access the databases at the same time, and therefore considering PIR with many simultaneously querying clients is sensible, particularly if it allows for cheaper server cost. Note that this is a substantially different goal from batch PIR [28, 31] which amortizes the cost of multiple queries from a *single client*.

The shuffle model has been considered also in problems orthogonal to PIR, including secure aggregation [8, 23, 32] and differential privacy [4, 10, 13, 14, 22]. Analogously to these works, we view shuffling as an *atomic* operation; existing literature on differential privacy [8] and anonymity [1, 18, 30, 37, 41] discusses how to implement shuffling efficiently (see details in Section 1.2).

The shuffle PIR model opens a promising direction toward constructing efficient single-server PIR protocols. In this work, we establish the *theoretical feasibility* of non-trivial single-server PIR with information-theoretic security in the shuffle model.

1.1 Our Results

This paper aims to develop a formal understanding of PIR in the shuffle model from a theoretical perspective. We briefly detail our results below.

Information-theoretic single-server PIR in the shuffle model (Sections 5 and 6). We present the first construction for single-server PIR in the shuffle model that has *sublinear communication* and *information-theoretic* security (with inverse-polynomial statistical error). Moreover, our construction is also *doubly efficient*: following one-time preprocessing on the server side, and without any state information on the client side, the server's per-query computation is sublinear in the database size.

▶ **Theorem 1** (Informal). *For every constant $0 < \gamma < 1$, there exists a single-server PIR protocol in the shuffle model such that, on database of size n, and following one-time preprocessing on the server side, the protocol has $O(n^\gamma)$ per-query computation and communication, and $O(n^{1+\gamma/2})$ server storage. This is achieved with the following information-theoretic security guarantee: for any inverse polynomial $\epsilon = 1/p_1(n)$, there exists a polynomial $p_2(n) = O(n^{1+4/\gamma} \cdot (p_1(n))^8)$ such that the protocol has ϵ-statistical security as long as the total number of queries made by (uncorrupted) clients is at least $p_2(n)$.*

As a key technique, we describe a generic *inner-outer* paradigm that composes together two standard (multi-server) PIR protocols: an outer and an inner layer, to build a PIR protocol in the shuffle model. Besides, our results are robust against imperfect shuffling/anonymity (details in the full version).

While the above protocol only achieves inverse-polynomial (rather than negligible) security error, this is in fact the standard notion of security in several important settings, including differential privacy [19,21], secure computation with partial fairness [16,24,39], and secure computation over one-way noisy communication [2]. Our protocol demonstrates that *information-theoretic* security is indeed feasible *without database replication.* Moreover, while concrete efficiency is not the focus of this work, we believe that our approach has potential for reducing the cost of standard-model PIR when properly combined with single-server schemes and settling for a constant-factor cost reduction that might be significant in practice (see Section 6.4 for details).

Lower bound on security (Section 6.5). In the inner-outer paradigm, we show a security lower bound when *any* generic PIR protocol is used as the outer layer, and a *constant*-server PIR from a broad class is used as the inner layer; in particular, $1/\mathsf{poly}(n)$ statistical security is tight in the sense that negligible security error cannot be achieved with polynomially many clients. We also discuss open problems (Section 7) on whether negligible security is possible (with polynomially many clients) by using other protocols in the inner layer.

1.2 Discussion on the Shuffle Model

Two-way shuffling. In the problems such as secure aggregation and differential privacy with shuffle model, the shuffled messages are delivered to a server for analytics. Our PIR setting is a bit different, since responses need to be communicated back from the server to the client, we require the shuffling to be *two-way.* Specifically, we require not only that clients can send messages anonymously to the server but also that the server can respond to clients while still keeping the client identities hidden. It is important to note that shuffling or anonymity does not trivialize the problem; it hides who sends the messages but not the content of the messages. In practice, this two-way shuffling can be realized in a number of ways [7,8,12,18,35], even without computational assumptions.

A hybrid model. PIR in the shuffle model can also be equivalently viewed as a hybrid model between the standard single-server and multi-server PIR models: as an abstraction, the shuffler models a second “server” which is assumed to not collude with the main database server but does not hold a copy of the database and can only perform *database-irrelevant* computations. This alone makes the shuffle model interesting for practical deployments: non-collusion between two (or more) servers holding the same database can be difficult to enforce (since it is likely for them to be operated by the same company for data ownership reasons) making it a strong assumption in practice; in contrast, if only one server holds the database, then the "two" servers can be reasonably run by independent (and possibly geographically distributed) entities. We also note that it could be interesting to let this second database-irrelevant server perform more generic computations instead of just acting as a shuffler; we leave this exploration to future work.

2 Technical Overview

In this section, we present a toy protocol, which is insecure but conveys our core ideas; we then outline the techniques for building our eventual protocol from the toy protocol.

An insecure toy protocol. The starting point is the classic two-server information-theoretic PIR scheme by Beimel et al. [9]. In this scheme, a client first deterministically encodes its queried index $i \in [n]$ to a bit string $\mathbf{z}$ of length $m = O(\log n)$ (we call $\mathbf{z}$ the encoding of queried index, or simply query), and splits $\mathbf{z}$ to two *additive* shares in $\mathbb{F}_2^m$, $\mathbf{z}_1$ and $\mathbf{z}_2$ (we call them sub-queries), and then sends them to the two servers respectively.

We construct PIR in the shuffle model based on this protocol. Abstractly, each client generates two sub-queries (or shares) $\mathbf{z}_1$ and $\mathbf{z}_2$ as if it was querying using the above two-server scheme but in fact sends both sub-queries to a single server through an anonymous channel (which shuffles the sub-queries together with that from many other clients). Observe that this is exactly an instance of secure aggregation in the split-and-mix approach [8, 23, 32], where each input is split into two shares; the hope is that the server would learn nothing given the shuffled encoding shares from many clients.

There are two issues with this toy protocol. The first issue is obvious – the server learns the sum of all the encoding strings, and therefore can easily distinguish two sets of query indices by comparing the sum of their shares and the sum of their encodings. Note that leaking the sum to the server is exactly the goal of secure aggregation, but the sum should not be leaked in the PIR context. This leakage can be easily eliminated by letting one of the clients add a dummy share (a random string) to hide the sum. The second issue is more involved. In fact, splitting each input into only two shares is not enough to guarantee security; this can be demonstrated through a simple counter-example: suppose that the server wishes to distinguish between the 2-additive shares of zeros and that of ones (sharing over $\mathbb{F}_2$) . In the latter case, there is always an equal number of ones and zeros in the shares, while this is not true for the former case. This approach can be generalized to a "counting" based strategy (for sharing over any Abelian groups) and allows for generic efficient distinguishing attacks (details in the full version). While splitting into more additive shares, e.g., 4, is sufficient [8], this means we need a 4-server PIR (that has additive sub-queries) and thus leads to worse communication – $O(n^{3/4})$ in the 4-server scheme compared to $O(n^{1/2})$ in the two-server scheme (Section 4.1). On the road map to our general protocol with $O(n^{\gamma})$ communication (for any $\gamma > 0$), the first checkpoint is to bypass the above attack and achieve a protocol with $O(n^{1/2})$ communication; it turns out that the key ideas used for this also play a pivotal role in our final protocol design.

Randomizing inputs via the inner-outer paradigm. The core reason why the simple split-and-mix approach does not work with two additive shares is the presence of arbitrary correlation among the queries; indeed, if all queries were independent and uniformly random, then using two shares works perfectly. Our key insight to navigate around this is to randomize the queries using another PIR, resulting in *uniform random but pairwise independent* queries which is later shown to be sufficient for security.

Our construction employs a novel approach – the *inner-outer* paradigm, which composes a k-server PIR protocol as an *outer* layer with the previous 2-server PIR protocol (with 2-additive shares) as the inner layer. At a high level, the outer layer PIR randomizes the client queries before they get processed through the inner layer PIR. Below we call the outer layer protocol as OPIR and the inner layer protocol as IPIR.

Formally, the composition works as follows: for any database $x \in \{0,1\}^n$, on input an arbitrary query index $i \in [n]$, the client first runs the OPIR query algorithm to generate k queries $q_1, \ldots, q_k$; note that they naturally satisfy pairwise independence and each is uniformly random in the OPIR query space $\mathcal{Q}$, simply because of the security property of any PIR. Instead of sending them directly to the server, these queries are *interpreted as indices* to a new database x' of size $|\mathcal{Q}|$, where x' consists of the answers to all the possible OPIR queries (i.e., elements in $\mathcal{Q}$). Now the client runs IPIR query algorithm on the each of the k "indices" in $\{1, 2, \ldots, |\mathcal{Q}|\}$, and sends the IPIR sub-queries to the server. Specifically, the client maps an index to its encoding in the two-server protocol, and splits the encoding into 2 additive shares (sub-queries) in $\mathbb{F}_2^m$ where $m = O(\log |\mathcal{Q}|)$. Finally, to have the compilation work, the server needs to build the database x' for IPIR in advance, which is feasible as long as $|\mathcal{Q}|$ is polynomial in n.

The upshot of this compilation is that the server now sees a set of shuffled shares generated from uniformly random and pairwise independent query indices to the database x'. As we shall show next, this randomization achieves that, for any two multi-sets of queried indices I, I' with distance at most δ, the resulting multi-sets after processing through OPIR will be J, J' will have distance in expectation $\sqrt{\delta}$, even though J, J' are larger than I, I'. The distance further decreases to $\sqrt[4]{\delta}$ after processing through IPIR (additive sharing). We will show that having each client add only one random noise sub-query (on top of its real sub-queries) is sufficient to hide the $\sqrt[4]{\delta}$ distance from the server.

Analyzing split and mix with pairwise independence. We now analyze the split and mix approach for pairwise independence queries which we get from the OPIR; we use a balls-and-bins formulation for this analysis. Specifically, the OPIR queries of all clients can be viewed as throwing $B = k \cdot C$ balls randomly into $|\mathcal{Q}|$ bins where C is the number of clients and $\mathcal{Q}$ is the OPIR query space. Since the balls are pairwise independent, we can bound the expected difference in the balls-and-bins configuration from any two such distributions by $\Theta_{|Q|}(\sqrt{B})$. This implies that the *differences* between any two sets of B OPIR sub-queries (and consequently, the query indices in IPIR) is proportional to $\sqrt{B}$.

As a second step, we show once again using a balls-and-bins formulation that for any sets of IPIR indices with difference δ, when the indices are split into two shares, the expected difference is proportional to $\sqrt{\delta}$. This implies that any two sets of original client indices, once put through both OPIR and IPIR, will differ on expectation by $\sqrt[4]{B}$. Our final step shows that adding just 1 noise query per client results in being able to "hide" this $\sqrt[4]{B}$ difference in order to get $1/\mathsf{poly}(n)$ security. This analysis goes through as long as the total number of clients C is at least $\Omega(n^{5+c})$ for some constant $c > 0$. More details are provided in Section 6.1. We also show a concrete instantiation using a Reed-Solomon code based OPIR.

Improving communication using CNF-shares. Following this, in Section 6.2, we show how a CNF-sharing based construction can be used as the IPIR to reduce the communication complexity; in particular, using an s-CNF sharing allows us to reduce the communication cost to $O(n^{1/s})$ given $\Omega(n^{2s+1}/\epsilon^8)$ clients for statistical security ϵ. This cleanly generalizes our earlier construction. The security proof follows a similar outline as before but is somewhat more involved. We find a nice group theoretic formulation of the problem of understanding the symmetries within the CNF-sharing, which allows us to greatly simplify the analysis by leveraging simple results from that domain.

Lower bound on security. We show a lower bound on security for protocols within our inner-outer paradigm, by showing that negligible statistical distance cannot be achieved in this realm. To prove this, we borrow an idea from Ghazi et.al [23, Theorem 6], and extend their results on secret sharing to the PIR context. We observe that the query algorithms of multi-server PIR protocols can be viewed as secret sharing; this allows us to show that if the total number of possible ways to secret share a query index is $K = p_1(n)$ and there are $C = p_2(n)$ clients, then there must exist two sets of input indices with some $1/p_3(n)$ statistical distance, where p_1, p_2, p_3 are all polynomials in n.

3 Related Work

We note that the shuffle PIR model substantially differ from standard PIR models in the literature; the only other relevant work in this model is by Ishai et al. [32]. In this section, we discuss models and techniques specifically related to shuffling, and defer a longer comprehensive literature review on PIR to our full version.

Differential privacy (DP) for PIR. A line of work [4, 40] considers the DP notion for PIR assuming client anonymity. Here, clients send their query indices via onion routing to the server, and privacy is guaranteed by the shuffling of client indices along with some noise queries. Here DP guarantees that the server cannot distinguish neighboring sets of queries (i.e., differing in exactly one client). Unfortunately, DP is substantially *weaker* than standard PIR security and therefore insufficient in any application where client queries can be *arbitrarily correlated*, as evidenced by several works which show how sensitive information can be extracted through *frequency analysis*-based attacks [25, 34, 42].

The "Split and mix" technique. A core idea in our construction follows from an ingenious split-and-mix approach for secure summation by Ishai et al. [32]. Specifically, they give a one-round single-server secure aggregation protocol as follows: Each client *splits* its input into k additive shares; then, as part of the shuffle model, these shares from all the C clients are *mixed* together before being sent to the aggregation server who simply outputs the sum of all the shares. The security goal here is that server cannot infer anything about a particular client's input. More precisely, the shuffled shares of any two tuples of client inputs (with equal sum) should look indistinguishable. Ishai et al. [32] show that statistical security of $2^{-\sigma}$ can be achieved by using per-input $k = \Theta(\log C + \log p + \sigma)$ additive shares over a group of size p. Recent works [8, 23] improve this bound to $k = \lceil 2 + \frac{2\sigma + \log_2(p)}{\log_2(C)} \rceil$ and show that at least 4 shares are necessary.

In our shuffle PIR context, we find that 2 additive shares are sufficient due to our query randomization technique and the usage of additional *noise* queries; this cannot be done in the summation setting as the final output could change. Towards reducing the communication of our PIR protocol, we also generalize the split-and-mix approach to CNF shares.

4 Preliminaries

Basic notation. For $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, 2, \dots, n\}$. $\mathbb{F}$ denotes a finite field. $\mathfrak{S}_c$ denotes the symmetric group containing all permutations of c elements. We use bold letters to denote vectors (e.g., $\mathbf{z}$). We use $\mathsf{SD}(\mathcal{D}_1, \mathcal{D}_2)$ to denote the statistical distance between the distributions $\mathcal{D}_1$ and $\mathcal{D}_2$.

Unless specified, logarithms are taken to the base 2. The notation $\mathsf{poly}(\cdot)$ refers to a fixed but unspecified polynomial in its parameter; we use $\mathsf{polylog}(\cdot)$ to mean $\mathsf{poly}(\log(\cdot))$. The notation $\widetilde{O}$ hides arbitrary polylogarithmic factors.

We use $\xrightarrow{\$}$ to denote uniformly random sampling, $\rightarrow$ for output by deterministic algorithms, and $\,{\scriptstyle\$}\!\!\rightarrow$ for output by randomized algorithms.

4.1 Multi-Server Information-Theoretic PIR

We begin with the standard notion of multi-server information-theoretic PIR below.

▶ **Definition 2** (PIR). *Let Σ be a finite alphabet. A k-server PIR protocol over Σ is a tuple $\Phi = (\mathsf{Setup}, \mathsf{Query}, \mathsf{Answer}, \mathsf{Recon})$ with the following syntax:*

- $\mathsf{Setup}(x) \rightarrow P_x$*: a deterministic algorithm executed by all servers that takes in an n-entry database $x \in \Sigma^n$ and outputs its encoding P_x.*
- $\mathsf{Query}(i; n) \,{\scriptstyle\$}\!\!\rightarrow ((q_1, \dots, q_k), \mathsf{st})$*: a randomized algorithm (parameterized by n) executed by the client that takes in an index $i \in [n]$, and outputs sub-queries $q_1, \dots, q_k$ and a state* st*. The sub-query q_ℓ is sent to the ℓ-th server.*
- $\mathsf{Answer}^\ell(P_x, q_\ell) \rightarrow a_\ell$*: a deterministic algorithm executed by the ℓ-th server that takes in the encoding P_x and a sub-query q_ℓ, and outputs an answer a_ℓ. Since the* Answer *algorithm may be different for different servers, we use ℓ to denote the algorithm used by server ℓ.*
- $\mathsf{Recon}((a_1, \dots, a_k), \mathsf{st}) \rightarrow x_i$*: a deterministic algorithm executed by the client that takes in answers $a_1, \dots, a_k$ (where a_ℓ is from the ℓ-th server) and the state* st*, and outputs $x_i \in \Sigma$.*

Φ needs to satisfy the following correctness and security properties:
Correctness. For all $n \in \mathbb{N}$, any database $x = (x_1, \dots, x_n) \in \Sigma^n$, and all $i \in [n]$,

$$\Pr\left[\mathsf{Recon}((a_1, \dots, a_k), \mathsf{st}) = x_i : \begin{array}{rcl} P_x & \leftarrow & \mathsf{Setup}(x) \\ ((q_1, \dots, q_k), \mathsf{st}) & \leftarrow\!{\scriptstyle\$} & \mathsf{Query}(i; n) \\ (a_1, \dots, a_k) & \leftarrow & (\mathsf{Answer}^\ell(P_x, q_\ell))_{\ell=1}^k \end{array} \right] = 1.$$

Intuitively, correctness says that the client always gets the correct value of x_i.
Security. For all $n \in \mathbb{N}$, $i \in [n]$, and $T \subset [k]$, define the distribution

$$\mathcal{D}_n(i, T) := \left\{ \{q_\ell\}_{\ell \in T} : ((q_1, \dots, q_k), \mathsf{st}) \leftarrow\!{\scriptstyle\$}\, \mathsf{Query}(i; n) \right\}.$$

We say that Φ has (t, ϵ)-privacy (where $t < k$, and $\epsilon = \epsilon(n)$), if for all $n \in \mathbb{N}$, any two indices $i, i' \in [n]$, and any set $T \subset [k]$ such that $|T| < t$, we have

$$\mathsf{SD}(\mathcal{D}_n(i, T), \mathcal{D}_n(i', T)) \leq \epsilon(n).$$

Intuitively, (t, ϵ)-privacy says that any set of less than t colluding servers has a distinguishing advantage at most ϵ.

We provide as background (Appendix A), common PIR schemes that will be important for our construction for PIR in the shuffle model. The constructions employ the following general outline: The servers encode the database $x \in \Sigma^n$ as a polynomial P_x. To query the database at position i, the client first encodes i into a vector $\mathbf{z}^{(i)}$ where the encoding is defined in a way that results in $P_x(\mathbf{z}^{(i)}) = x_i$. The client now evaluates P_x at $\mathbf{z}^{(i)}$ while hiding $\mathbf{z}^{(i)}$ from the servers: it secret shares $\mathbf{z}^{(i)}$ into k shares, and each share is sent to one of the k servers (through e.g., additive or Shamir sharing). Each server can then evaluate P_x on one share and send the result to the client, who is able to reconstruct the entry x_i.

Other notation. For a PIR protocol Φ, we use $\mathcal{E}_\Phi$ to denote the encoding space of all indices. We use $\mathcal{Q}_\Phi$ to denote the space of all possible sub-queries (note that $\mathcal{Q}_\Phi$ may not equal $\mathcal{E}_\Phi$). For example, in the two-server construction above, $\mathcal{E}_\Phi$ contains all binary strings with Hamming weight d, and the space $\mathcal{Q}_\Phi$ is $\mathbb{F}_2^m$, i.e, in this case $\mathcal{E}_\Phi \subset \mathcal{Q}_\Phi$.

4.2 Balls and Bins

We formulate the core analysis of our constructions using the widely-used balls-and-bins problem, which we provide background and notation for here. Abstractly, the balls-and-bins problem analyzes the distribution of B (identical) balls thrown into N bins according to some distribution D (often independent and uniformly at random). To denote a final configuration of balls, we use a N-length vector $\mathbf{u} = (u_0, \dots, u_{N-1})$ where u_i denotes the number of balls in bin i. We say that $\mathbf{u} = (u_0, \dots, u_{N-1})$ is (B, N)-valid if each $u_i \in \mathbb{Z}^{\geq 0}$ and $\sum_i u_i = B$. Since our analysis often deals with sharing over a group $\mathbb{G}$, we may also label the bins using elements from $\mathbb{G}$; when $\mathbb{G}$ is unspecified, it is taken to be $\mathbb{Z}_N$.

▶ **Definition 3.** *Given (B, N)-valid configurations $\mathbf{u} = (u_0, \dots, u_{N-1})$ and $\mathbf{v} = (v_0, \dots, v_{N-1})$, we define the following useful terms:*

- *The edit distance, denoted by $\mathsf{ED}(\mathbf{u}, \mathbf{v})$ is defined as $\mathsf{ED}(\mathbf{u}, \mathbf{v}) = \frac{1}{2}\sum_{i=0}^{N-1} |u_i - v_i|$. Intuitively, this denotes the number of balls that need to be moved to convert $\mathbf{u}$ to $\mathbf{v}$. Note that the distance is symmetric since $\mathsf{ED}(\mathbf{u}, \mathbf{v}) = \mathsf{ED}(\mathbf{v}, \mathbf{u})$. The edit distance between two distributions $\mathcal{U}$ and $\mathcal{V}$, denoted by $\mathsf{ED}(\mathcal{U}, \mathcal{V})$; can now be defined as $\mathbb{E}_{\mathbf{u} \sim \mathcal{U}, \mathbf{v} \sim \mathcal{V}}[\mathsf{ED}(\mathbf{u}, \mathbf{v})]$.*
- *The ball-intersection $\mathbf{u} \sqcap \mathbf{v}$ is $(c_0, \dots, c_{N-1})$ where each $c_i = \min(u_i, v_i)$.*
- *The ball-difference $\mathbf{u} \ominus \mathbf{v}$ is $(u'_0, \dots, u'_{N-1})$ where each $u'_i = \max(0, u_i - v_i)$.*

5 Single-Server PIR in the Shuffle Model: Definitions and Preliminary Results

We now formally define single-server PIR in the shuffle model, which considers many query-making clients. Importantly, no coordination is assumed among clients.

▶ **Definition 4** (PIR in the shuffle model). *Let Σ be a finite alphabet. A (single-server) PIR protocol (over Σ) in the shuffle model is a tuple $\mathsf{ShPIR} = (\mathsf{Setup}, \mathsf{Query}, \mathsf{Answer}, \mathsf{Recon})$ with a syntax similar to that of a k-server PIR (Definition 2) except for a few changes given below:*

- $\mathsf{Setup}(x) \to P_x$*: a deterministic algorithm executed by the server that takes in an n-entry database $x \in \Sigma^n$ and outputs its encoding P_x.*
- $\mathsf{Query}(i; n) \xrightarrow{\$} (q_1, \dots, q_k)$*: a randomized algorithm (parameterized by n) executed by the client that takes in an index $i \in [n]$, and outputs sub-queries $q_1, \dots, q_k$. Unlike in Definition 2, k may be a function of n; this is possible since the shuffle model does not require k physical servers. Further, all sub-queries will be sent to the same server. For simplicity, here we omit the state in Definition 2.*

- $\mathsf{Answer}(P_x, q_\ell) \to a_\ell$: *a deterministic algorithm executed by the server that takes in the encoding P_x and a sub-query q_ℓ, and outputs an answer a_ℓ. Unlike in Definition 2, there is a single* Answer *algorithm.*
- $\mathsf{Recon}(a_1, \dots, a_k) \to x_i$: *a deterministic algorithm executed by the client that takes in answers $a_1, \dots, a_k$, where for all $\ell \in [k]$, a_ℓ is the answer to the client's sub-query q_ℓ; and outputs $x_i \in \Sigma$.*

ShPIR *needs to satisfy the following correctness property:*

Correctness. *For all $n \in \mathbb{N}$, database $x = (x_1, \dots, x_n) \in \Sigma^n$, and $i \in [n]$,*

$$\Pr\left[\mathsf{Recon}(a_1,\dots,a_k) = x_i : \begin{array}{rcl} P_x & \leftarrow & \mathsf{Setup}(x) \\ (q_1,\dots,q_k) & \leftarrow_{\$} & \mathsf{Query}(i;n) \\ (a_1,\dots,a_k) & \leftarrow & (\mathsf{Answer}(P_x,q_\ell))_{\ell=1}^k \end{array}\right] = 1.$$

ShPIR *also needs to satisfy the following security property in the model where client queries are shuffled before being sent to the server.*

Security. *We will parameterize security by a shuffler Π and a minimum number of honest client queries C. Formally, let $\Pi = \{\Pi_c\}_{c\in\mathbb{N}}$ be an ensemble such that Π_c is a distribution over the symmetric group $\mathfrak{S}_c$. When Π is unspecified, we assume that each Π_c is a uniform distribution over $\mathfrak{S}_c$; we refer to this as the uniform or perfect shuffler. We discuss imperfect shufflers in the full version.*

For a given n, Π, and C, and given a tuple $I = (i_1, \dots, i_C) \in [n]^C$ of client query indices, define the distribution

$$\widetilde{\mathcal{D}}_{n,\Pi,C}(I) = \left\{ \pi(\mathbf{q}) : \begin{array}{l} (q_1^{(1)},\dots,q_k^{(1)}) \leftarrow_{\$} \mathsf{Query}(i_1;n) \\ \cdots \\ (q_1^{(C)},\dots,q_k^{(C)}) \leftarrow_{\$} \mathsf{Query}(i_C;n) \\ \mathbf{q} \leftarrow (q_1^{(1)},\dots,q_k^{(1)},\dots,q_1^{(C)},\dots,q_k^{(C)}) \\ \pi \overset{\$}{\leftarrow} \Pi_{kC} \end{array} \right\}.$$

Then, we say that ShPIR *is (Π, C, ϵ)-secure if for every $n \in \mathbb{N}$ and all $C^* \geq C(n)$, and $I, I' \in [n]^{C^*}$, it holds that:*

$$\mathsf{SD}(\widetilde{\mathcal{D}}_{n,\Pi,C^*}(I), \widetilde{\mathcal{D}}_{n,\Pi,C^*}(I')) \leq \epsilon(n).$$

Efficiency metrics. We measure the efficiency of PIR constructions in the shuffle model using a few metrics below. Since we consider many clients querying the server, we will characterize the cost per query.

- *Per-query server computation*: for answering each query, the number of bits that the server reads from the database and the preprocessing bits.
- *Per-query communication*: the sizes of the client query and the server response.
- *Server storage*: the total number of bits, including the preprocessing bits, that are stored by the server.
- *Message complexity*: for each query, the number of anonymous messages required to send. This is separately considered from the communication cost, since we need to take into account the anonymity cost. In particular, this will help us delineate between, e.g., sending one anonymous message of size s and sending s anonymous messages each of size 1 (since the latter may have more network overhead).

While our main focus is the server and the anonymity cost, we may also consider *per-query client computation*, which is the computational complexity for issuing each query and reconstructing the answer. One may also consider *client storage* which is omitted in this work as the clients in our constructions are stateless.

Warm-up impossibility result. When considering PIR with multiple clients, it is useful to study the minimum number of clients required for security. We show that for any *linear* PIR (i.e., its encoding function is linear), which includes the constructions mentioned in Section 4.1 and others [9, 15], the number of clients required is at least the database size. We also show that no linear PIR protocol in the shuffle model has statistical security better than $\frac{n-C}{n-1}$ for $C < n$. See details in the full version.

6 General Constructions for Single-Server Shuffle PIR

We now present generic ways to build asymptotically efficient PIR protocols in the shuffle model from standard multi-server PIR constructions. The high-level idea is to compose together a protocol OPIR at the *outer* layer with a protocol IPIR at the *inner* layer, for randomizing the query indices. We call this the *inner-outer* paradigm for ShPIR.

Motivating the inner-outer paradigm. Recall that following the split-and-mix technique, the analysis of [8, 23] directly implies a shuffle PIR protocol with 4 additive shares and $O(n^{3/4})$ communication. We find that using 2 additive shares (which would give $O(n^{1/2})$ communication) are not sufficient for two reasons: (1) client queries are not individually random; and (2) client queries may be arbitrarily correlated with each other. For example, it is easy to distinguish between sets of client indices that are far apart (e.g. all querying for index i vs index i'; see the full version for details), even if extra *noise* queries are added by the clients to reduce the statistical distance. Furthermore, if the queries are uniformly random (even if not independent), three additive shares are enough [8] although two shares are still not sufficient here. This motivates our two-layer approach below.

The insight of having OPIR. The key insight we use to navigate around this is to first *randomize* the query indices by using a separate *outer* PIR, which we denote as OPIR. The goal of this OPIR protocol is two fold: first, it reduces the distance between the two multi-sets I and I'; and second, it transforms the queries in a way that makes them pairwise-independent which turns out to be sufficient for us to prove security. Concretely, the OPIR protocol takes two multi-sets I and I', who may differ by as much as $\delta = C$, and constructs two new (larger) query multi-sets J and J', whose difference is now proportional to $\sqrt{\delta}$, and whose elements are now pairwise-independent. Then J and J' will be used as query indices of an *inner* PIR with additive (or CNF) shares. In this way, the server sees the IPIR sub-queries as if they were generated from random (and pairwise independent) query indices.

ShPIR compilation. To compile the overall ShPIR protocol, the server will need encode the database x twice: once using OPIR and once using IPIR. More precisely, the server first sets up a database consisting of the answers to every possible OPIR sub-queries based on x: it defines a new database $x' = (x'_1, \ldots, x'_{n'})$ of size $n' = |\mathcal{Q}_{\mathsf{OPIR}}|$ where each entry x'_i is set to be OPIR.Answer(P_x, L_i) where L_i denotes the i-th element in the sorting of $\mathcal{Q}_{\mathsf{OPIR}}$. If OPIR.Answer is different for different servers, then a size kn' (where k is the number of OPIR servers) database can be used, which concatenates all the n'-sized databases where the ℓ-th database is defined using $\mathsf{OPIR.Answer}^\ell(P_x, L_i)$; see Construction B.1 for details. Now x', from the perspective of IPIR, is the database to be taken into the setup algorithm, i.e., the server runs $\mathsf{IPIR.Setup}(x')$, and the setup for ShPIR is done.

To query an index $i \in [n]$, a client will first use OPIR to generate queries $q_1, \ldots, q_k$ which are each uniformly random in the space $\mathcal{Q}_{\mathsf{OPIR}}$. Each of these q_ℓ can now be treated as an index i'_ℓ of the database x', following which the client will use IPIR.Query to fetch the i'_ℓ-th entry in x' that corresponds to q_ℓ. As a result, the final sub-queries to be sent to the server (along with additional noise) are generated by the client running IPIR.Query on the indices i'_ℓ for $\ell \in [k]$. The full details of the composition are given as Construction B.1 (Appendix B).

6.1 Composition with an Additive Two-Server IPIR

We start with our generic composition which uses an IPIR with two additive shares. We provide an overview of the core proof here; the full details are given in the full version.

▶ **Theorem 5** (ShPIR Composition Theorem for additive IPIR). *Let Φ be any k-server t-private information-theoretic PIR scheme where $k > t > 2$; denote its sub-query space size by Q and its answer size by A. Let Ψ be 2-additive PIR defined in Construction A.1. Then, for any database size $n \in \mathbb{N}$, given any $\epsilon > 0$, there exists a constant c_0 such that for $C \geq (c_0 Q^5)/(k\epsilon^8)$, the construction $\mathsf{ShPIR}(\Phi, \Psi)$ is a (Π, C, ϵ)-secure PIR in the shuffle model where Π is uniform. Here, Q, k, ϵ, C may all be functions of n. Furthermore, when $Q = \widetilde{O}(n)$ and assuming one-time preprocessing, the construction has:*

- *per-query server computation $O(A \cdot k^{\frac{3}{2}} \cdot Q^{\frac{1}{2}})$,*
- *per-query client computation $O(A \cdot k^{\frac{3}{2}} \cdot Q^{\frac{1}{2}})$,*
- *per-query communication $O(A \cdot k^{\frac{3}{2}} \cdot Q^{\frac{1}{2}})$,*
- *server storage $\widetilde{O}(A \cdot k^{\frac{3}{2}} \cdot Q^{\frac{3}{2}})$.*

▶ Remark 6 (Reduced cost for homogeneous servers). If OPIR has different Answer algorithms for the servers, the ShPIR server needs to store k sub-databases, where for ℓ-th sub-database the server treats $q \in \mathcal{Q}_{\mathsf{OPIR}}$ as the ℓ-th share and stores the corresponding answers. If OPIR.Answer is the same for all k servers, then ShPIR server only needs to store one such sub-database; as a result, both the per-query server computation and communication will be $O(A \cdot k \cdot Q^{\frac{1}{2}})$, and the server storage will be $O(A \cdot Q^{\frac{3}{2}})$. The client computation will be $O(A \cdot k \cdot Q^{\frac{1}{2}})$. See details in the full version.

6.1.1 Proof Outline of Theorem 5

Basic background. Consider a client query index $i \in [n]$. Recall that our k-server OPIR will first encode i into the space $\mathcal{E}_{\mathsf{OPIR}}$ and then split it into k sub-queries in the space $\mathcal{Q}_{\mathsf{OPIR}}$. When composing with the IPIR, these k sub-queries will now be interpreted as IPIR query indices within the IPIR database of size $|\mathcal{Q}_{\mathsf{OPIR}}|$. Each of the k indices will now be encoded within the IPIR encoding space $\mathcal{E}_{\mathsf{IPIR}}$, and then split into 2 shares in the space $\mathcal{Q}_{\mathsf{IPIR}}$. Note that the space $\mathcal{Q}_{\mathsf{OPIR}}$ and $\mathcal{E}_{\mathsf{IPIR}}$ have the same size, which is the size of the IPIR database, and that $\mathcal{E}_{\mathsf{IPIR}} \subset \mathcal{Q}_{\mathsf{IPIR}}$. Going forward, for clarity, we keep using "sub-queries" for OPIR but use "shares" to mean the sub-queries for IPIR.

Given C clients, we will have kC total IPIR query indices encoded into $\mathcal{E}_{\mathsf{IPIR}}$; denote this by $\mathsf{y} = (y_1, \ldots, y_{kC})$ and let $\widetilde{\mathsf{y}}$ (of length $2kC$) denote its shares in $\mathcal{Q}_{\mathsf{IPIR}}$. Our main goal is to analyze the properties of $\widetilde{\mathsf{y}}$ since this will be the view of the server. In particular, given two lists of original query indices $I = (i_1, \ldots, i_C)$ and $I' = (i'_1, \ldots, i'_C)$, and their resulting shares $\widetilde{\mathsf{y}}$ and $\widetilde{\mathsf{y}}'$, we want to understand whether an adversary can find e.g., which of I or I' corresponds to $\widetilde{\mathsf{y}}$.

Balls-and-bins-formulation. We now describe how to formulate our core analysis as a balls-and-bins problem. A key starting observation here is that a uniformly random shuffler Π will eliminate any ordering within $\widetilde{\mathsf{y}}$ (and similarly for y). In turn, this allows us to essentially do our analysis using a balls-and-bins formulation, where each share in $\widetilde{\mathsf{y}}$ corresponds to a ball in one of $|\mathcal{Q}_{\mathsf{IPIR}}|$ bins. More precisely, the distribution of the shuffled shares in $\widetilde{\mathsf{y}}$ is exactly a $|\mathcal{Q}_{\mathsf{IPIR}}|$-dimensional distribution where the each component represents the distribution of the number of balls in that bin. Towards this, we also find it helpful to analyze y using a similar balls-and-bins formulation.

The crux of our analysis now boils down to quantifying the statistical distance between the distribution of balls over bins resultant from any two sets of original query indices I and I'. Specifically, define $\mathcal{Y}(I)$ to be the distribution of the balls-and-bins configuration of IPIR query indices y resultant from the original query indices I; define $\widetilde{\mathcal{Y}}(I)$ to be the distribution of its shares (i.e., corresponding to $\widetilde{\mathsf{y}}$). Roughly, the goal now is to show that for any I and I', we can bound $\mathsf{SD}(\widetilde{\mathcal{Y}}(I), \widetilde{\mathcal{Y}}(I'))$ with some inverse polynomial in the number of clients.

Looking ahead however, we will require some extra balls to be added uniformly at random, essentially to "smooth out" the distribution of $\widetilde{\mathsf{y}}$; this can also be thought of as uniformly random *noise*. In the PIR context, this effectively corresponds to each client sending a random sub-query in $\mathcal{Q}_{\mathsf{IPIR}}$. We denote the balls-and-bins distribution of the shares with noise added as $\widetilde{\mathcal{Y}}^*(I)$.

▶ Remark 7 (Noise and communication complexity). We note that adding noise for each IPIR query index does not increase the asymptotic communication complexity for IPIR, i.e., the communication for an n-sized database is still $O(\sqrt{n})$. This is because the server will still evaluate each noise share either as the first or second share without changing the database encoding polynomial making the communication still $O(\sqrt{n})$. Note that adding noise is substantially different from splitting to more shares, i.e., if each IPIR index was instead split into more additive shares (corresponding to using an IPIR with more servers), then the number of variables in the encoding polynomial itself will be larger, which would increase the asymptotic communication.

Main proof steps. At a high level, we leverage balls-and-bins style analyses to bound the statistical distance between $\widetilde{\mathcal{Y}}^*(I)$ and $\widetilde{\mathcal{Y}}^*(I')$. The rough idea will be to first compute the *edit distance* between the balls-and-bins configurations corresponding to the IPIR shares and then use that to bound the statistical distance after adding the random noise. Our proof proceeds in three major steps which we outline below.

Proof Step 1: *(Analyzing the edit distance of* OPIR *sub-queries).* Consider two lists of client indices $I = (i_1, \ldots, i_C)$ and $I' = (i'_1, \ldots, i'_C)$. Abstractly, the first part of our proof shows that the edit distance between the OPIR sub-queries generated from I and I' is *not too large.*

Recall that the t-out-of-k OPIR sub-queries generated are individually uniformly random, and are $(t-1)$-wise independent (and therefore also pairwise independent). Therefore, we can formulate our objective as the following balls-and-bins problem given in Lemma 8.

▶ **Lemma 8.** *Suppose that B balls are thrown into N bins. Let $\mathcal{B}$ and $\mathcal{B}'$ be any two distributions of the final balls-and-bins configuration where each ball is thrown uniformly at random, and any two balls are independently thrown. Then* $\mathbb{E}_{\mathbf{u}\sim\mathcal{B},\mathbf{v}\sim\mathcal{B}'}[\mathsf{ED}(\mathbf{u},\mathbf{v})] \leq \sqrt{BN/2}$.

Casting this result to our construction, since each client index generates k OPIR sub-queries and there are C clients in total, the expectation of edit distance (or differences) between any two sets of OPIR sub-queries (and consequently, the IPIR indices) is at most $\sqrt{kC\,|\mathcal{Q}_{\mathsf{OPIR}}|/2}$.

Proof Step 2: *(Analyzing the edit distance of 2-additive sharing in the* IPIR*).* Now that we have a bound on the edit distance between OPIR sub-queries (and consequently IPIR indices), our next step is to analyze the edit distance for $\mathcal{Q}_{\mathsf{IPIR}}$ shares. Recall that each encoded $\mathcal{E}_{\mathsf{IPIR}}$ index is split into two additive shares. We model this as another balls-and-bins problem:

Consider a (B, N)-valid configuration $\mathbf{u}$ and let $\mathsf{Share}_{\mathbf{u}}$ denote the distribution of randomly splitting each ball in $\mathbf{u}$ (in a group $\mathbb{G}$), i.e., for each ball b, throw one ball into a random bin $u \leftarrow\!\!\$\, \mathbb{G}$, and another into bin $b - u$. The goal now is to bound the edit distance between $\mathsf{Share}_{\mathbf{u}}$ and $\mathsf{Share}_{\mathbf{v}}$ given the edit distance between $\mathbf{u}$ and $\mathbf{v}$.

To begin, we show that in the context of the final statistical distance, it is sufficient to only consider the parts of $\mathbf{u}$ and $\mathbf{v}$ that are different. Let $\mathsf{Share}^{\ell}_{\mathbf{u}}$ denote the distribution of the balls-and-bins configuration when further throwing ℓ balls independently and uniformly at random following the sharing $\mathsf{Share}_{\mathbf{u}}$. In particular, we show that,

$$\mathsf{SD}(\mathsf{Share}^{\ell}_{\mathbf{u}}, \mathsf{Share}^{\ell}_{\mathbf{v}}) \leq \mathsf{SD}(\mathsf{Share}^{\ell}_{\mathbf{u}\ominus\mathbf{v}}, \mathsf{Share}^{\ell}_{\mathbf{v}\ominus\mathbf{u}})$$

where $\ominus$ denotes the ball-difference operation defined in Section 4.2. Essentially, this will allow us to look at the splitting of only those balls that differ between $\mathbf{u}$ and $\mathbf{v}$; in particular, given (B, N)-valid $\mathbf{u}$ and $\mathbf{v}$ with edit distance δ, we will only need to concern ourselves with the (δ, N)-valid $\mathbf{u}' = \mathbf{u} \ominus \mathbf{v}$ and $\mathbf{v}' = \mathbf{v} \ominus \mathbf{u}$. We use this to show that $\mathbb{E}\left[\mathsf{ED}(\mathsf{Share}_{\mathbf{u}'}, \mathsf{Share}_{\mathbf{v}'})\right] \leq \sqrt{2\delta N}$. Combining this with the first part, we get:

$$\begin{aligned}\mathbb{E}_{\mathbf{u}\sim\mathcal{B},\mathbf{v}\sim\mathcal{B}'}\left[\mathsf{ED}(\mathsf{Share}_{\mathbf{u}\ominus\mathbf{v}}, \mathsf{Share}_{\mathbf{v}\ominus\mathbf{u}})\right] &\leq \sqrt{2N} \cdot \mathbb{E}_{\mathbf{u}\sim\mathcal{B},\mathbf{v}\sim\mathcal{B}'}\left[\sqrt{\mathsf{ED}(\mathbf{u},\mathbf{v})}\right] \\ &\leq \sqrt{2N}\,(BN/2)^{1/4} = (2)^{1/4}B^{1/4}(N)^{3/4}\end{aligned}$$

where the second step is by the concave Jensen's inequality.

Proof Step 3: *(Bounding the final statistical distance).* We are now ready to bound the final statistical distance between the final views of the server: $\widetilde{\mathcal{Y}}^*(I)$ and $\widetilde{\mathcal{Y}}^*(I')$. For this, we leverage a recent analysis by Boyle et al [11]. A straightforward corollary of their result can be abstractly stated as follows: Consider ℓ balls thrown independently and uniformly at random into N bins and let $\mathcal{U}_j$ denote the final distribution after another ball is added into bin j. Then for all bins j and j', we have $\mathsf{SD}(\mathcal{U}_j, \mathcal{U}_{j'}) \leq \sqrt{N/\ell}$. Informally, this can also be thought of as a "toy in sand" problem of being able to hide the location (bin j or bin j') of an initial ball (i.e., the toy) after throwing in N random balls as noise (i.e., the sand). The same analysis can be extended to show that if there are Δ initial balls, after which ℓ random balls are thrown, the statistical distance will be bounded by $\Delta \cdot \sqrt{N/\ell}$. In the context of our PIR analysis, intuitively, Δ will represent the edit distance between $\mathsf{Share}_{\mathbf{u}\ominus\mathbf{v}}$ and $\mathsf{Share}_{\mathbf{v}\ominus\mathbf{u}}$, while the ℓ extra balls will represent the additional "noise" IPIR queries made. Note that when using this balls-and-bins analysis, we need to account for the fact that the edit distance is a distribution in our case, rather than a fixed number; it is straightforward to do so by using standard first-moment techniques (since we have a bound on the expectation).

Casting these analyses back to our PIR context, first notice that $\widetilde{\mathcal{Y}}^*(I)$ is nothing but the distribution $\mathsf{Share}^{\ell}_{\mathbf{u}\sim\mathcal{B}(I)}$ where $\mathcal{B}(I)$ is the distribution of OPIR sub-queries resulting from the indices I. Looking ahead, we will use $\ell = kC$ uniformly random IPIR queries (i.e., k per client) as noise. A crucial point here is that the number of extra balls per client needs to be constant in C so that the individual communication complexity of each client does not depend on the how many clients are making queries. In fact, this also required our bound on the ED of the 2-additive sharing to be $o(\delta)$.

Combining the results from the previous parts, we show our main result:

$$\mathsf{SD}(\widetilde{\mathcal{Y}}^*(I), \widetilde{\mathcal{Y}}^*(I')) < \frac{3 \cdot N^{5/8}}{B^{1/8}} = \frac{3\,|\mathcal{Q}_{\mathsf{IPIR}}|^{5/8}}{(kC)^{1/8}}.$$

since $N = |\mathcal{Q}_{\mathsf{IPIR}}|$ bins (query-space) and $B = kC$ balls (total sub-queries).

A final task is bounding $|\mathcal{Q}_{\mathsf{IPIR}}|$ by Q (i.e., the size of OPIR sub-query space). The high-level idea here is that we let each IPIR database entry be A bits and consequently $|\mathcal{Q}_{\mathsf{IPIR}}|$ can be made $\widetilde{O}(Q)$. Now, assuming that there are $C = \Omega(n^{5+\nu}/k)$ client queries for some constant $\nu > 0$, the statistical distance can be bounded by some inverse polynomial $1/\mathsf{poly}(n)$ in n. More specifically, suppose that we wanted to bound the statistical distance by some inverse polynomial $\epsilon(n)$. Then, assuming at least $C(n) = \Omega(n^5/(k \cdot \epsilon^8))$ client queries, the statistical distance is bounded by ϵ. Consequently, the construction satisfies (Π, C, ϵ)-security in the shuffle model where Π is the uniform shuffler.

6.2 Reducing Communication using CNF Shares

In this section, we describe how to generalize the IPIR to use CNF shares instead of additive shares. The upshot is that it allows us to reduce the communication complexity of the resultant ShPIR protocol to $O(n^c)$ for any constant $c > 0$.

Construction outline. In a standard multi-server PIR, using s additive shares instead of 2 results in an increased communication cost of $O(n^{(s-1)/s})$ but this can be reduced to $O(n^{1/s})$ at the cost of a stronger non-collusion assumption using a CNF sharing where each server is given a different $s-1$ sized subset of the additive shares. We show that the same strategy in fact also works in our inner-outer paradigm by using an IPIR with CNF-shares (the composed protocol is given in Figure A.2). This compilation is particularly interesting since it requires *no extra non-collusion assumptions* to get the gain in efficiency (since the shuffle model already consists only of a single server). Instead, the trade-off will arise in the minimum number of clients required for security.

▶ **Theorem 9** (ShPIR Composition Theorem for CNF IPIR). *Let Φ be any k-server t-private information-theoretic PIR scheme where $k > t > 2$; denote its sub-query space size by Q and its answer size by A. Let Ψ be the s-CNF PIR defined in Construction A.2. Then, for any database size $n \in \mathbb{N}$, and given any $\epsilon > 0$, there exists a constant c_0 such that for $C \geq (c_0 Q^{2s+1})/(k\epsilon^8)$, the construction* ShPIR(Φ, Ψ) *is a (Π, C, ϵ)-secure PIR in the shuffle model where Π is uniform. Here, Q, k, ϵ, C may all be functions of n. Furthermore, when $Q = \widetilde{O}(n)$ and assuming one-time preprocessing, the construction has:*

- *per-query server computation $O(A \cdot k^{1+1/s} \cdot Q^{1/s})$,*
- *per-query client computation $O(A \cdot k^{1+1/s} \cdot Q^{1/s})$,*
- *per-query communication $O(A \cdot k^{1+1/s} \cdot Q^{1/s})$,*
- *server storage $\widetilde{O}(A \cdot k^{1+1/s} \cdot Q^{1+1/s})$,*

Similar to Remark 6, if OPIR.Answer *is the same for all k servers, then both the per-query server computation and communication will be $O(A \cdot k \cdot Q^{1/s})$, and the server storage will be $O(A \cdot Q^{1+1/s})$. The client computation will be $O(A \cdot k \cdot Q^{1/s})$.*

Proof outline. The overall structure of the proof is very similar to the one for an additive IPIR; the main difference being in the second proof part to analyze the balls-and-bins distribution after the IPIR sharing which now involves CNF shares instead of additive shares.

Let s-CNF-Share$_{\mathbf{u}}$ be the distribution of the balls-and-bins configuration upon sharing each ball in $\mathbf{u}$ into s CNF shares in $\mathbb{G}^{s-1}$. Now, given (δ, N)-valid configurations $\mathbf{u}$ and $\mathbf{v}$, we want to bound the edit distance between s-CNF-Share$_{\mathbf{u}}$ and s-CNF-Share$_{\mathbf{v}}$; Through a natural group theoretic formulation, this turns out essentially reduce to understanding the (cyclic rotational) symmetries of the CNF-sharing. Concretely, this allows us to show that:

$$\mathsf{ED}(s\text{-}\mathsf{CNF\text{-}Share}_{\mathbf{u}}, s\text{-}\mathsf{CNF\text{-}Share}_{\mathbf{v}}) \leq sN^{(s-1)/2}\sqrt{\delta}.$$

Notice how this bound asymptotically generalizes the one from the 2-additive IPIR construction. Once we have this bound, the rest of the security proof of proceeds in exactly the same way as the one for Add-ShPIR. The complete proof is given in the full version.

6.3 Concrete Constructions based on Reed-Muller Code

We can now concretely instantiate OPIR with the Reed-Muller PIR and IPIR with the CNF PIR to achieve our main result below.

▶ **Theorem 10.** *For every constant $0 < \gamma < 1$, there exists a Reed-Muller PIR Φ and a $(\lceil 2/\gamma \rceil)$-CNF PIR Ψ, such that on any database size $n \in \mathbb{N}$, given any $\epsilon > 0$, for all $C \geq c_0 n^{1+4/\gamma}/\epsilon^8$ where c_0 is some constant, the construction* ShPIR(Φ, Ψ) *is a (Π, C, ϵ)-secure PIR where Π is uniform. Furthermore, assuming one-time preprocessing, we get:*

- *per-query server computation $O(n^\gamma)$,*
- *per-query client computation $O(n^\gamma)$,*
- *per-query communication $O(n^\gamma)$,*
- *per-query message complexity $O(n^\gamma)$,*
- *server storage is $\widetilde{O}(n^{1+\gamma/2})$.*

We defer the full proof of Theorem 10 to the full version. One thing to note here is that the reduced communication per client with CNF shares comes at a price – to achieve the same level of security, we need a larger number of clients.

▶ Remark 11 (Sub-polynomial communication assuming super-polynomial number of clients). An interesting consequence of the CNF-based IPIR is that it also enables more efficient protocols in the shuffle model. Using a $(\log n)$-server CNF-based protocol as our IPIR, we can achieve communication of $O(\mathsf{polylog}(n))$ with the assumption that there are at least some super-polynomial $n^{O(\log n)}$ number of clients. This results in better asymptotic complexity than the best existing protocols [20] in the standard-model PIR which use a constant numbers of servers. Note that the shuffle model compilation means that still only one server is required for our protocol and therefore we do not require the non-collusion assumptions of the standard-model CNF-based PIR.

▶ Remark 12 (Negligible security with slightly sublinear communication). Our main result only achieves inverse-polynomial rather than negligible security error. We note that if one settles for *slightly sublinear* communication, there is a simple solution that achieves negligible security error and proceeds as follows. The server writes the n-bit database as an $m \times m$ matrix over $\mathbb{Z}_2$ where $m = \sqrt{n}$. Each client writes the column it is interested in as a unit vector $q \in \mathbb{Z}_2^m$. Assuming C clients query at the same time, where C is super-linear in n, each client splits the vector q into $k = O((m + \sigma)/\log C)$ additive shares, for security parameter $\sigma = \log^2 n$. For each query $q' \in \mathbb{Z}_2^m$, the server responds with $X \cdot q' \in \mathbb{Z}_2^m$. By the tight security analysis of the *additive* split-and-mix protocol [8, 23, 32], the security error is negligible in n, i.e., $\Theta(1/n^{\log n})$, and both the query and the answer are of size $k \cdot m = O(n/\log n)$.

6.4 Combining with Standard-Model PIR

Our shuffle PIR can be used as a blackbox to reduce server cost for standard single-server PIR by any constant factor (even $10\times$ is a concretely substantial improvement).

Take any standard single-server PIR scheme stdPIR and denote the shuffle PIR construction as ShPIR. The server organizes the size-n database as an $\ell \times (n/\ell)$ matrix where ℓ is a constant. The key idea here is to use stdPIR to retrieve a column and ShPIR to retrieve a row. The server treats each column as a database in ShPIR and runs ShPIR.Setup on it. The server stores the preprocessed results as lookup tables (hence n/ℓ tables in total).

Suppose a client wants to retrieve the entry at r-th row and c-th column. The client runs the query algorithm of ShPIR on index $r \in [\ell]$ and generates k sub-queries. Then the client sends k messages anonymously, where the j-th message consists of the j-th sub-query of ShPIR and a stdPIR query for index $c \in [n/\ell]$. On receiving each message, the server first processes the sub-query of ShPIR (essentially n/ℓ table lookup operations), which results in n/ℓ elements; then the server processes the stdPIR query on these n/ℓ elements.

Compared to running stdPIR on a size-n database, this technique reduces server computation by a factor of ℓ. And the ShPIR database size is ℓ, which neither requires too many clients nor incurs high anonymity cost. The tradeoff is that a client sends k messages in the stdPIR-ShPIR combination instead of one message when using stdPIR only.

6.5 Lower Bound on Security

We show that for shuffle PIR protocols constructed in the inner-outer paradigm, $1/\mathsf{poly}(n)$ statistical security is tight in the sense that negligible security cannot be achieved with polynomially many clients using the additive inner PIR. The proof is deferred to the full version. This result does not rule out the information-theoretic constructions with negligible error, in particular, an interesting open problem to consider is instantiating the inner PIR with the Reed-Muller construction.

▶ **Theorem 13** (Lower bound on security for ShPIR). *Let Φ be any multi-server PIR scheme. Denote the number of possible vectors of sub-queries as K_Φ. Let Ψ be a constant-server additive PIR (Construction A.1). On any database size $n \in \mathbb{N}$, for all (Π, C, ϵ)-secure* ShPIR(Φ, Ψ) *constructions where C, K_Φ and K_Ψ are all bounded by polynomial $p_1(n)$, there exists a polynomial p_2 such that $\epsilon \geq 1/p_2(n)$.*

7 Conclusion and Open Questions

We demonstrate that PIR in the shuffle model can circumvent several limitations of standard-model PIR. This includes information-theoretic security with a single server, which opens a direction of constructing concretely efficient single-server schemes in the future.

The main technical question we leave open in this work is the possibility of obtaining similar results with negligible security error (recall that we can achieve this with slightly sublinear communication, see Remark 12). We conjecture that polylogarithmic communication per client with negligible security can be achieved by instantiating both OPIR and IPIR with the Reed-Muller PIR construction with a polylogarithmic security threshold and a polylogarithmic communication complexity.

Finally, an interesting direction for future research is obtaining concretely efficient PIR schemes in the shuffle model, possibly by settling for computational security.

References

1 Ittai Abraham, Benny Pinkas, and Avishay Yanai. Blinder - scalable, robust anonymous committed broadcast. In *CCS*, pages 1233–1252, 2020.

2 Shweta Agrawal, Yuval Ishai, Eyal Kushilevitz, Varun Narayanan, Manoj Prabhakaran, Vinod M. Prabhakaran, and Alon Rosen. Secure computation from one-way noisy communication, or: Anti-correlation via anti-concentration. In *CRYPTO*, pages 124–154, 2021.

3 Carlos Aguilar-Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. XPIR: Private information retrieval for everyone. In *PETS*, 2016.

4 Kinan Dak Albab, Rawane Issa, Mayank Varia, and Kalman Graffi. Batched differentially private information retrieval. In *USENIX Security*, pages 3327–3344, 2022.

5 Asra Ali, Tancrède Lepoint, Sarvar Patel, Mariana Raykova, Phillipp Schoppmann, Karn Seth, and Kevin Yeo. Communication-computation trade-offs in PIR. In *USENIX Security*, pages 1811–1828, 2021.

6 Sebastian Angel, Hao Chen, Kim Laine, and Srinath Setty. PIR with compressed queries and amortized query processing. In *IEEE S&P*, pages 962–979, 2018.

7 Borja Balle, James Bell, and Adrià Gascón. Amplification by shuffling without shuffling. In *CCS*, pages 2292–2305, 2023.

8 Borja Balle, James Bell, Adria Gascon, and Kobbi Nissim. Private summation in the multi-message shuffle model. In *CCS*, pages 657–676, 2020.

9 Amos Beimel, Yuval Ishai, and Eyal Kushilevitz. General constructions for information-theoretic private information retrieval. In *Journal of Computer and System Sciences*, 2005.

10 Andrea Bittau, Ulfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. PROCHLO: Strong privacy for analytics in the crowd. In *SOSP*, pages 441–459, 2017.

11 Elette Boyle, Niv Gilboa, Yuval Ishai, and Victor I. Kolobov. Programmable distributed point functions. In *CRYPTO*, pages 121–151, 2022.

12 David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. In *Communications of the ACM (CACM)*, 1981.

13 Albert Cheu, Adam D. Smith, Jonathan R. Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In *EUROCRYPT*, pages 375–403, 2019.

14 Albert Cheu and Jonathan R. Ullman. The limits of pan privacy and shuffle privacy for learning and estimation. In *STOC*, pages 1081–1094, 2021.

15 Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *FOCS*, pages 41–50, 1995.

16 Richard Cleve. Limits on the security of coin flips when half the processors are faulty. In *STOC*, pages 364–369, 1986.

17 Alex Davidson, Gonçalo Pestana, and Sofía Celi. FrodoPIR: Simple, scalable, single-server private information retrieval. In *PETS*, 2023.

18 Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *USENIX Security*, 2004.

19 Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *PODS*, 2003.

20 Zeev Dvir and Sivakanth Gopi. 2-server pir with sub-polynomial communication. In *STOC*, 2015.

21 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.

22 Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Shuang Song, Kunal Talwar, and Abhradeep Thakurta. Encode, shuffle, analyze privacy revisited: Formalizations and empirical evaluation. *CoRR*, abs/2001.03618, 2020. URL: `https://arxiv.org/abs/2001.03618`.

23 Badih Ghazi, Pasin Manurangsi, Rasmus Pagh, and Ameya Velingker. Private aggregation from fewer anonymous messages. In *EUROCRYPT*, 2020.

24 Dov Gordon and Jonathan Katz. Partial fairness in secure two-party computation. In *EUROCRYPT*, 2010.

25 Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, and Kenneth G Paterson. Learning to reconstruct: Statistical learning theory and encrypted database attacks. In *IEEE S&P*, 2019.

26 Trinabh Gupta, Natacha Crooks, Whitney Mulhern, Srinath Setty, Lorenzo Alvisi, and Michael Walfish. Scalable and private media consumption with Popcorn. In *NSDI*, 2016.

27 Daniel Günther, Maurice Heymann, Benny Pinkas, and Thomas Schneider. Gpu-accelerated pir with client-independent preprocessing for large-scale applications. In *USENIX Security*, 2022.

28 Ryan Henry. Polynomial batch codes for efficient IT-PIR. In *PETS*, 2016.

29 Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. One server for the price of two: Simple and fast single-server private information retrieval. In *USENIX Security*, 2023.

30 Kyle Hogan, Sacha Servan-Schreiber, Zachary Newman, Ben Weintraub, Cristina Nita-Rotaru, and Srinivas Devadas. Shortor: Improving tor network latency via multi-hop overlay routing. In *IEEE S&P*, 2022.

31 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Batch codes and their applications. In *STOC*, 2004.

32 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography from anonymity. In *FOCS*, 2006.

33 Mitsuru Ito, Akira Saito, and Takao Nishizeki. Secret sharing schemes realizing general access structure. In *IEEE Global Telecommunication Conference*, 1987.

34 Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. Generic attacks on secure outsourced databases. In *CCS*, 2016.

35 Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-go-mixes providing probabilistic anonymity in an open system. In *Information Hiding*, 1998.

36 Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *FOCS*, 1997.

37 Donghang Lu, Thomas Yurek, Samarth Kulshreshtha, Rahul Govind, Aniket Kate, and Andrew K. Miller. Honeybadgermpc and asynchromix: Practical asynchronous MPC and its application to anonymous communication. In *CCS*, pages 887–903, 2019.

38 Samir Jordan Menon and David J. Wu. Spiral: Fast, high-rate single-server pir via fhe composition. In *IEEE S&P*, 2022.

39 Tal Moran, Moni Naor, and Gil Segev. An optimally fair coin toss. In *TCC*, 2009.

40 Raphael R. Toledo, George Danezis, and Ian Goldberg. Lower-cost ϵ-private information retrieval. In *PETS*, 2016.

41 Jelle van den Hoof, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *SOSP*, 2015.

42 Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou. All your queries are belong to us: The power of file-injection attacks on searchable encryption. In *USENIX Security*, 2016.

A Background on Multi-Server PIR Constructions

A.1 Two-Server PIR with Additive Shares

The first construction we describe is a PIR scheme from Beimel et al. [9] which uses two non-colluding servers. Figure A.1 contains the full description.

Setup. Consider a field $\mathbb{F}$ within which Σ can be encoded. The Setup algorithm encodes a database $x \in \Sigma^n$ into an m-variate polynomial $P_x \in \mathbb{F}[Z_1, \ldots, Z_m]$ as follows. First, choose m and $d < m$ such that $\binom{m}{d} \geq n$, and let $M = (M_1, \ldots M_n)$ denote a list of n monomials in the variables $Z_1, \ldots, Z_m$ with total degree exactly d and the degree of each variable at most

Let x be a database with size n and $\mathbb{F}$ be a field, where each entry x_i is in $\Sigma = \mathbb{F}$.

- $\mathsf{PIR.Setup}(x) \to P$:
  1. Choose m, d such that $\binom{m}{d} \geq n$.
  2. Let $M = (M_1, \ldots, M_n)$ be a list of n monomials in $\mathbb{F}[Z_1, \ldots, Z_m]$ with total degree d and intermediate degree at most 1. Sort all monomials that have m variables with degree d by a lexicographic order of the variables indices.
  3. Compute $P_x = \sum_{i=1}^{n} x_i M_i \in \mathbb{F}[Z_1, \ldots, Z_m]$.
  4. Compute a $2m$-variate degree-d polynomial P from P_x such that $P(Z_{1,1}, Z_{1,2}, \ldots, Z_{m,1}, Z_{m,2}) = P_x(Z_{1,1} + Z_{1,2}, \ldots, Z_{m,1} + Z_{m,2})$.
  5. Output P.
- $\mathsf{PIR.Query}(i; n) \to ((q_1, q_2), \mathsf{st})$, where $i \in [n]$:
  1. Let $\mathbf{z} = (z_1, \ldots, z_m)$ be the i-th binary vector such that $z_j = 1$ if and only if the monomial M_i contains the variable Z_j.
  2. Let $\mathbf{z}_1 \overset{\$}{\leftarrow} \mathbb{F}_2^m$, $\mathbf{z}_2 \leftarrow \mathbf{z} - \mathbf{z}_1$; and let $q_\ell \leftarrow \mathbf{z}_\ell$ for $\ell = 1, 2$. Set $\mathsf{st} = (\mathbf{z}_1, \mathbf{z}_2)$.
  3. Output $((q_1, q_2), \mathsf{st})$.
- $\mathsf{PIR.Answer}^\ell(P, q_\ell) \to a_\ell$ (for $\ell = 1, 2$):
  1. Let $\{M'_j\}_{j \in [2^m n]}$ be all monomials where the number $Z_{\_,\ell}$ is at least half of the variables.
  2. Output $a_\ell \leftarrow \sum_{j \in [2^m n]} M'_j(q_\ell)$.
- $\mathsf{PIR.Recon}((a_1, a_2), \mathsf{st}) \to x_i$:
  1. Parse st as $(\mathbf{z}_1, \mathbf{z}_2)$.
  2. Compute $x_i \leftarrow a_1(\mathbf{z}_2) + a_2(\mathbf{z}_1)$ (note that a_1 and a_2 are polynomials).
  3. Output x_i.

Construction A.1 A two-server information-theoretic PIR [9].

1. For simplicity, we pick the first n such monomials in lexicographic order of the variable indices. The encoding P_x is now simply the linear combination $P_x = \sum_{i=1}^{n} x_i M_i$.[1]

Query. The Query algorithm starts by encoding the query index $i \in [n]$ into a binary vector $\mathbf{z}^{(i)} = (z_1^{(i)}, \ldots, z_m^{(i)}) \in \{0,1\}^m$ defined such that each $z_j^{(i)} = 1$ if and only if the monomial M_i contains the variable Z_j. Observe here that the Hamming weight of $\mathbf{z}^{(i)}$ is d since the monomials are also of degree d. Such encoding ensures that $P_x(\mathbf{z}_i) = x_i$. Then the sub-queries are generated by splitting $\mathbf{z}^{(i)}$ into two additive shares $\mathbf{z}_1^{(i)} = (z_{1,1}^{(i)}, \ldots, z_{m,1}^{(i)})$ and $\mathbf{z}_2^{(i)} = (z_{1,2}^{(i)}, \ldots, z_{m,2}^{(i)})$, i.e., $\mathbf{z}^{(i)} = \mathbf{z}_1^{(i)} + \mathbf{z}_2^{(i)}$. Here, $\mathbf{z}_\ell^{(i)}$ is sent to the ℓ-th server for $\ell = 1, 2$.

Answer. The Answer^ℓ algorithm run by the servers first views the database encoding P_x as a $2m$-variate polynomial P'_x defined as:

$$P'_x(Z_{1,1}, Z_{1,2}, \ldots, Z_{m,1}, Z_{m,2}) = P_x(Z_{1,1} + Z_{1,2}, \ldots, Z_{m,1} + Z_{m,2}).$$

Now, the ℓ^{th} server selects all the monomial terms in P'_x such that the number of $Z_{\_,\ell}$ (i.e., the variables where the second subscript is ℓ) is at least half of the variables in that term (in the exactly half case, the monomials are split between the two servers in a pre-determined way). Note that the total number of monomials in P'_x is $2^d \cdot n$, so there should be $2^{d-1} \cdot n$ monomials for each server. The ℓ-th server then evaluates its selected monomials at the

[1] One can choose a more complicated encoding in [9] (E1 encoding scheme) that allows better parameters, namely $\sum_{\ell=0}^{d} \binom{m}{\ell} \geq n$.

point $\mathbf{z}_\ell^{(i)}$ and responds with the sum as the answer a_ℓ (which is now a polynomial in the remaining m variables). Further, observe that each monomial in P'_x is of degree d, and so after the server evaluation, the answer polynomial a_ℓ will be of degree at most $d/2$.

Reconstruction. Finally, given answer polynomials a_1, a_2, the client evaluates a_1 at $\mathbf{z}_2^{(i)}$ and a_2 at $\mathbf{z}_1^{(i)}$, and sums up the evaluation results in $\mathbb{F}$ to get $P_x(\mathbf{z}^{(i)}) = x_i$.

Cost. The parameters m and d can be chosen to be both $\Theta(\log n)$ such that $\binom{m}{d} \geq n$. In this case, the query size is $O(\log n)$ (since m elements in $\mathbb{F}_2$ are sent to each server) and the answer size is $O(\sqrt{n})$ (since specifying an m-variate polynomial of degree $d/2$ requires $\binom{m}{d/2} = O(\sqrt{n})$ terms).

```
Let $x$ be a database with size $n$, each entry $x_i$ is in $\Sigma = \mathbb{F}$. There are $s$ non-colluding servers.
- PIR.Setup$(x) \to P$:
    1. Choose $m, d$ such that $\binom{m}{d} \geq n$.
    2. Let $M = (M_1, \ldots, M_n)$ be a list of $n$ monomials in $\mathbb{F}[Z_1, \ldots, Z_m]$ with total degree
       exactly $d$ and intermediate degree at most 1. Sort all monomials that have $m$ variables
       with degree $d$ by a lexicographic order of the variables indices.
    3. Compute $P_x = \sum_{i=1}^n x_i M_i \in \mathbb{F}[Z_1, \ldots, Z_m]$.
    4. Compute a $sm$-variate degree-$d$ polynomial $P$ from $P_x$ such that
       $P(Z_{1,1}, \ldots, Z_{1,s}, \ldots, Z_{m,1} \ldots Z_{m,s}) = P_x(Z_{1,1} + \ldots + Z_{1,s}, \ldots, Z_{m,1} + \ldots + Z_{m,s})$.
    5. Output $P$.

- PIR.Query$(i; n) \to ((q_1, \ldots, q_s), \mathsf{st})$, where $i \in [n]$:
    1. Let $\mathbf{z} = (z_1, \ldots, z_m)$ be the $i$-th binary vector such that $z_j = 1$ if and only if the
       monomial $M_i$ contains the variable $Z_j$.
    2. Let $\mathbf{z}_1, \ldots, \mathbf{z}_{s-1} \xleftarrow{\$} \mathbb{F}_2^m$ and $\mathbf{z}_s \leftarrow \mathbf{z} - \sum_{j=1}^{s-1} \mathbf{z}_j$.
    3. Let $q_\ell \leftarrow (\mathbf{z}_{\ell+1}, \ldots, \mathbf{z}_s, \mathbf{z}_1, \ldots, \mathbf{z}_{\ell-1})$ for $\ell \in [s]$.        // cyclic shift
    4. Set $\mathsf{st} = (\mathbf{z}_1, \ldots, \mathbf{z}_s)$.
    5. Output $((q_1, \ldots, q_s), \mathsf{st})$.
- PIR.Answer$^\ell(P, q_\ell) \to a$, for $\ell \in [s]$:
    1. Let $\{M'_j\}_{j \in [s^m n]}$ be all monomials pre-determined
       such that the number of $Z\_{,\ell}$ is at most $1/s$ fraction.
    2. Output $a \leftarrow \sum_{j \in [s^d n]} M'_j(q_\ell)$.
- PIR.Recon$((a_1, \ldots, a_\ell), \mathsf{st}) \to x_i$:
    1. Parse st as $(\mathbf{z}_1, \ldots, \mathbf{z}_s)$.
    2. Compute $x_i \leftarrow \sum_{\ell \in [s]} a_\ell(\mathbf{z}_1, \ldots, \mathbf{z}_\ell - 1, \mathbf{z}_{\ell+1}, \mathbf{z}_s)$.
    3. Output $x_i$.
```

Construction A.2 An s-server PIR with CNF shares [9]. Note that when $s = 2$, this is simply the 2-server additive PIR.

***k*-server PIR with additive shares.** The above protocol can also be generalized to k servers where the encoding $\mathbf{z}$ is now split into k additive shares. In this case, the servers express the m-variate degree-d polynomial P_x as km-variate degree-d polynomial P'_x. Let $\mathcal{Z}_\ell$ be the set of monomials such that for each monomial, there are more $Z\_{,\ell}$ than $Z\_{,\ell'}$ for any $\ell' \neq \ell$. The set $\mathcal{Z}_\ell$ is assigned to the ℓ-th server. Moreover, the monomials in P'_x but not in any of $\mathcal{Z}\_{,\ell}$'s will be divided to k servers in a pre-determined way. To issue a query for index i, the client encodes it as before to a binary string $\mathbf{z} \in \mathbb{F}_2^m$, and then splits it to k additive shares over $\mathbb{F}_2^m$, denoted as $\mathbf{z}_1, \ldots, \mathbf{z}_k$. The client sends to the ℓ-th server the share $\mathbf{z}_\ell$, and the server evaluates the assigned monomials using $\mathbf{z}_\ell$. The evaluation result is a polynomial of degree $(k-1)d/k$; this implies the answer size (which dominates the communication cost) is $O(n^{(k-1)/k})$.

Observe that using more additive shares gives worse efficiency but better privacy (since collusion between any $k-1$ servers can be tolerated). Efficiency can be significantly improved to $O(n^{1/k})$ using CNF shares [33] (instead of additive shares) where each server is now given a different $(k-1)$-sized subset of the additive shares. This is because the evaluation of P_x at $k-1$ shares results in an answer polynomial of degree at most $O(n^{1/k})$. The efficiency gain, however, comes at the cost of much stronger non-collusion assumption for PIR, namely that no two database servers can collude. Looking ahead, an interesting consequence of using the shuffle model is that our CNF-sharing based construction (Section 6.2) can significantly reduce communication *without* making any non-collusion assumptions on database servers (since there is only one database).

For simplicity, going forward, we will refer to the k-server PIR with additive shares as k-*additive* PIR and its CNF-variant as k-CNF PIR.

A.2 k-Server PIR with Shamir Shares

In this section, we describe the k-server t-private PIR that uses Shamir secret sharing from [9]. Full description is provided in Figure A.3. We also call this the Reed-Muller PIR as it is closely related to Reed-Muller code.

```
Let $x = (x_1, \dots, x_n) \in \mathbb{F}^n$ be a database.

  $\mathsf{PIR.Setup}(x) \to P_x$:
  1. Choose parameters $m, d, k, t$ such that
     $\binom{m+d}{d} \geq n$ and $|\mathbb{F}| > k > td$.
  2. Compute $P_x = \sum_{i=1}^{n} x_i P^{(i)}(z_1, \dots, z_m)$, where $P^{(i)}(\mathsf{PIR.Enc}(i)) = 1$ and $P^{(i)}(\mathsf{PIR.Enc}(j)) = 0$
     for all $i, j \in [n]$ and $i \neq j$.
  3. Output $P_x$.

  $\mathsf{PIR.Query}(i; n) \to ((q_1, \dots, q_k), \mathsf{st})$, where $i \in [n]$:
  1. Run $\mathsf{PIR.Enc}(i)$ and gets $\mathbf{z} \in \mathbb{F}^m$.
  2. Choose a set of degree-$t$ random polynomials $R = (R_1, \dots, R_m)$ such that $R(\mathbf{0}) = \mathbf{z}$.
  3. For $\ell \in [k]$:
       - Randomly choose $r_\ell$ from $\mathbb{F}$.
       - Set $q_\ell \leftarrow Q(r_\ell)$. Note that each $q_\ell \in \mathbb{F}^m$.
  4. Set $\mathsf{st} = (r_1, \dots, r_k)$.
  5. Output $((q_1, \dots, q_k), \mathsf{st})$.

  $\mathsf{PIR.Answer}(P_x, q) \to a$:
  1. Compute $a \leftarrow P_x(a)$.
  2. Output $a$.

  $\mathsf{PIR.Recon}((a_1, \dots, a_k), \mathsf{st}) \to x_i$:
  1. Parse $\mathsf{st} = (r_1, \dots, r_k)$.
  2. Interpolate a degree-$td$ univariate polynomial $R \circ P_x$ from $\{(r_\ell, a_\ell)\}_{\ell=1}^{k}$.
  3. Output $x_i \leftarrow (R \circ P_x)(0)$.
```

Construction A.3 A k-server t-private PIR based on Reed-Muller code [9].

Setup. Consider a field $\mathbb{F}$ within which Σ can be encoded. The Setup algorithm encodes a database $x \in \Sigma^n$ into a polynomial $P_x \in \mathbb{F}[Z_1, \ldots, Z_m]$ as follows: First, choose m and d such that $\binom{m+d}{d} \geq n$ and $|\mathbb{F}| > k > td$ (typically m, d, t are chosen first and then k and the field size $|\mathbb{F}|$ are deteremined accordingly). Let $\alpha_0, \ldots, \alpha_d$ be distinct elements in $\mathbb{F}$ (note that $d < |\mathbb{F}|$). The index i is encoded to the i-th vector $\mathbf{z}^{(i)}$ of the form $(\alpha_{\lambda_1}, \ldots, \alpha_{\lambda_m}) \in \mathbb{F}^m$ where $\sum_{j=1}^{m} \lambda_j \leq d$. There exists a set of polynomials $P^{(i)}(z_1, \ldots, z_m)$ of degree at most d such that $P^{(i)}(\mathbf{z}^{(i)}) = 1$ and $P^{(i)}(\mathbf{z}^{(j)}) = 0$ for all $i, j \in [n]$ and $i \neq j$. The full details of this encoding and the construction of $P^{(i)}$'s are provided in [9, Appendix B].

Query. To generate the sub-queries, after encoding the index i to $\mathbf{z}^{(i)}$, the client first chooses m univariate polynomials $(R_1, \ldots, R_m) = R$ each of degree t such that $R(\mathbf{0}) = (R_1(0), \ldots, R_m(0)) = \mathbf{z}^{(i)}$. It then randomly picks $r_1, \ldots, r_k \in \mathbb{F}$ and computes the sub-query to be sent to the ℓ^{th} server as $q_\ell = R(r_\ell) \in \mathbb{F}^m$.

Answer. The Answer^ℓ algorithm evaluates P_x at q_ℓ and sends back $a_\ell = P_x(q_\ell)$. Note that the answer algorithm for this protocol is the same for all k servers.

Reconstruction. Finally, the Recon algorithm uses Lagrange interpolation on the points (r_1, a_1), …, (r_k, a_k) to compute a degree td polynomial $S = P_x \circ R$; the evaluation $S(0)$ will give the desired database entry x_i. This interpolation is possible when $k > td$ and $|\mathbb{F}| > k$.

Other notation. For a PIR protocol Φ, we use $\mathcal{E}_\Phi$ to denote the encoding space of all indices. We use $\mathcal{Q}_\Phi$ to denote the space of all possible sub-queries (note that $\mathcal{Q}_\Phi$ may not equal $\mathcal{E}_\Phi$). For example, in the two-server construction above, $\mathcal{E}_\Phi$ contains all binary strings with Hamming weight d, and the space $\mathcal{Q}_\Phi$ is $\mathbb{F}_2^m$, i.e, in this case $\mathcal{E}_\Phi \subset \mathcal{Q}_\Phi$.

B Composed PIR Construction

The complete composed PIR construction in the inner-outer paradigm is given in Construction B.1.

ShPIR **Composition.** A shuffle model PIR protocol $\mathsf{ShPIR}(\mathsf{OPIR}, \mathsf{IPIR})$ built using the inner-outer paradigm from a k-server OPIR, and a s-server IPIR is defined as follows:

```
- $\mathsf{ShPIR.Setup}(x) \to P$:
  1. Let $P_x \leftarrow \mathsf{OPIR.Setup}(x)$.
  2. Define a database $x'$ of size $n'$ as follows:
     - Let $n^* = |\mathcal{Q}_{\mathsf{OPIR}}|$ and let $L = (L_1, \dots, L_{n^*})$ denote the sorting of the sub-query
       space $\mathcal{Q}_{\mathsf{OPIR}}$.
     - If the Answer algorithm is the same for all OPIR servers:
         For all $i \in [n^*]$, let $x'_i \leftarrow \mathsf{OPIR.Answer}(P_x, L_i)$.
         As a result, $x'$ is of size $n' = n^*$.
     - If the Answer algorithm is different for the $k$ OPIR servers:
         For $i \in [n^*], \ell \in [k]$: let $x'_{i+n'\cdot(\ell-1)} \leftarrow \mathsf{OPIR.Answer}^{\ell}(P_x, L_i)$.
         As a result, $x'$ is of size $n' = kn^*$.
  3. Run $\mathsf{IPIR.Setup}(x')$ and output its result as $P$.

- $\mathsf{ShPIR.Query}(i; n) \to (q_1, \dots, q_h)$, where $i \in [n]$ and $h = k(s+1)$:
  1. Initialize $(u_{\ell,j})_{\ell\in[k], j\in[s]}$.
  2. Let $(q'_1, \dots, q'_k) \leftarrow\$ \mathsf{OPIR.Query}(i; n)$.
  3. For $\ell \in [k]$,
     - If the Answer algorithm is the same for all $k$ OPIR servers:
         Map $q'_\ell$ to the corresponding index $i'_\ell \in [n']$,
         i.e., $x_{i'_\ell} = \mathsf{OPIR.Answer}(P_x, q'_\ell)$.
     - If the Answer algorithm is different for the $k$ OPIR servers:
         Map $q'_\ell$ to the corresponding index $i'_\ell \in [kn']$,
         i.e., $x_{i'_\ell} = \mathsf{OPIR.Answer}^{\ell}(P_x, q'_\ell)$.
     - Let $(\widetilde{q}_1, \dots, \widetilde{q}_s) \leftarrow\$ \mathsf{IPIR.Query}(i'_\ell; n')$.
     - Set $(u_{\ell,1}, \dots, u_{\ell,s}) \leftarrow (\widetilde{q}_1, \dots, \widetilde{q}_s)$.
  4. Let $(r_1, \dots, r_k) \overset{\$}{\leftarrow} \mathcal{Q}_{\mathsf{OPIR}}$.    // dummies
  5. Output $(u_{1,1}, \dots, u_{k,s}, r_1, \dots, r_k)$.

- $\mathsf{ShPIR.Answer}(P, q) \to a$:
  1. If IPIR has the same Answer algorithms for server, return $a = \mathsf{IPIR.Answer}(P, q)$;
     otherwise return

       $$a = \left\{(\mathsf{IPIR.Answer}^{\ell}(P, q), \mathsf{label}\ \ell)\right\}_{\ell\in[s]}.$$

- $\mathsf{ShPIR.Recon}(a_1, \dots, a_h) \to x_i$:
  1. Initialize $(v_{\ell,j})_{\ell\in[k], j\in[s]}$ and $(a'_\ell)_{\ell\in[k]}$.
  2. For $\ell \in [k]$, $j \in [s]$:
     - Let $a_{(\ell-1)\cdot k+j}$ be the answer to sub-query $q_{(\ell-1)\cdot k+j}$, namely $u_{\ell,j}$.
     - If IPIR has different Answer algorithms for the servers, parse $a_{(\ell-1)\cdot k+j}$ as

         $$\{(\widetilde{a}_1, \mathsf{label}\ 1), \dots, (\widetilde{a}_s, \mathsf{label}\ s)\},$$ let $v_{\ell,j} := \widetilde{a}_j$ (whose associated label is $j$).

     - If IPIR has the same Answer algorithms for the servers, let $v_{\ell,j} = a_{(\ell-1)\cdot k+j}$.
  3. For $\ell \in [k]$:
     - $a'_\ell \leftarrow \mathsf{IPIR.Recon}(v_{\ell,1}, \dots, v_{\ell,s})$.
  4. Output $x_i \leftarrow \mathsf{OPIR.Recon}(a'_1, \dots, a'_k)$.
```

Construction B.1 Composed ShPIR built using the inner-outer paradigm.

Improved Trade-Offs Between Amortization and Download Bandwidth for Linear HSS

Keller Blackwell
Department of Computer Science, Stanford University, CA, USA

Mary Wootters
Departments of Computer Science and Electrical Engineering, Stanford University, CA, USA

Abstract

A *Homomorphic Secret Sharing* (HSS) scheme is a secret-sharing scheme that shares a secret x among s servers, and additionally allows an output client to reconstruct some function $f(x)$ using information that can be locally computed by each server. A key parameter in HSS schemes is *download rate*, which quantifies how much information the output client needs to download from the servers. Often, download rate is improved by *amortizing* over ℓ instances of the problem, making ℓ also a key parameter of interest.

Recent work [23] established a limit on the download rate of linear HSS schemes for computing low-degree polynomials and constructed schemes that achieve this optimal download rate; their schemes required amortization over $\ell = \Omega(s \log(s))$ instances of the problem. Subsequent work [6] completely characterized linear HSS schemes that achieve optimal download rate in terms of a coding-theoretic notion termed *optimal labelweight codes*. A consequence of this characterization was that $\ell = \Omega(s \log(s))$ is in fact necessary to achieve optimal download rate.

In this paper, we characterize *all* linear HSS schemes, showing that schemes of any download rate are equivalent to a generalization of optimal labelweight codes. This equivalence is constructive and provides a way to obtain an explicit linear HSS scheme from *any* linear code. Using this characterization, we present explicit linear HSS schemes with slightly sub-optimal rate but with much improved amortization $\ell = O(s)$. Our constructions are based on algebraic geometry codes (specifically Hermitian codes and Goppa codes).

2012 ACM Subject Classification Theory of computation → Cryptographic primitives; Theory of computation → Error-correcting codes

Keywords and phrases Error Correcting Codes, Homomorphic Secret Sharing

Digital Object Identifier 10.4230/LIPIcs.ITC.2024.7

Related Version *Full Version*: `https://arxiv.org/abs/2403.08719`

Funding *Keller Blackwell*: KB is supported by a National Science Foundation Graduate Research Fellowship and by a graduate fellowship award from Knight-Hennessy Scholars at Stanford University. KB's work is supported partially supported by NSF grant CCF-2231157.
Mary Wootters: MW's work was partially supported by NSF grants CCF-1844628, CCF-2133154, and CCF-2231157.

Acknowledgements We thank the anonymous referees for helpful feedback.

1 Introduction

A *Homomorphic Secret Sharing* (HSS) scheme is a secret sharing scheme that supports computation on top of the shares [4, 12, 13]. Homomorphic Secret Sharing has been a useful primitive in cryptography, with applications ranging from private information retrieval to secure multiparty computation (see, e.g., [9, 13]).

In this work, we focus on information-theoretically secure HSS schemes for the class of degree d, m-variate polynomials. Suppose m secrets, $x_1, \dots, x_m$, are shared independently with a t-private secret sharing scheme Share[1]and that server j receives the m shares $y_{k,j}$ for $k \in [m]$. Denote by $\mathrm{POLY}_{d,m}(\mathbb{F}) \subseteq \mathbb{F}[X_1, \dots, X_m]$ an arbitrary set of degree d, m-variate polynomials. Given a polynomial $f \in \mathrm{POLY}_{d,m}(\mathbb{F})$, each server j does some local computation on its shares $y_{k,j}$ for $k \in [m]$ to obtain an *output share* $z_j = \mathsf{Eval}(f, j, (y_{1,j}, \dots, y_{m,j}))$. An *output client* receives the output shares $z_1, \dots, z_s$ and runs a recovery algorithm Rec to obtain $f(x_1, \dots, x_m) = \mathsf{Rec}(z_1, \dots, z_s)$. The HSS scheme π is given by the tuple of functions $(\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$; see Definition 7 for a formal definition.

Parameters of interest

A key parameter of interest in an HSS scheme is the *download rate* (Definition 12), which is the ratio of the number of bits in $f(x)$ to the number of bits in all of the output shares z_j. Ideally this rate would be as close to 1 as possible, because that would mean that the output client does not have to download too much more information than it wishes to compute.

Another parameter of interest is the *amortization* that the scheme uses. As in previous work [23, 6], we consider HSS schemes for low-degree polynomials that amortize over ℓ instances of the problem. This means that we have ℓ batches of m secrets, $x_k^{(i)}$ for $i \in [\ell]$ and $k \in [m]$, and ℓ polynomials $f_1, \dots, f_\ell$. Each of these $m\ell$ secrets is shared independently, as before, but the output shares z_j are allowed to depend on *all* ℓ batches. Then the output client is responsible for computing $f_i(x_1^{(i)}, \dots, x_m^{(i)})$ for all $i \in [\ell]$.

Trade-offs between download rate and amortization

[23, 6] previously studied the optimal download rate possible for linear HSS schemes[2], and also studied what amount of amortization is necessary to obtain this optimal rate. In this work, we show that by backing off from the optimal rate by a small amount, one can get asymptotic improvements in the amortization parameter.

In more detail, [23] showed that for t-private, s-server linear HSS schemes for m-variate degree-d polynomials, the best download rate possible is $1 - dt/s$. They achieved this download rate with schemes that had amortization $\ell = \Omega(s \log(s))$. In follow-up work, [6] showed that in fact amortization $\ell = \Omega(s \log(s))$ was *necessary* to achieve the optimal download rate of $1 - dt/s$. Their result followed from a characterization optimal-rate linear HSS schemes in terms of a coding theoretic notion they introduce, termed *optimal labelweight codes*.

In our work, informally, we show that by backing off from the optimal rate by a small amount, we are able to get asymptotic improvements in the amortization required; in some cases we need only $\ell = O(s)$. We obtain this by generalizing the characterization from [6] to *all* HSS schemes, not just optimal ones. We discuss our main results below in more detail.

[1] A t-private secret sharing scheme shares a secret x among s servers by computing s *shares*, $\mathsf{Share}(x) = (y_1, \dots, y_s)$. The t-privacy guarantee means that no t of the servers should be able to learn anything about x given their shares.

[2] A *linear* HSS scheme is a scheme where both Share and Rec are linear over some field $\mathbb{F}$. Note that Eval need not be linear.

1.1 Main Results

For all of our results, we consider *CNF sharing* [27] (see Definition 16). It is known that CNF sharing is universal for linear secret sharing schemes, in that t-CNF shares can be locally converted to shares of *any* linear t-private secret sharing scheme [18].

Main Contributions

1. **A complete characterization of the Rec functions for *all* linear HSS schemes for $\mathbf{POLY}_{d,m}(\mathbb{F})$.** As mentioned above, [6] gives a characterization of *optimal*-download-rate linear HSS schemes in terms of codes with good labelweight. In our work, we extend that characterization to *all* linear HSS schemes.
   Our characterization is constructive, and in particular it gives an efficient algorithm to convert any code with good labelweight into a linear HSS scheme, and vice-versa.
2. **Improved amortization without much loss in rate.** The work [6] showed that to achieve optimal rate, it was necessary to have amortization $\ell = \Omega(s \log s)$. Leveraging our characterization from Item 1, we give efficient constructions of linear HSS schemes that achieve *near*-optimal download rate while requiring amortization parameter only $\ell = O(s)$. We compute the parameters of our constructions in practical parameter regimes and show that our schemes achieve a near order-of-magnitude savings in amortization parameter, even for reasonable values of s, d, m.

   We describe our results in greater detail below.

(1) Characterization of arbitrary-rate linear HSS schemes

Theorem 2 below is a characterization of *all* linear HSS schemes for $\mathrm{POLY}_{d,m}(\mathbb{F})$. In particular, our characterization extends that of [6], which only characterized optimal-rate linear HSS schemes. We show that the Rec algorithms for such schemes (with CNF sharing) are equivalent to a class of linear codes with sufficiently good *labelweight*, a generalization of Hamming distance that was introduced by [6].

▶ **Definition 1** (Labelweight). *Let $\mathcal{C} \subseteq \mathbb{F}^n$ be a linear code of dimension ℓ. Let $\mathcal{L} : [n] \to [s]$ be any surjective function, which we refer to as a* labeling *function. The* labelweight *of $\mathbf{c} \in \mathcal{C}$ is the number of distinct labels that the support of $\mathbf{c}$ touches:*

$$\Delta_{\mathcal{L}}(\mathbf{c}) = |\{\mathcal{L}(i) \,:\, i \in [n], c_i \neq 0\}|.$$

The labelweight *of $\mathcal{C}$ is the minimum labelweight of any nonzero codeword:*

$$\Delta_{\mathcal{L}}(\mathcal{C}) = \min_{c \in \mathcal{C} \setminus \{0\}} \Delta_{\mathcal{L}}(\mathbf{c}).$$

In particular, if $s = n$ and $\mathcal{L}(j) = j$ for all $j \in [n]$, then $\Delta_{\mathcal{L}}(\mathcal{C})$ is just the minimum Hamming distance of $\mathcal{C}$. Thus, the labelweight of a code generalizes the standard notion of distance.

Our main characterization theorem is the following.

▶ **Theorem 2** (Linear HSS schemes are equivalent to labelweight codes. (Informal, see Theorem 18)). *Let $\pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ be a t-private, s-server linear HSS for $\mathrm{POLY}_{d,m}(\mathbb{F})$ with download rate R and amortization parameter ℓ. Let $G \in \mathbb{F}^{\ell \times (\ell/R)}$ be the matrix that represents* Rec *(see Observation 10). Then there is some labeling function $\mathcal{L}$ so that G is the generator matrix for a code $\mathcal{C}$ of dimension ℓ, with rate R and with $\Delta_{\mathcal{L}}(\mathcal{C}) \geq dt + 1$.*

Conversely, suppose that there is a labeling function $\mathcal{L} : [n] \to [s]$ and a linear code $\mathcal{C} \subseteq \mathbb{F}^n$ of dimension ℓ with rate R and $\Delta_{\mathcal{L}}(\mathcal{C}) \geq dt + 1$. Then any generator matrix G of $\mathcal{C}$ describes a linear reconstruction algorithm Rec *for an s-server t-private linear HSS for* $\mathrm{POLY}_{d,m}(\mathbb{F})$ *that has download rate R and amortization parameter ℓ.*

We remark that the converse direction is constructive: given the description of such a code $\mathcal{C}$, the proof (see Theorem 20) gives an efficient construction of the Eval function as well as the Rec function.

(2) Achieving practical trade-off between download rate and amortization parameter

Using the complete characterization of all linear HSS schemes, we construct linear HSS schemes that achieve near-optimal download rate at amortization parameters that are strictly linear in in the number of servers s. Through the lens of Theorem 2, this is equivalent to constructing high-labelweight, high-rate linear codes.

While the construction of [6] use Reed-Solomon codes as a starting point to constructing optimal rate linear HSS schemes, we use two well-studied families of algebraic geometry (AG) codes – Hermitian codes and Goppa codes. For any code $\mathcal{C}$, observe that the minimum labelweight $\Delta_{\mathcal{L}}(\mathcal{C})$ is sharply upper-bounded by the code's minimum distance. Therefore, so as to maximize labelweight, we use the trivial labeling scheme where $n = s$ and $\mathcal{L} : [n] \to [s], x \mapsto x$ is the identity function. Such labelweight codes, though straightforward, yield linear HSS schemes with attractive parameters for realistic server counts. Furthermore, their intuitive construction underscores the fundamental relationship between classical codes and linear HSS. We loosely summarize these results in Table 1; see Theorems 22, 27 for additional details and formal statements.

Table 1 Comparing our AG-based constructions to [23], [6].

	[23],[6]	Hermitian-based HSS	Goppa-based HSS
Download Rate	$1 - dt/s$	$1 - dt/s - O(s^{-1/3})$	$1 - (dt/s) \cdot O(\log(dt))$
Amortization	$(s - dt)\log(s)$	$s - dt - O(s^{2/3})$	$s - dt \cdot O(\log(dt))$

Furthermore, for completeness we show in Appendix A that, perhaps surprisingly, a random coding approach does not lead to amortization savings over [23], [6], even backing off from the optimal rate. More precisely, linear HSS schemes instantiated from random labelweight codes result in HSS schemes with amortization parameter at least $\Omega(s \log(s))$. This motivates additional study of linear codes with algebraic structure as a basis for linear HSS with attractive parameters.

1.2 Technical Overview

In this section, we give a high-level overview of the techniques underpinning Theorem 2, which states that any t-private, s-server linear HSS scheme for $\mathrm{POLY}_{d,m}(\mathbb{F})$ is equivalent to a labelweight code with minimum labelweight $\geq dt + 1$.

To give some intuition for the connection, we recount the simplest non-trivial case of the forward direction, which was proven by [6]. We consider *HSS for concatenation* [23]: ℓ secrets $\mathbf{x} = \left(x^{(1)}, \ldots, x^{(\ell)}\right) \in \mathbb{F}^\ell$ are shared independently among s servers who in turn communicate them to an output client. The objective is for the output client to download as little information as possible - in particular, significantly less than the naive solution of simply downloading $t + 1$ shares of each secret.

Let $\mathbf{z} \in \mathbb{F}^n$ be the n-tuple of $\mathbb{F}$-symbols downloaded by the output client; since the output client instantiates a linear reconstruction algorithm Rec, there exists $G \in \mathbb{F}^{\ell \times n}$ such that $G\mathbf{z} = \mathbf{x}$. Define a labeling function $L : [n] \to [s]$ satisfying the property that for all $i \in [n]$, $L(i) = r \in [s]$ if and only if $\mathbf{z}_i$ was downloaded from server r.

▷ Claim 3. The rows of G generate a linear code $\mathcal{C}$ satisfying $\Delta_{\mathcal{L}}(\mathcal{C}) \geq t + 1$.

Proof. Suppose towards a contradiction that for some non-zero $\mathbf{m} \in \mathbb{F}^\ell$, $\Delta_{\mathcal{L}}(\mathbf{m}G) \leq t$. Then $\mathbf{m}G\mathbf{x} = \mathbf{c}\mathbf{x}$ (for some non-zero $\mathbf{c} \in \mathbb{F}^n$) is a linear combination of secrets recoverable by a set of t servers, contradicting the t-privacy they were originally secret shared with. ◁

In this example, the function evaluated on the secret shares is identity; the generalization of this example requires consideration of more general functions, but the fundamental principle is similar.

The converse requires showing that any labelweight code $\mathcal{C}$ implies a linear HSS scheme. In the setting of optimal download rate, [6] leveraged the specific properties of optimal labelweight codes in order to prove this; their work relied on the fact that optimal labelweight codes are highly structured. In particular, [6] showed that, in the optimal download rate setting:

(i) the output client must download an equal number of symbols from each server; and

(ii) up to elementary row operations, the matrix parameterizing the output client's linear Rec algorithm is a rectangular array of invertible matrices, with the property that any square sub-array is itself invertible.

These strong symmetry properties are key to the equivalence result of [6]. However, in our setting, where we do not assume that the optimal download rate is attained, *neither of the aforementioned properties hold.* Our result thus requires proving additional properties of labelweight codes.

1.3 Related Work

Though linear HSS schemes are implicit in classical protocols for secure multi-party computation and private information retrieval [4, 3, 15, 19, 1, 2, 16], the systematic study of HSS was introduced by [13]. Most HSS schemes reply on cryptographic hardness assumptions [10, 21, 11, 12, 22, 13, 14, 8, 17, 28, 29, 20].

In contrast, the HSS schemes presented in this work are *information-theoretically secure.* The information-theoretic setting was explored in [13] and was further studied in [23] and [6]; these latter two works are the closest to our work, and we discuss them more below.

The work of [23] focused on the download rate of information-theoretically secure HSS schemes (both linear and non-linear) and proved a tight impossibility result regarding the highest download rate achievable by linear HSS schemes. They paired this with an explicit construction that showed a large amortization parameter is a sufficient condition for a linear HSS scheme to achieve optimal download rate.

▶ **Theorem 4** **([23]).** *Let $s, d, t \in \mathbb{Z}^+$ such that $s > dt$. Let π be a t-private, s-server linear HSS scheme for* $\mathrm{POLY}_{d,m}(\mathbb{F})$*. Then* $\mathsf{DownloadRate}(\pi) \leq 1 - dt/s$*.*

Furthermore, for all integers $j \geq \log_{|\mathbb{F}|}(s)$, there exists a t-private, s-server linear HSS π satisfying $\mathsf{DownloadRate}(\pi) = 1 - dt/s$ *with amortization parameter $\ell = j(s - dt)$.*

The work [6] focused on linear schemes with *optimal* download rate, meeting the bound showed in [23]. They proved that in fact a large amortization parameter is necessary for a linear HSS scheme to achieve optimal download rate.

▶ **Theorem 5** ([6]). *There exists a t-private, s-server linear HSS scheme for $POLY_{d,m}(\mathbb{F})$ with download rate $(s-dt)/s$ and amortization parameter ℓ only if $\ell = j(s-dt)$ for some $j \in \mathbb{Z}+$ satisfying*

$$j \geq \lceil \max \left\{ \log_q(s-dt+1), \log_q(dt+1) \right\} \rceil.$$

The key technique in their proof was showing that optimal-rate linear HSS is in fact equivalent to optimal labelweight linear codes; more precisely, they showed the following theorem.

▶ **Theorem 6** ([6]). *There exists a t-private, s-server linear HSS scheme for $POLY_{d,m}(\mathbb{F})$ with download rate $(s-dt)/s$ and amortization parameter ℓ if and only if there exists a linear code $\mathcal{C} \subseteq \mathbb{F}^n$ with information rate $(s-dt)/s$ and dimension ℓ; and surjection $L : [n] \to [s]$ such that $\Delta_L(\mathcal{C}) \geq dt+1$.*

Our work extends the characterization of [6] to *all* linear HSS schemes with arbitrary download rate; in particular, we show that arbitrary-rate linear HSS schemes are equivalent to a broader class of labelweight codes than those considered by [6]. Though our proof syntactically resembles that of Theorem 6 from [6], as discussed in Section 1.2, we need to overcome additional technical difficulties introduced by the lack of strong symmetries in the more general arbitrary-rate setting. Furthermore, we present explicit constructions that approach the optimal download rate from Theorem 15, while asymptotically improving the amortization parameter ℓ.

1.4 Organization

In Section 2, we set notation and record a few formal definitions that we will need. In Section 3, we show that linear HSS schemes (with arbitrary download rate) are equivalent to codes with sufficient labelweight: Lemma 19 establishes that HSS schemes imply codes with sufficient labelweight, and Theorem 20 constructively establishes the converse.

In Section 4, we derive labelweight codes from Hermitian codes and construct the corresponding linear HSS scheme by Theorem 20. We formally state its parameters in Theorem 22 and compare its performance against constructions from [23] and [6]. In Section 5, we do the same but use Goppa codes as the basis for a family of labelweight codes.

2 Preliminaries

We begin by setting notation and the basic definitions that we will need throughout the paper. (We note that these definitions and notation closely follow that of [23] and [6]).

Notation. For $n \in \mathbb{Z}^+$, we denote by $[n]$ the set $\{1, 2, \ldots, n\}$. We use bold symbols (e.g., $\mathbf{x}$) to denote vectors. For an object w in some domain $\mathcal{W}$, we use $\|w\| = \log_2(|\mathcal{W}|)$ to denote the number of bits used to represent w.

2.1 Homomorphic Secret Sharing

We consider homomorphic secret sharing (HSS) schemes with m inputs and s servers; each input is shared independently. We denote by $\mathcal{F} = \{f : \mathcal{X}^m \to \mathcal{O}\}$ the class of functions we wish to compute, where $\mathcal{X}$ and $\mathcal{O}$ are input and output domains, respectively.

▶ **Definition 7** (HSS). *Given a collection of s servers and a function class $\mathcal{F} = \{f : \mathcal{X}^m \to \mathcal{O}\}$, consider a tuple $\pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$, where $\mathsf{Share} : \mathcal{X} \times \mathcal{R} \to \mathcal{Y}^s$, $\mathsf{Eval} : \mathcal{F} \times [s] \times \mathcal{Y} \to \mathcal{Z}^*$, and $\mathsf{Rec} : \mathcal{Z}^* \to \mathcal{O}$ are as follows*[3]*:*

- $\mathsf{Share}(x_i, r_i)$*: For $i \in [m]$, Share takes as input a secret $x_i \in \mathcal{X}$ and randomness $r_i \in \mathcal{R}$; it outputs s shares $(y_{i,j} : j \in [s]) \in \mathcal{Y}^s$. We refer to the $y_{i,j}$ as* input shares*; server j holds shares $(y_{i,j} : i \in [m])$.*
- $\mathsf{Eval}\left(f, j, (y_{1,j}, y_{2,j}, \dots, y_{m,j})\right)$*: Given $f \in \mathcal{F}$, server index $j \in [s]$, and server j's input shares $(y_{1,j}, y_{2,j}, \dots, y_{m,j})$, Eval outputs $z_j \in \mathcal{Z}^{n_j}$, for some $n_j \in \mathbb{Z}$. We refer to the z_j as* output shares.
- $\mathsf{Rec}(z_1, \dots, z_s)$*: Given output shares $z_1, \dots, z_s$, Rec computes $f(x_1, \dots, x_m) \in \mathcal{O}$.*

We say that $\pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ is a s-server HSS scheme for $\mathcal{F}$ if the following requirements hold:

- ***Correctness:*** *For any m inputs $x_1, \dots, x_m \in \mathcal{X}$ and $f \in \mathcal{F}$,*

$$\Pr_{\mathbf{r} \in \mathcal{R}^m}\left[\mathsf{Rec}(z_1, \dots, z_s) = f(x_1, \dots, x_m) : \begin{array}{l} \forall i \in [m],\ (y_{i,1}, \dots, y_{i,s}) \leftarrow \mathsf{Share}(x_i, r_i) \\ \forall j \in [s],\ z_j \leftarrow \mathsf{Eval}\left(f, j, (y_{1,j}, \dots, y_{m,j})\right) \end{array}\right] = 1$$

  Note that the random seeds $r_1, \dots, r_m$ are independent.
- ***Security:*** *Fix $i \in [m]$; we say that π is t-*private *if for every $T \subseteq [s]$ with $|T| \le t$ and $x_i, x_i' \in \mathcal{X}$, $\mathsf{Share}(x_i)|_T$ has the same distribution as $\mathsf{Share}(x_i')|_T$, over the randomness $\mathbf{r} \in \mathcal{R}^m$ used in Share.*

▶ Remark 8. We remark that in the definition of HSS, the reconstruction algorithm Rec does *not* need to know the identity of the function f being computed, while the Eval function does. In some contexts it makes sense to consider an HSS scheme for $\mathcal{F} = \{f\}$, in which case f is fixed and known to all. Our results in this work apply for general collections $\mathcal{F}$ of low-degree, multivariate polynomials, and in particular cover both situations.

We focus on *linear* HSS schemes, where both Share and Rec are $\mathbb{F}$-linear over some finite field $\mathbb{F}$; note that Eval need not be linear.

▶ **Definition 9** (Linear HSS). *Let $\mathbb{F}$ be a finite field.*

- *We say that an s-server HSS $\pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ has* linear reconstruction *if:*
  - *$\mathcal{Z} = \mathbb{F}$, so each output share $z_i \in \mathbb{F}^{n_i}$ is a vector over $\mathbb{F}$;*
  - *$\mathcal{O} = \mathbb{F}^o$ is a vector space over $\mathbb{F}$; and*
  - *$\mathsf{Rec} : \mathbb{F}^{\sum_i n_i} \to \mathbb{F}^o$ is $\mathbb{F}$-linear.*
- *We say that π has* linear sharing *if $\mathcal{X}$, $\mathcal{R}$, and $\mathcal{Y}$ are all $\mathbb{F}$-vector spaces, and Share is $\mathbb{F}$-linear.*
- *We say that π is* linear *if it has both linear reconstruction and linear sharing. Note there is no requirement for Eval to be $\mathbb{F}$-linear.*

The assumption of linearity implies that the function Rec can be represented by a matrix, as per the following observation that was also used by [6].

▶ **Observation 10** ([6]). *Let ℓ, t, s, d, m, n be integers. Let $\pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ be a t-private, s-server HSS for some function class $\mathcal{F} \subseteq \mathrm{POLY}_{d,m}(\mathbb{F})^\ell$ with linear reconstruction $\mathsf{Rec} : \mathbb{F}^n \to \mathbb{F}^\ell$.*

[3] By $\mathcal{Z}^*$, we mean a vector of some number of symbols from $\mathcal{Z}$.

Then there exists a matrix $G_\pi \in \mathbb{F}^{\ell \times n}$ so that, for all $f \in \mathcal{F}$ and for all secrets $\mathbf{x} \in (\mathbb{F}^m)^\ell$, there exists some $\mathbf{z} \in \mathcal{F}^n$ such that

$$\mathsf{Rec}(\mathbf{z}) = G_\pi \mathbf{z} = f(\mathbf{x}) = \left[f_1(\mathbf{x}^{(1)}),\ f_2(\mathbf{x}^{(2)}), \ldots,\ f_\ell(\mathbf{x}^{(\ell)}) \right]^\mathsf{T}.$$

For a linear HSS π, we call G_π as in the observation above the *reconstruction matrix* corresponding to Rec. It was shown in [6] that any such reconstruction matrix must be full rank.

▶ **Lemma 11** ([6]). *Let t, s, d, m, ℓ be positive integers so that $m \geq d$ and $n \geq \ell$, and let π be a t-private s-server linear HSS for some $\mathcal{F} \subseteq \mathrm{POLY}_{d,m}(\mathbb{F})$, so that $\mathcal{F}$ contains an element $(f_1, \ldots, f_\ell)$ where for each $i \in [\ell]$, f_i is non-constant. Then $G_\pi \in \mathbb{F}^{\ell \times n}$ has rank ℓ.*

Finally, we formally define the *download rate* of an HSS scheme.

▶ **Definition 12** (Download cost, dowload rate). *Let s, t be integers and let $\mathcal{F}$ be a class of functions with input space $\mathcal{X}^m$ and output space $\mathcal{O}$. Let π be an s-server t-private HSS for $\mathcal{F}$. Let $z_i \in \mathcal{Z}^{n_i}$ for $i \in [s]$ denote the output shares.*

- *The* download cost *of π is given by*

$$\mathit{DownloadCost}(\pi) := \sum_{i \in [s]} \|z_i\|,$$

where we recall that $\|z_i\| = n_i \log_2 |\mathcal{Z}|$ denotes the number of bits used to represent z_i.

- *The* download rate *of π is given by*

$$\mathit{DownloadRate}(\pi) := \frac{\log_2 |\mathcal{O}|}{\mathit{DownloadCost}(\pi)}.$$

Thus, the download rate is a number between 0 and 1, and we would like it to be as close to 1 as possible.

2.2 Polynomial Function Classes

Throughout, we will be interested in classes of functions $\mathcal{F}$ comprised of low-degree polynomials.

▶ **Definition 13.** *Let $m > 0$ be an integer and $\mathbb{F}$ be a finite field. We define*

$$\mathrm{POLY}_{d,m}(\mathbb{F}) := \{ f \in \mathbb{F}[X_1, \ldots, X_m] : \deg(f) \leq d \}$$

to be the class of all m-variate polynomials of degree at most d, with coefficients in $\mathbb{F}$.

We are primarily interested in *amortizing* HSS computation over ℓ instances of $\mathrm{POLY}_{d,m}(\mathbb{F})$, as discussed in the Introduction. We can capture this as part of Definition 7 by taking the function class $\mathcal{F}$ to be (a subset of) $\mathrm{POLY}_{d,m}(\mathbb{F})^\ell$ for some $\ell \in \mathbb{Z}^+$. Note that this corresponds to the amortized setting discussed in the Introduction.

▶ **Definition 14.** *Let $\mathcal{F} \subseteq \mathrm{POLY}_{d,m}(\mathbb{F})^\ell$. We say that $\mathcal{F}$ is* non-trivial *if there exists some $\mathbf{f} = (f_1, \ldots, f_\ell) \in \mathcal{F}$ so that for all $i \in [\ell]$, f_i contains a monomial with at least d distinct variables.*

The work [23] showed that any linear HSS scheme for $\mathrm{POLY}_{d,m}(\mathbb{F})^\ell$ (for any ℓ) can have download rate at most $(s - dt)/s$: We recall the following theorem from [23].

▶ **Theorem 15** ([23]). *Let t, s, d, m, ℓ be positive integers so that $m \geq d$. Let $\mathbb{F}$ be any finite field and π be a t-private s-server linear HSS scheme for $\mathrm{POLY}_{d,m}(\mathbb{F})^\ell$. Then $dt < s$, and $\mathit{DownloadRate}(\pi) \leq (s - dt)/s$.*

2.3 CNF Sharing

The main Share function that we consider in this work is *CNF sharing* [27].

▶ **Definition 16** (t-private CNF sharing)**.** *Let $\mathbb{F}$ be a finite field. The t-private, s-server CNF secret-sharing scheme over $\mathbb{F}$ is a function* $\mathsf{Share} : \mathbb{F} \times \mathbb{F}^{\binom{s}{t}-1} \to \left(\mathbb{F}^{\binom{s-1}{t}}\right)^s$ *that shares a secret $x \in \mathbb{F}$ as s shares $y_j \in \mathbb{F}^{\binom{s-1}{t}}$, using $\binom{s}{t} - 1$ random field elements, as follows.*

Let $x \in \mathbb{F}$, and let $\mathbf{r} \in \mathbb{F}^{\binom{s}{t}-1}$ be a uniformly random vector. Using $\mathbf{r}$, choose $y_T \in \mathbb{F}$ for each set $T \subseteq [s]$ of size t, as follows: The y_T are uniformly random subject to the equation

$$x = \sum_{T\subseteq[s]:|T|=t} y_T.$$

Then for all $j \in [s]$, define $\mathsf{Share}(x, \mathbf{r})_j = (y_T \,:\, j \notin T) \in \mathbb{F}^{\binom{s-1}{t}}$.

We observe that CNF-sharing is indeed t-private. Any $t+1$ servers between them hold all of the shares y_T, and thus can reconstruct $x = \sum_T y_T$. In contrast, any t of the servers (say given by some set $S \subseteq [s]$) are missing the share y_S, and thus cannot learn anything about x.

The main reason we focus on CNF sharing is that it is *universal* for linear secret sharing schemes:

▶ **Theorem 17** ([18])**.** *Suppose that $x \in \mathbb{F}$ is t-CNF-shared among s servers, so that server j holds $y_j \in \mathbb{F}^{\binom{s-1}{t}}$, and let* Share' *be any other linear secret-sharing scheme for s servers that is (at least) t-private. Then the shares y_j are locally convertible into shares of* Share'. *That, is there are functions $\phi_1, \dots, \phi_s$ so that $(\phi_1(y_1), \dots, \phi_s(y_s))$ has the same distribution as* $\mathsf{Share}'(x, \mathbf{r})$ *for a uniformly random vector $\mathbf{r}$.*

2.4 Linear Codes

Throughout, we will be working with *linear codes* $\mathcal{C} \subset \mathbb{F}^n$, which are just subspaces of $\mathbb{F}^n$. For a linear code $\mathcal{C} \subseteq \mathbb{F}^n$ of dimension ℓ, a matrix $G \in \mathbb{F}^{\ell\times n}$ is a *generator matrix* for $\mathcal{C}$ if $\mathcal{C} = \mathrm{rowSpan}(G)$. Note that generator matrices are not unique. The *rate* of a linear code $\mathcal{C} \subset \mathbb{F}^n$ of dimension ℓ is defined as $\mathsf{Rate}(\mathcal{C}) := \frac{\ell}{n}$.

3 Equivalence of Linear HSS and Labelweight Codes

In this section we show that linear HSS schemes for low-degree multivariate polynomials are equivalent to linear codes with sufficient labelweight. Concretely, we have the following theorem, which formalizes the statement of Theorem 2.

▶ **Theorem 18.** *Let ℓ, t, s, d, m, n be integers, with $m \geq d$, $\ell \leq n$. There exists a t-private, s-server $\mathbb{F}$-linear HSS $\pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ for any non-trivial $\mathcal{F} \subseteq POLY_{d,m}(\mathbb{F})^\ell$, with download rate $\mathsf{DownloadRate}(\pi) = \ell/n$, if and only if there exists a linear code $\mathcal{C} \subseteq \mathbb{F}^n$ with rate $\mathsf{DownloadRate}(\pi)$ and a labeling $\mathcal{L} : [n] \to [s]$ so that $\Delta_{\mathcal{L}}(\mathcal{C}) \geq dt + 1$.*

The work of [6] proved this equivalence for only the optimal-rate setting and left the equivalence in an arbitrary-rate setting as an open question. Theorem 18 settles this question and shows that linear HSS and linear codes of sufficient labelweight are indeed equivalent in *all* parameter regimes. The proof of Theorem 18 follows from Lemma 19 (for the forward direction) and Theorem 20 (for the converse) below.

We begin with the forward direction.

▶ **Lemma 19** (Follows from the analysis of [6]). *Let ℓ, t, s, d, m, n be integers, with $m \geq d$, $\ell \leq n$. Suppose there exists a t-private, s-server $\mathbb{F}$-linear HSS $\pi =$ (Share, Eval, Rec) for any non-trivial (see Definition 14) $\mathcal{F} \subseteq POLY_{d,m}(\mathbb{F})^{\ell}$, with download rate DownloadRate$(\pi) = \ell/n$. Then there exists a linear code $\mathcal{C} \subseteq \mathbb{F}^n$ with rate DownloadRate(π) and a labeling $\mathcal{L} : [n] \to [s]$ so that $\Delta_{\mathcal{L}}(\mathcal{C}) \geq dt + 1$.*

Though the statement of Lemma 19 is more general than its optimal-rate counterpart in [6], the proof is analogous; a careful reading of Lemma 12 in [6] shows that this forward direction does not leverage any of the strong symmetries of optimal rate linear HSS. Thus, we refer the reader to [6] for a proof.

Thus, in the rest of this section we focus on the converse, which does deviate from the analysis of [6], as we cannot leverage the same strong symmetries that they did, as discussed in Section 1.2. We first formally state the converse.

▶ **Theorem 20.** *Let ℓ, t, s, d, m, n be integers, with $m \geq d$. Suppose that there exists a linear code $\mathcal{C} \subseteq \mathbb{F}^n$ with dimension ℓ and rate ℓ/n. Suppose there exists a labeling $\mathcal{L} : [n] \to [s]$ so that $\Delta_{\mathcal{L}}(\mathcal{C}) \geq dt + 1$. Then there exists a t-private, s-server linear HSS $\pi =$ (Share, Eval, Rec) for $\mathrm{POLY}_{d,m}(\mathbb{F})^{\ell}$ with download rate ℓ/n and amortization parameter ℓ.*

The main ingredient in proving this direction without the strong symmetries of optimal rate linear HSS is the following lemma, which neatly generalizes the results of Lemma 13 and Corollaries 14, 15 of [6].

▶ **Lemma 21.** *Let $\mathcal{C}$ be a length n, dimension ℓ linear code over $\mathbb{F}_q$ with generator matrix $G \in \mathbb{F}_q^{\ell \times n}$. Let $\mathcal{L} : [n] \to [s]$ be a surjective labeling such that $\Delta_{\mathcal{L}}(\mathcal{C}) \geq dt + 1$.*

For $\Lambda \subseteq [s]$, let $G(\Lambda)$ denote the restriction of G to the columns $r \in [n]$ so that $\mathcal{L}(r) \in \Lambda$. Then for any $|\Lambda| \geq s - dt$, $G(\Lambda)$ has full row rank.

Proof. Let $\Lambda = \Lambda' \cup \Lambda''$ where $\Lambda' \cap \Lambda'' = \varnothing$ and $|\Lambda'| = s - dt$. If $G(\Lambda')$ achieves full row rank, then so does $G(\Lambda)$, since adding columns to a matrix does not induce linear independence among its rows. Hence, it suffices to consider only $|\Lambda| = s - dt$.

Up to a permutation of columns, G can be written as $G = [\, G(\Lambda) \mid G([s] \setminus \Lambda)]$. Let w denote the number of columns in $G(\Lambda)$.

Assume towards a contradiction that there exists some $\mathbf{v} \in \mathbb{F}_q^{\ell}$ such that $\mathbf{v}G(\Lambda) = 0^w$. Then

$$\mathbf{v}G = [\mathbf{v}G(\Lambda) \mid \mathbf{v}G([s] \setminus \Lambda)] = [0^w \mid \mathbf{v}G([s] \setminus \Lambda)]\,.$$

Since $|[s] \setminus \Lambda| = dt$, it follows that

$$\Delta_{\mathcal{L}}(\mathbf{v}G) = \Delta_{\mathcal{L}}(\mathbf{v}G([s] \setminus \Lambda)) \leq dt$$

which contradicts $\Delta_{\mathcal{L}}(\mathcal{C}) \geq dt + 1$. ◀

Let G_π be a reconstruction matrix. At a high level, Lemma 21 says that any sufficiently large submatrix of G_π, obtained by only considering columns labeled with a sufficiently large subset $\Lambda \subseteq [s]$, must be full-rank. [6] proved that such a property held for G_π in the optimal-rate regime that relied heavily on the fact that, in optimal-rate linear HSS, the output client downloads an equal number of output symbols from each server. This is equivalent to requiring that the sets $\mathcal{L}^{-1}(y) := \{x \in [n] : \mathcal{L}(x) = y\}$ be the same size for all $y \in [s]$. The proof of Lemma 21 shows that, perhaps surprisingly, sufficiently large submatrices of G_π still achieve full-rank even when the output client is allowed to download arbitrary numbers of output symbols from each server.

The remainder of the proof of Theorem 20 proceeds in a familiar syntax to that of [6]; we omit its presentation here and refer the interested reader to the full manuscript [7].

4 Linear HSS from Hermitian Codes

Linear HSS schemes presented by [23], [6] both achieve optimal download rate $1 - dt/s$ but require large amortization parameters to do so. [23] showed it was sufficient to take amortization parameter $\ell = (s - dt)\log(s) = O(s\log(s))$, and [6] proved that such an amortization parameter is in fact necessary in many parameter regimes, and is off by at most 1 otherwise.

It is a natural to ask whether linear HSS schemes that achieve a better trade-off between download rate and amortization parameter exist. *Specifically, can we make minor concessions to download rate and save substantially on the amortization needed?*

Through the lens of Theorem 20, this is equivalent to asking whether there exists a labelweight code with minimum labelweight $\geq dt + 1$ that achieves good rate at low dimension. A natural first attempt at an existential result would be via random coding; specifically, building a linear HSS scheme by starting with a random linear code and following the construction of Theorem 20. Unfortunately (and perhaps surprisingly!), we show in Appendix A that this results in strictly worse parameters than [23], [6].

In the following sections we take a different approach. We derive our labelweight codes straightforwardly from well-studied algebraic geometric codes: we set the number of servers s equal to the block length n of the codes and label each coordinate by the identity function $\mathcal{L} : [n] \to [s], x \mapsto x$. In this setting, labelweight is equivalent to Hamming weight. Note that the trivial labeling maximizes labelweight; the reverse direction of Theorem 18 showed that maximizing labelweight given a fixed download rate implies a linear HSS scheme where the greatest values of d, t can be considered.

This section constructs a family of linear HSS schemes from Hermitian codes; notably, such schemes achieve asymptotically optimal download rate while requiring an amortization parameter that is only linear in s.

▶ **Theorem 22.** *Let ℓ, t, s, d, m be positive integers and q a prime power satisfying $m \geq d$, $s - dt > 0$, and $s = q^3$. Then there exists an explicit t-private, s-server HSS $\pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ for any non-trivial $\mathcal{F} \subseteq \mathrm{POLY}_{d,m}(\mathbb{F}_{q^2})$ with*

$$\mathsf{DownloadRate}(\pi) = 1 - \frac{dt}{s} - \frac{s^{1/3} + 1}{2s^{2/3}}$$

and amortization parameter

$$\ell = s - dt - \frac{s^{2/3} - s^{1/3}}{2}.$$

We note that the above download rate is off of the optimal $1 - dt/s$ by only a $O(s^{-1/3})$ term; *it converges asymptotically to the optimal rate* $1 - dt/s$. Furthermore, it achieves this near-optimal download rate while requiring amortization only linear in s. We place these parameters in the context of [23], [6] in Figure 2.

Table 2 Comparison of Theorem 22 to [23], [6]. When $q = O(1)$ and $s = \omega(1)$, the download rate in Theorem 22 approaches the optimal rate; while the amortization is asmyptotically better.

	[23],[6]	Theorem 22
Download Rate	$1 - dt/s$	$1 - dt/s - O(s^{-1/3})$
Amortization	$(s - dt)\log_{q^2}(s)$	$s - dt - O(s^{2/3})$

We can compare these download rates (and the amortization parameters required to achieve them) for up to 1,000 servers $s \in (dt, 1000]$ in Table 3; we visualize this data in Figure 2.

The key takeaway from these numerical illustrations of Theorem 22 is that *even in non-asymptotic parameter regimes, small concessions in rate result in notable savings in amortization.*

Table 3 Comparison of download rates, amortization parameters from [23], [6] and Theorem 22 when $d = t = 2$.

# Servers	[23], [6]		Theorem 22		% Difference	
	DL Rate	Amort.	DL Rate	Amort.	DL Rate	Amort.
50	0.92	69	0.75	42	-18%	-39%
100	0.96	145	0.83	88	-13%	-39%
200	0.98	294	0.88	182	-10%	-38%
300	0.98	444	0.90	277	-8%	-38%
400	0.99	594	0.91	373	-7%	-37%
500	0.99	744	0.92	469	-7%	-37%
1000	0.99	1494	0.94	951	-5%	-36%

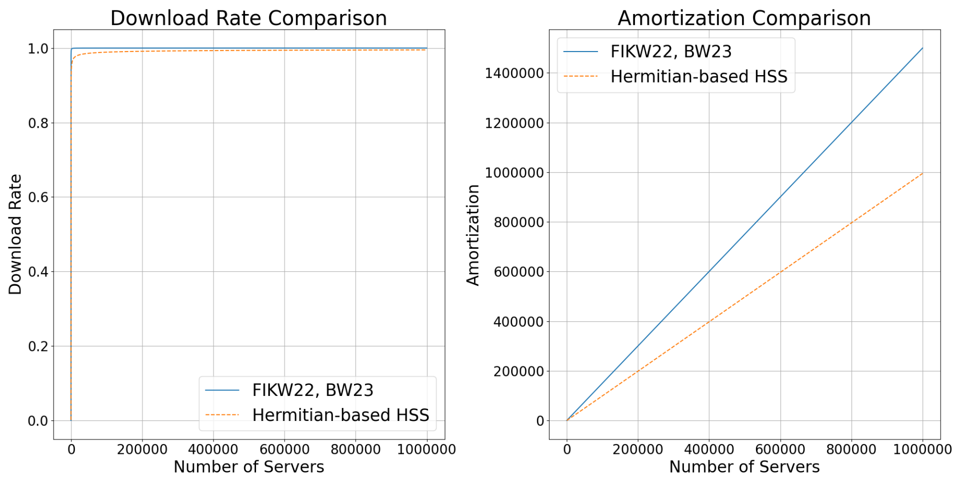


Figure 1 The left (right) plot compares the download rates (amortization parameters) of [23], [6] with that achieved by Theorem 22 when $d = t = 2$. The x-axis denotes the number of servers and ranges from 1 to 1,000,000 to illustrate the asymptotic convergence of Theorem 22 to the optimal rate of [23], [6] at a constant factor less amortization.

4.1 Hermitian Code Definition, Parameters

The construction proceeds by building an optimal labelweight code from Hermitian codes before applying the construction of Theorem 20 to derive the specification of a linear HSS scheme. We begin by recalling the definition and key properties of Hermitian codes. We defer a full treatment of this well-studied family of algebraic geometry codes to [30], [26].

▶ **Definition 23** (Hermitian Curve [30]). *The (affine) Hermitian Curve is given by the planar curve*

$$g(x, y) = y^q + y - x^{q+1}.$$

▶ **Definition 24** (Hermitian Code [26]). *Let $k \in \mathbb{Z}^+$ and let $M \subseteq \mathbb{F}_{q^2}[x, y]$ denote the set of all bi-variate polynomials $f(x, y)$ with total degree $\deg(f) < k$. Denote by*

$$\mathcal{Z} := \left((x, y) \in \mathbb{F}_{q^2}^2 \ : \ g(x, y) = 0\right)$$

the affine rational points of g, and fix any arbitrary ordering of its elements. Then the k-dimensional Hermitian code $\mathcal{H}$ is given by the set of codewords

$$\mathcal{H} := \{\mathrm{ev}_{\mathcal{Z}}(f) \ : \ f \in M\}$$

where $\mathrm{ev}_{\mathcal{Z}}(f) = (f(x, y) : (x, y) \in \mathcal{Z})$ denotes the standard evaluation map.

▶ **Theorem 25** (Hermitian Code Parameters [26]). *The k-dimensional Hermitian code $\mathcal{H}$ is a linear code of length $n = q^3$ and rate k/n with minimum distance*

$$q^3 - k - \frac{q(q-1)}{2} + 1. \tag{1}$$

4.2 Proof of Theorem 22

We first show the following lemma.

▶ **Lemma 26.** *Let $s = q^3, d, t \in \mathbb{Z}^+$ for some prime power q such that $s - dt > 0$. There exists a linear code $\mathcal{C} \subseteq \mathbb{F}_{q^2}^n$ and labeling function $L : [n] \to [s]$ satisfying $\Delta_L(\mathcal{C}) \geq dt + 1$ with rate*

$$R = 1 - \frac{dt}{s} - \frac{s^{1/3} + 1}{2s^{2/3}}$$

and dimension

$$k = s - dt - \frac{s^{2/3} - s^{1/3}}{2}. \tag{2}$$

Proof. Let $\mathcal{H}$ be the k-dimensional Hermitian code defined over alphabet $\mathbb{F}_{q^2}$. By Theorem 25, such a code has length $n = s = q^3$.

Allowing dimension k to be as specified in Equation 2, it follows from Equation 1 that $\Delta(\mathcal{H})$ is given by

$$s - \left(s - dt - \frac{s^{2/3} - s^{1/3}}{2}\right) - \frac{s^{1/3}(s^{1/3} - 1)}{2} + 1 = dt + 1.$$

The rate of $\mathcal{H}$ is given by

$$R_{\mathcal{H}} = \frac{1}{s}\left(s - dt - \frac{s^{2/3} - s^{1/3}}{2}\right) = 1 - \frac{dt}{s} - \frac{s^{1/3} + 1}{2s^{2/3}}.$$

Set $\mathcal{H} = \mathcal{C}$ and $L : [s] \to [s], x \mapsto x$; it immediately follows that $R_{\mathcal{H}} = R_{\mathcal{C}}$ and $\Delta(\mathcal{H}) = \Delta_L(\mathcal{C}) = dt + 1$, as desired. ◀

We are now prepared to prove Theorem 22 by applying Theorem 20.

Proof of Theorem 22. By Lemma 26, there exists a linear code $\mathcal{C} \subseteq \mathbb{F}_{q^2}^s$ and a labeling $L : [s] \to [s], x \mapsto x$ such that $\Delta_L(\mathcal{C}) \geq dt + 1$; furthermore $\mathcal{C}$ has dimension

$$\ell = s - dt - \frac{s^{2/3} - s^{1/3}}{2}$$

and rate

$$R = 1 - \frac{dt}{s} - \frac{s^{1/3}+1}{2s^{2/3}}.$$

By Theorem 20, the existence of such a labelweight code is equivalent to the existence of a linear HSS scheme with corresponding parameters; in particular, there exists a t-private, s-server, $\mathbb{F}_{q^2}$-linear HSS scheme $\pi = \pi(\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ that achieves download rate $\mathsf{DownloadRate}(\pi) = R$ and amortization parameter ℓ. ◀

5 Linear HSS from Goppa Codes

In this section we construct a family of linear HSS schemes from Goppa codes; unlike Theorem 22, these schemes do not achieve asymptotically optimal rate. However, this family of schemes stands apart from those of Theorem 22 by allowing us to compute over the binary field regardless of the number of servers employed. Furthermore, such schemes achieve a super-constant factor of amortization savings at practical server counts. We first state the result before considering its performance in realistic parameter regimes.

▶ **Theorem 27.** *Let ℓ, t, s, d, m, u be positive integers satisfying $m \geq d$, $s - dt > 0$, and $s = 2^u$, where*

$$u > \log_2\left(2(dt)^2 - 4dt + 2(dt+1)\sqrt{(dt)^2 - 2dt + 2} + 3\right).$$

Then there exists an explicit t-private, s-server HSS $\pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ for some non-trivial $\mathcal{F} \subseteq \mathrm{POLY}_{d,m}(\mathbb{F}_2)$ with

$$\mathit{DownloadRate}(\pi) = 1 - u\frac{dt}{s}$$

and amortization parameter $\ell = s - udt$.

Noting that $u \geq 3$ for all $d, t \in \mathbb{Z}^+$, we see that the download rate does not converge asymptotically to $1 - dt/s$ as the construction of Theorem 22 does; however, we show in Table 4 that for small parameter values, Theorem 27 vastly outperforms Theorem 22 in terms of preserving rate and saving on amortization. In particular, compared to Theorem 27, *the construction of Theorem 27 concedes less rate while delivering an order-of-magnitude savings in amortization* in practical parameter regimes. We illustrate these results graphically in Figure 2.

Table 4 Comparison of Download Rates and Amortization Values with Percentage Differences between FIKW and Goppa.

# Servers	[23],[6]		Theorem 27		% Reduction	
	DL Rate	Amortize	DL Rate	Amortize	DL Rate	Amortize
64	0.93	360	0.65	42	-31%	-88%
128	0.96	868	0.82	106	-15%	-88%
256	0.98	2016	0.91	234	-7%	-88%
512	0.99	4572	0.96	490	-3%	-89%
1024	0.99	10200	0.98	1002	-1.8%	-90%
2048	0.99	22484	0.99	2026	-0.9%	-91%

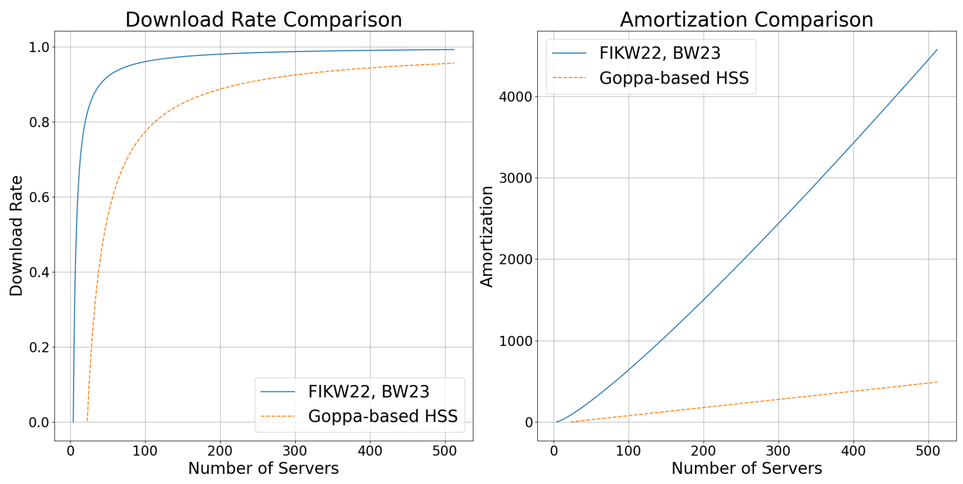


Figure 2 The left (right) plot compares the download rates (amortization parameters) of [23], [6] with that achieved by Theorem 27 when $d = t = 2$. The x-axis represents the number of servers and ranges from 1 to 512. This emphasizes the super-constant amortization savings of Theorem 27 at practical parameter regimes relative to [23], [6], with small concessions to rate.

5.1 Goppa Code Definition, Parameters

The proof of Theorem 27 is constructive; it proceeds by building an optimal labelweight code from Goppa codes before applying the construction of Theorem 20 to arrive at a linear HSS scheme with the desired properties. We begin by recalling the definition and key properties of binary Goppa codes, deferring a fuller treatment to [5], [24].

▶ **Definition 28** (Goppa Polynomial [5]). *For some $n \in \mathbb{Z}^+$, fix $V = \{\alpha_1, \dots, \alpha_n\} \subseteq \mathbb{F}_{2^u}$, $u \in \mathbb{Z}^+$. A Goppa polynomial is a polynomial*

$$g(x) = g_V(x) = g_0 + g_1 x + \cdots + g_r x^r \in \mathbb{F}_{2^u}[x]$$

satisfying $\deg(g) = r$ *and* $g(\alpha_i) \neq 0$ *for all* $\alpha_i \in V$.

Given the definition of the Goppa polynomial above, we can define a binary Goppa code.

▶ **Definition 29** (Goppa Codes [5]). *Let $n, u, r \in \mathbb{Z}^+$. Fix $V = \{\alpha_1, \dots, \alpha_n\} \subseteq \mathbb{F}_{2^u}$, $u \in \mathbb{Z}^+$ and let $g_V \in \mathbb{F}_{2^u}[x]$ be a Goppa polynomial of degree r. Then the Goppa code is the set of codewords given by*

$$\Gamma_{n,u,r} = \Gamma_{n,u,r}(g, V) := \left\{ \mathbf{c} = (c_1, \dots, c_n) \in \mathbb{F}_2^n \ : \ \sum_{i=1}^{n} \frac{c_i}{x - \alpha_i} \equiv 0 \mod g(x) \right\}$$

.

The parameters of Goppa codes are given by the following theorem.

▶ **Theorem 30** (Goppa Code Parameters [5]). *For $n, u, r \in \mathbb{Z}^+$ let $\Gamma = \Gamma_{n,u,r}$ be a binary Goppa code as in Definition 29. Then Γ is a linear code of length n, dimension $k \geq n - ur$, and minimum distance $d(\Gamma) \geq r + 1$.*

The parameters given by Theorem 30 only allow us to determine rate and minimum distance up to a lower bound, making it difficult to ascertain download rate and amortization when used to construct linear HSS schemes. Fortunately, these lower bounds are known to be sharp under additional assumptions. The following theorem gives one such instance.

▶ **Theorem 31** ([31]). *Fix $u, r \in \mathbb{Z}^+$ satisfying*

$$2r - 2 < \frac{2^u - 1}{2^{u/2}} \tag{3}$$

and let $g \in \mathbb{F}_{2^u}[x]$ be a Goppa polynomial of degree r with no repeated roots. Set $V = \mathbb{F}_{2^u}$ and let $\Gamma = \Gamma_{2^u,u,r}(g, V)$ be a Goppa code as in Definition 29. Then Γ is a binary linear code with dimension precisely $k = n - ur$.

We observe that, performing the appropriate manipulations, Equation 3 is satisfied for all

$$u \geq \max\left\{ \left\lceil \log_2\left(2r^2 - 4r + 2(r+1)\sqrt{r^2 - 2r + 2} + 3\right)\right\rceil , \right. \tag{4}$$
$$\left. \log_2\left(2r^2 - 4r + 2(r+1)\sqrt{r^2 - 2r + 2} + 3\right) + 1\right\}.$$

5.2 Proof of Theorem 27

In this section we prove Theorem 27. We first show the following lemma.

▶ **Lemma 32.** *Let $s, d, t, u \in \mathbb{Z}^+$ satisfy $s - dt > 0$ and $s = 2^u$, where*

$$u = \max\left\{ \left\lceil \log_2\left(2(dt)^2 - 4dt + 2(dt+1)\sqrt{(dt)^2 - 2dt + 2} + 3\right)\right\rceil , \right. \tag{5}$$
$$\left. \log_2\left(2(dt)^2 - 4dt + 2(dt+1)\sqrt{(dt)^2 - 2dt + 2} + 3\right) + 1\right\}.$$

There exists a linear code $\mathcal{C} \subseteq \mathbb{F}_2^n$ and labeling function $L : [n] \to [s]$ satisfying $\Delta_L(\mathcal{C}) \geq dt + 1$ with rate $R = 1 - udt/s$ and dimension $\ell = s - udt$.

Proof. Fix $V = \mathbb{F}_{2^u}$ and let $g \in \mathbb{F}_{2^u}[x]$ be an irreducible polynomial of degree $r = dt$; set $n = 2^u$. Let $\Gamma = \Gamma_{n,u,r}(g, V)$ be the binary Goppa code given by Definition 29. It follows from Equation 5 and the observation of Equation 4 that Γ has dimension $k = n - ur = s - udt$. It follows from Theorem 30 that Γ has minimum distance $d(\Gamma) \geq r + 1 = dt + 1$. Set $\mathcal{C} = \Gamma$ and define $L : [n] \to [s], x \mapsto x$ to be the identity labeling. It immediately follows that $\mathcal{C}$ has the desired rate and dimension. ◀

It is now straightforward to prove Theorem 27 by leveraging Theorem 20.

Proof of Theorem 27. By Lemma 32, there exists a linear code $\mathcal{C} \subseteq \mathbb{F}_2^s$ and a labeling $L : [s] \to [s]$ such that $\Delta_L(\mathcal{C}) \geq dt + 1$; furthermore $\mathcal{C}$ has dimension $\ell = s - udt$ and rate $R = 1 - udt/s$. By Theorem 20, the existence of such a labelweight code is equivalent to the existence of a linear HSS scheme with corresponding parameters. ◀

References

1 Donald Beaver and Joan Feigenbaum. Hiding instances in multioracle queries. In *STACS 90*, pages 37–48, 1990.

2 Donald Beaver, Joan Feigenbaum, Joe Kilian, and Phillip Rogaway. Security with low communication overhead. In *CRYPTO '90*, pages 62–76, 1990.

3 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, 1988.

4 Josh Cohen Benaloh. Secret sharing homomorphisms: Keeping shares of A secret sharing. In Andrew M. Odlyzko, editor, *CRYPTO '86*, pages 251–260, 1986.

5 Elwyn Berlekamp. Goppa codes. *IEEE Transactions on Information Theory*, 19(5):590–592, 1973.

6 Keller Blackwell and Mary Wootters. A characterization of optimal-rate linear homomorphic secret sharing schemes, and applications. In *15th Innovations in Theoretical Computer Science Conference (ITCS 2024)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.

7 Keller Blackwell and Mary Wootters. Improved trade-offs between amortization and download bandwidth for linear hss. *arXiv preprint arXiv:2403.08719*, 2024.

8 Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In *CRYPTO*, pages 489–518, 2019.

9 Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, and Michele Orrù. Homomorphic secret sharing: optimizations and applications. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2105–2122, 2017.

10 Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In *EUROCRYPT 2015, Part II*, pages 337–367, 2015.

11 Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 509–539. Springer, 2016. `doi:10.1007/978-3-662-53018-4_19`.

12 Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1292–1303. ACM, 2016. `doi:10.1145/2976749.2978429`.

13 Elette Boyle, Niv Gilboa, Yuval Ishai, Huijia Lin, and Stefano Tessaro. Foundations of homomorphic secret sharing. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 21:1–21:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. `doi:10.4230/LIPIcs.ITCS.2018.21`.

14 Elette Boyle, Lisa Kohl, and Peter Scholl. Homomorphic secret sharing from lattices without FHE. In *EUROCRYPT 2019, Part II*, pages 3–33, 2019.

15 David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, 1988.

16 Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. *J. ACM*, 1998.

17 Geoffroy Couteau and Pierre Meyer. Breaking the circuit size barrier for secure computation under quasi-polynomial LPN. In *EUROCRYPT 2021, Part II*, pages 842–870, 2021.

18 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, 2005. `doi:10.1007/978-3-540-30576-7_19`.

19 Ronald Cramer, Ivan Damgård, and Ueli M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *EUROCRYPT*, 2000.

20 Quang Dao, Yuval Ishai, Aayush Jain, and Huijia Lin. Multi-party homomorphic secret sharing and sublinear mpc from sparse lpn. In *Annual International Cryptology Conference*, pages 315–348. Springer, 2023.

21 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, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 93–122. Springer, 2016. `doi:10.1007/978-3-662-53015-3_4`.

22 Nelly Fazio, Rosario Gennaro, Tahereh Jafarikhah, and William E. Skeith III. Homomorphic secret sharing from Paillier encryption. In *Provable Security*, 2017.

23 Ingerid Fosli, Yuval Ishai, Victor I Kolobov, and Mary Wootters. On the download rate of homomorphic secret sharing. In *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

24 Valerii Denisovich Goppa. Codes associated with divisors. *Problemy Peredachi Informatsii*, 13(1):33–39, 1977.

25 Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. *Essential coding theory*. Draft from http://www.cse.buffalo.edu/atri/courses/coding-theory/book, 2019.

26 James William Peter Hirschfeld, Gábor Korchmáros, and Fernando Torres. *Algebraic curves over a finite field*, volume 20. Princeton University Press, 2008.

27 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.

28 Claudio Orlandi, Peter Scholl, and Sophia Yakoubov. The rise of paillier: Homomorphic secret sharing and public-key silent OT. In *EUROCRYPT 2021, Part I*, pages 678–708, 2021.

29 Lawrence Roy and Jaspal Singh. Large message homomorphic secret sharing from DCR and applications. In *CRYPTO 2021, Part III*, pages 687–717, 2021.

30 Henning Stichtenoth. *Algebraic Function Fields and Codes*. Springer Publishing Company, Incorporated, 2nd edition, 2008.

31 M. Van der Vlugt. The true dimension of certain binary goppa codes. *IEEE Transactions on Information Theory*, 36(2):397–398, 1990. doi:10.1109/18.52487.

A Linear HSS from Random Codes

In this section we show that the natural random coding approach does not appear to yield linear HSS schemes that meaningfully outperform the $\ell = O(s\log(s))$ amortization parameter required by [23], [6]. Indeed, the standard argument established that random linear codes correspond to linear HSS schemes that can attain good download rates, but – like [6, 23] – only with large amortization parameters.

A.1 Notation

To justify the notion that "random labelweight codes don't outperform Reed-Solomon codes in linear HSS amortization", we proceed by generalizing the well-known Gilbert-Varshamov Bound to the labelweight setting. One standard proof of this result (see, e.g., [25]) analyzes the distance of a random linear code, and we follow the same path here. We first introduce some notation.

▶ **Definition 33** (Labelweight Ball). *Let $L : [n] \to [s]$ be a surjective labeling. We define the labelweight ball $B_L(r)$ of radius $0 \le r \le n$ to be the set*

$$B_L(r) := \left\{c \in \mathbb{F}_q^n \;:\; \Delta_L(c) \le r\right\}$$

and define the volume of the labelweight ball to be $\mathrm{Vol}_L(r) = |B_L(r)|$.

For the purposes of our analysis, we consider only a fixed labeling function.

▶ **Assumption 1.** *Let $n, s, w \in \mathbb{Z}^+$ such that $n = sw$. In this section we will only consider the labeling*

$$L : [n] \to [s],\ x \mapsto \left\lceil \frac{x}{s} \right\rceil.$$

When paired with a code of length n, this balanced labeling simply labels the first w coordinates with 1, the second w coordinates with 2, and continues analogously until the last w coordinates are labeled with s. Intuitively, for fixed $n, s \in \mathbb{Z}^+$, such a balanced labeling pattern maximizes labelweight in expectation over random linear codes.

Under this fixed, balanced labeling function of Assumption 1, we have the following algebraic formulation of labelweight ball volume.

▶ **Observation 34.** *Let $n, s, w, r \in \mathbb{Z}^+$ such that $n = sw$ and $0 \leq r \leq n$. Then*

$$\mathrm{Vol}_L(r) = |B_L(r)| = \sum_{i=0}^{r} \binom{s}{i} (q^w - 1)^i .$$

Finally, we will need to define relative labelweight for labelweight codes, which is the natural analogue of relative minimum distance for linear codes.

▶ **Definition 35** (Relative Labelweight). *Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be a linear code and $L : [n] \to [s]$ a surjective labeling such that $\Delta_L(\mathcal{C}) = d$. We define the relative labelweight of $\mathcal{C}$ to be $\delta = d/s$.*

A.2 Generalization of q-ary Entropy

In the standard proof of the Gilbert-Varshamov bound, the volume of a Hamming ball is estimated by the q-ary entropy function. To generalize the proof to labelweight, we introduce the following generalization of the q-ary entropy function, which captures the volume of labelweight balls.

▶ **Definition 36** (Generalized q-ary Entropy). *Let $q \geq 2$, $w \geq 1$. For $x \in (0,1)$, we denote by $H_{q,w}(x)$ the generalized q-ary entropy function:*

$$H_{q,w}(x) = x \log_q(q^w - 1) - x \log_q(x) - (1-x)\log_q(1-x),$$

where $H_{q,w}(0), H_{q,w}(1)$ are defined as the limit of $H_{q,w}$ as $x \to 0, 1$, respectively.

Note that the case where $w = 1$ is the standard q-ary entropy function. We notice that, when properly normalized, the generalized entropy function can be approximated linearly.

▶ **Observation 37.** *For all $x \in [0, 1 - 1/q^w]$, $x \leq w^{-1} H_{q,w}(x) \leq x + \log_q(2^{1/w})$.*

Proof. Observe that

$$\begin{aligned} g(x) := w^{-1} H_{q,w}(x) - x &= w^{-1}\left(x \log_q(q^w - 1) - x \log_q(x) - (1-x)\log_q(1-x)\right) - x \\ &\leq w^{-1}\left(-x \log_q(x) - (1-x)\log_q(1-x)\right). \end{aligned}$$

Since $-x\log_q(x) - (1-x)\log_q(1-x)$ is a concave function which attains its maximal value when $x = 1/2$, it follows that $w^{-1}H_{q,w}(x) - x \leq w^{-1}\log_q(2)$ as desired. The lower bound follows from observing that g is itself a concave function, since

$$g''(x) = -\frac{1}{w \cdot x \cdot (1-x) \cdot \ln(q)} \leq 0 \; \forall x \in [0, 1 - 1/q^w]$$

and that its values at the endpoints of the domain $[0, 1 - 1/q^w]$ are non-negative; i.e., $g(0), g(1 - 1/q^w) \geq 0$. ◀

Equipped with this definition, our goal becomes to express the volume of a given labelweight ball in terms of the generalized entropy function. To do so, we note two helpful relations; we omit the proofs, which are elementary algebraic manipulations.

▶ **Observation 38.** *Let $s, p \in \mathbb{R}$ such that $s, p \geq 0$. Then*

$$q^{-sH_{q,w}(p)} = (1-p)^{(1-p)s} \left(\frac{p}{q^w - 1}\right)^{ps}$$

▶ **Observation 39.** *Let $w \in \mathbb{Z}^+$ and $p \in [0,1)$ satisfy $0 \leq p \leq 1 - 1/q^w$. Then*

$$\frac{p}{(1-p)(q^w-1)} \leq 1.$$

We now give the volume of a labelweight ball in terms of the generalized entropy function.

▶ **Lemma 40.** *Let $s, w \in \mathbb{Z}^+$ and $p \in [0,1)$ satisfy $0 \leq p \leq 1 - 1/q^w$ and $ps \in \mathbb{Z}^+$. Then*

$$\mathrm{Vol}_L(ps) \leq q^{sH_{q,w}(p)}.$$

Proof. Observe that

$$1 = (p + (1-p))^s = \sum_{i=0}^{s} \binom{s}{i} p^i (1-p)^{s-i} \geq \sum_{i=0}^{ps} \binom{s}{i} p^i (1-p)^{s-i}.$$

Multiplying through by $1 = (q^w-1)^i/(q^w-1)^i$ and applying Observation 39 yields

$$1 \geq \sum_{i=0}^{ps} \binom{s}{i} (q^w - 1)^i (1-p)^s \left(\frac{p}{(1-p)(q^w-1)}\right)^{ps}.$$

Finally, applying Observation 38 yields

$$\begin{aligned} 1 &\geq \sum_{i=0}^{ps} \binom{s}{i} (q^w-1)^i q^{-sH_{q,w}(p)} \\ &= \mathrm{Vol}_L(ps) q^{-sH_{q,w}(p)}. \end{aligned}$$ ◀

A.3 Gilbert-Varshamov Bound for Random Labelweight Codes

We are finally equipped to prove a generalization of the Gilbert-Varshamov bound for labelweight codes. This generalization will quantify the rate, and labelweight trade-off we can guarantee through random linear codes; viewed through the lens of Theorem 20, this tells us the download rate and amortization parameters that can be guaranteed by linear HSS scheme constructed from random linear codes.

▶ **Theorem 41.** *For $q \geq 2$, let $n, s, w \in \mathbb{Z}^+$ satisfy $n = sw$. Let $\delta \in [0, 1 - 1/q^w]$ satisfy $\delta s \in \mathbb{Z}^+$. For $\varepsilon \in [0, 1 - H_{q,w}(\delta)]$, let*

$$k = n - sH_{q,w}(\delta) - n\varepsilon \tag{6}$$

and let $G \in \mathbb{F}_q^{k \times n}$ be chosen uniformly at random.

Then with probability $> 1 - q^{-\varepsilon n}$, G is the generator matrix of a length n, dimension k, and relative labelweight $\geq \delta$ linear code with rate

$$R = 1 - \frac{sH_{q,w}(\delta)}{n} - \varepsilon.$$

Note that when $n = s$ and $w = 1$, Theorem 41 becomes the standard Gilbert-Varshamov Bound. Before we show the proof of Theorem 41, we interpret its statement in terms of linear HSS parameters.

▶ **Example 42.** Let $s, d, t \in \mathbb{Z}^+$ satisfying $s - dt > 0$ parameterize a linear HSS scheme as in Definition 7. Let s be as stated in Theorem 41 and set $\delta = (dt+1)/s$.

For the sake of illustration, suppose $w = \log_q(s)$ and $\varepsilon > 0$ a negligible constant. Let $\mathcal{C}$ denote the linear code with properties guaranteed by Theorem 41 and let π denote the t-private, s-server linear HSS constructed from $\mathcal{C}$ as in Theorem 20. Applying Observation 37 to Theorem 20, π has download rate at most

$$\mathsf{DownloadRate}(\pi) \leq 1 - \frac{dt+1}{s} - \varepsilon = 1 - \frac{dt}{s} - O(s^{-1})$$

with amortization parameter *at least*

$$\ell \geq (1-\varepsilon)s\log_q(s) - s\log_q(2) - (dt+1)\log_q(s) = \Omega(s\log(s))$$

for sufficiently small ε. In particular, we note that such a construction has an amortization parameter (at least) on the same $\Omega(s\log(s))$ order as that of [23], [6], while achieving a rate comparable to that of our Hermitian code-based construction of Theorem 22. We summarize this situation in Table 5.

Table 5 Comparison of Theorem 22 to Example 42.

	Thm. 22 (Hermitian code-based)	Ex. 42 (Random code-based)
Download Rate	$1 - dt/s - O(s^{-1/3})$	$\leq 1 - dt/s - O(s^{-1})$
Amortization	$s - dt - O(s^{2/3})$	$\Omega(s\log(s))$

We conclude this section by proving Theorem 41.

Proof of Theorem 41. Let $\mathcal{C} = \{\mathbf{m}G : \mathbf{m} \in \mathbb{F}_q^k\}$ be the linear code generated by G. It suffices to show that $\Delta_L(\mathbf{m}G) \geq d$ for all non-zero $\mathbf{m}$.

Accordingly, let $\mathbf{m} \in \mathbb{F}_q^k$ be a uniformly random non-zero vector; then $\mathbf{m}G$ is uniformly distributed over $\mathbb{F}_q^n$. It follows from Lemma 40 that

$$\Pr\left[\delta_L(\mathbf{m}G) < d\right] = \frac{\mathrm{Vol}_L(d-1)}{q^n} \leq \frac{q^{sH_{q,w}(\delta)}}{q^n} = q^{-k}\, q^{-n\varepsilon}.$$

Taking the Union Bound over all $\mathbf{m} \in \mathbb{F}_q^k$ yields the observation that with probability $1 - q^{-n\varepsilon}$, $\Delta_L(\mathcal{C}) \geq d$ as desired. ◀

Breaking RSA Generically Is Equivalent to Factoring, *with Preprocessing*

Dana Dachman-Soled
University of Maryland, College Park, MD, USA

Julian Loss
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Adam O'Neill
Manning College of Information & Computer Sciences, UMass Amherst, MA, USA

Abstract

We investigate the relationship between the classical RSA and factoring problems when *preprocessing* is considered. In such a model, adversaries can use an unbounded amount of precomputation to produce an "advice" string to then use during the online phase, when a problem instance becomes known. Previous work (*e.g.*, [Bernstein, Lange ASIACRYPT '13]) has shown that preprocessing attacks significantly improve the runtime of the best-known factoring algorithms. Due to these improvements, we ask whether the relationship between factoring and RSA fundamentally changes when preprocessing is allowed. Specifically, we investigate whether there is a superpolynomial gap between the runtime of the best attack on RSA with preprocessing and on factoring with preprocessing.

Our main result rules this out with respect to algorithms that perform *generic* computation on the RSA instance $x^e \bmod N$ yet arbitrary computation on the modulus N, namely a careful adaptation of the well-known generic ring model of Aggarwal and Maurer (Eurocrypt 2009) to the preprocessing setting. In particular, in this setting we show the existence of a factoring algorithm with polynomially related parameters, for any setting of RSA parameters.

Our main technical contribution is a set of new information-theoretic techniques that allow us to handle or eliminate cases in which the Aggarwal and Maurer result does not yield a factoring algorithm in the standard model with parameters that are polynomially related to those of the RSA algorithm. These techniques include two novel compression arguments, and a variant of the Fiat-Naor/Hellman tables construction that is tailored to the factoring setting.

2012 ACM Subject Classification Security and privacy → Public key (asymmetric) techniques; Security and privacy → Information-theoretic techniques

Keywords and phrases RSA, factoring, generic ring model, preprocessing

Digital Object Identifier 10.4230/LIPIcs.ITC.2024.8

Related Version *Full Version*: `https://eprint.iacr.org/2022/1261`

Funding *Dana Dachman-Soled*: Supported in part by NSF grants #CNS-1933033, #CNS-1453045 (CAREER), and by financial assistance awards 70NANB15H328 and 70NANB19H126 from the U.S. Department of Commerce, National Institute of Standards and Technology.
Julian Loss: Supported by the European Union, ERC-2023-STG, Project ID: 101116713. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Acknowledgements We thank Nikki Sigurdson for collaboration in the early stages of this work.

1 Introduction

1.1 Motivation and Main Results

Background. Use of the RSA function [26] $f_{N,e}(x) = x^e \bmod N$ where $N = pq$ is ubiquitous in practice, and attacks against it have been the subject of intensive study, see *e.g.* [4]. A key question about its security is its relationship to factoring N. While it is trivial to see that factoring N allows one to invert RSA, the converse is a major open problem. To make progress on this question, researchers have studied it in restricted (aka. idealized) models of computation. To our knowledge, this approach was initiated by Boneh and Venkatesan [5], who showed that a reduction from factoring to low-exponent RSA that is a *straight-line program* (SLP) gives rise to an efficient factoring algorithm. An SLP is simply an arithmetic program (performing only ring operations) that does not branch. A complementary approach, which we pursue in this work, is to consider RSA *adversaries* that are restricted. The best known result of this nature is due to Aggarwal and Maurer (which we abbreviate as AM) [1], who showed that breaking RSA and factoring are *equivalent* wrt. so-called "generic-ring algorithms" (GRAs), namely ones that treat the ring $\mathbb{Z}_N$ like a black-box, only performing ring operations and equality checks that allow branching. Put another way, GRAs work in any efficient ring isomorphic to $\mathbb{Z}_N$. Note that SLPs are a special case of GRAs.

In the context of any cryptographic problem or protocol it is valuable to consider *preprocessing attacks*, because an adversary may be willing to perform highly intensive computation to break many instances of the problem, if that computation only has to be performed once. To model this, one considers an *unbounded* algorithm that produces a short "advice" string that can be used to efficiently solve a problem instance once it becomes known (much more efficiently than without the advice string). Note that above-mentioned attacks on RSA from [4] do not take advantage of preprocessing. However, in the preprocessing setting, Bernstein and Lange [3] describe a Number Field Sieve (NFS) with preprocessing, based on work by Coppersmith [7], which significantly reduces the exponent in the running-time compared to the standard NFS factoring algorithm, and they use this to get an improved attack on RSA. Thus, a natural question is:

> Does the relationship between RSA and factoring fundamentally change in the preprocessing setting?

The Need for a New Model. To answer this question, we need to formalize a model of computation for this setting. The generic ring model (GRM) of AM considers an algorithm (called a generic ring algorithm or GRA) models generic computation on $\mathbb{Z}_N$ via a directed acyclic graph where nodes are labelled with constants (or the input indeterminate) in $\mathbb{Z}_N$ and operations $(+, \times, \div)$; execution corresponds to a walk in the graph according to suitable rules. Now, it is instructive to see why this model, extended to the preprocessing setting in the obvious way, is not suitable. In such an extension, at the end of the preprocessing stage, the adversary outputs a GRA to run in the online stage. But then observe that the best the adversary against RSA with preprocessing is the one that simply outputs, in the preprocessing stage, a GRA of size at most some T that obtains optimal advantage, where the advantage is computed with respect to the random choice of N with bitlength at most security parameter κ and random choice of $y = x^e \pmod N$. The description of this optimal GRA would then be passed to the online stage. This model, however, does not capture our intuition for the best-possible preprocessing attack against RSA in the GRM. For example, the following simple adversary is better but not captured: During preprocessing, it picks many instance-solution pairs $((N_1, e_1, x_1^{e_1} \bmod N_1), x_1), \ldots, ((N_i, e_i, x_i^{e_i} \bmod N_i), x_i)$ and inserts them into a hash table, then outputs this hash table as well as the aforementioned optimal GRA. Then, in the online stage, it performs a hash table lookup on the input RSA instance (N^*, e^*, y^*) to obtain a list of possible preimages $x_1^* \ldots, x_j^*$ (these all being stored in the same location in the hash table). It returns x_k^* such that $(x_k^*)^{e^*} \bmod N^* = y^*$ if it exists, and runs the optimal GRA otherwise. This attack is not captured because the hash table lookup uses the *bit-representation of the RSA instance*, while a GRA is agnostic of the particular representation of the ring. However, the attack is still "generic" because it works for *any* bit-representation of the ring (not just the canonical encoding of $\mathbb{Z}_N$). While this is a simple example, it captures techniques originating from Hellman tables [19], a common strategy for preprocessing algorithms in practice. In other words, such strategies crucially use the bit-representation of the problem instance.

Our New "GRM-with-Preprocessing" Model. To capture these types of representation-specific strategies, we will associate integers y of bitlength at most κ with *random labels*. We note that Damgard and Koprowski [12] and later Dodis et al. [15] previously considered random labels to model the *multiplicative group* $\mathbb{Z}_N^*$ (with N known to the adversary); it has not been done for the full ring $\mathbb{Z}_N$. (See below for a detailed comparison with these works.) That is, in our model, we consider an injective mapping π that encodes every element in $\{0,1\}^\kappa$ as a unique random string in $\{0,1\}^m$, where $m > \kappa$. We let the unbounded preprocessing algorithm read the entire description π and perform arbitrary computation. It produces a short advice string *state* that is passed to the online phase. The online algorithm is split into two parts, an intermediate algorithm, and a GRA. The intermediate algorithm is *non-generic* and gets the RSA instance $(N^*, e^*, \pi((x^*)^{e^*} \bmod N^*))$, where $N^* = pq$ has bit-length κ, but does *not* get access to π. This intermediate algorithm is what allows computation that *depends on the bit-representation of the RSA instance*, so that the next online stage can leverage the result in addition to the advice from the preprocessing stage. The intermediate algorithm outputs an *oracle-aided* GRA that computes relative to π, which we then run on the RSA instance. By "relative to π," we mean an addition step of the oracle-aided GRA takes as input two strings $y_1, y_2 \in \{0,1\}^m$ and outputs $\pi(\pi^{-1}(y_1) + \pi^{-1}(y_2) \pmod N)$. (Multiplication and division proceed analogously.). We call $S = |state|$ the space of the adversary and its running-time is specified by the pair (T_1, T_2), where T_1 is the runtime of the *intermediate algorithm*, and T_2 is the run-time of the GRA output by the intermediate algorithm. (Note that we require that $T_2 \leq T_1$.) We refer to this model as the "GRM-with-preprocessing" for simplicity. It is instructive to see that the simple adversary related to Hellman tables above is captured by it. This is because the intermediate algorithm can perform a hash table lookup on the input RSA instance $(N^*, e^*, y^* = \pi((x^*)^{e^*} \bmod N^*))$ to get a list $\pi(x_1^*), \ldots, \pi(x_j^*)$ of possible preimages, returning the GRA that on input y^* returns $\pi(x_k^*)$ such that $y^* = \pi(x_k^*)^{e^*} \bmod N^*$ if it exists (note finding this value uses access to π, *i.e.*, the ring operations) and runs the aforementioned optimal GRA otherwise. We view the GRM-with-preprocessing model as a main conceptual contribution of our work and contend that it faithfully captures our intuition for preprocessing attacks in the GRM.

We note that the study of upper and lower bounds on preprocessing in idealized models was pioneered by Corrigan-Gibbs and Kogan (CK) [10], who treated discrete-log-related problems in the case of groups. (See a more detailed comparison below.) Our modeling above follows their approach, but a key difference is that in their setting the group is fixed throughout the offline and online phase, whereas in our setting the group is fixed together with the RSA instance only in the online phase. Moreover, while the proofs of our results (described in Section 2) like those of CK use compression, the details differ substantially. For example, we do not use the random self-reducibility of RSA, whereas CK rely on self-reducibility of discrete log. This makes our results potentially applicable to broader settings.

A Result in the Random Injective Function Model. We present two main results below, which both emanate from a more basic result in *random injective function model* (RIM). In the RIM, the adversary has access to a random injective function with suitable parameters. We show that in the GRM-with-preprocessing model, any RSA algorithm with preprocessing implies the existence of a factoring algorithm with preprocessing in the RIM, with polynomially related parameters. This gets us a long way in answering our question for RSA algorithms in the GRM-with-preprocessing model and shows that the relationship of RSA and factoring does not fundamentally change in this setting, as long as we permit the factoring algorithm to operate in the RIM.

▶ **Theorem 1** (Informal). *Suppose there is an RSA adversary in the GRM-with-preprocessing model with space S_r and running-time $(T_{1,r}, T_{2,r})$ that succeeds with probability ϵ_r. Then there is a factoring adversary in the random injective function model (RIM) with space $S_f = S_r + O(1)$ and running-time $T_f = \mathsf{poly}(\kappa, T_{1,r}, T_{2,r}, 1/\epsilon_r)$ that succeeds with probability $\epsilon_f = \mathsf{poly}(\epsilon_\mathsf{r})$.*

See Theorem 7 for the formal statement. We will explain the bounds (which are identical) in the context of our random oracle model result below. Since our model allows an inefficient preprocessing phase, the RI function cannot easily be removed from our final factoring algorithm while maintaining the desired polynomially related space complexity and runtime from Theorem 1. The reason is that in the preprocessing phase of the factoring algorithm, the entire RI function could be queried and global information about it could be stored in the preprocessing advice. In this case, it is no longer possible for the online part of the factoring algorithm to simulate the RI "on the fly" since the responses generated by the simulator need to be consistent with the global information learned in the preprocessing phase. One approach to removing the RI would be to show that the global information about the random injective function (which has length S_f) can be simulated by fixing the input/output of some set of some q queries to the random injective function, and showing that any remaining queries not in this set can still be chosen "on the fly." This "bit-fixing" technique has been studied in a number of works, *e.g.* [9, 14]. However, this line of work proved a lower bound that q must be larger than $S_f T_f/(\epsilon_f)^2$ for simulation by the plain-model adversary to be ϵ_f-indistinguishable to an RIM adversary making T_f queries (note that we require $\approx \epsilon_f$-indistinguishability to guarantee that the factoring algorithm in the plain model still succeeds with probability $\mathsf{poly}(\epsilon_f) = \mathsf{poly}(\epsilon_r)$). For us, this would lead to trivial parameter settings. We show how to remove the RI function using an alternative argument below.

In particular, we extend the RIM result in two ways.

A result in the random oracle model. We first note that the RIM is much less natural to study factoring-with-preprocessing in than its counterpart the *random oracle model* (ROM) [2], hence we would like a result in the latter. The classical result of Luby and Rackoff shows that a 4-round Feistel network with random oracles in place of round functions is indistinguishable from a random permutation with forwards and backwards access. However, the distinguishing probability of an (unbounded) adversary is $\Omega(\frac{q^2}{2^{\kappa/2}})$, where $\kappa/2$ is the input/output length of the random oracle, and q is the number of queries made by the adversary, and this bound is known to be tight. In the preprocessing setting, the adversary can query the entire random oracle $q = 2^{\kappa/2}$, and so the distinguishing probability becomes vacuous. We present a technique to lift the Luby-Rackoff result to the case of unbounded preprocessing by using a slight modification of a 4-round Feistel network to implement a random injective function, instead of a random permutation. This 4-round Feistel will use round functions with input/output length $m/2$ to implement an injective function with domain size of $2^\kappa \ll 2^{m/2}$ and will thus circumvent the issue discussed above. We thus obtain the following result (see Theorem 11 for the formal statement), with the same concrete bounds as the RIM result.

▶ **Theorem 2** (Informal). *Suppose there is an RSA adverary in the GRM with preprocessing model with space S_r and running-time $(T_{1,r}, T_{2,r})$ that succeeds with probability ϵ_r. Then there is a factoring adversary in the random oracle model (ROM) with space $S_f = S_r + O(1)$ and running-time $T_f = \mathsf{poly}(\kappa, T_{1,r}, T_{2,r}, 1/\epsilon_r)$ that succeeds with probability $\epsilon_f = \mathsf{poly}(\epsilon_r)$.*

Note that the space complexity of our factoring algorithm is essentially the same as that of the RSA algorithm, namely $S + O(1)$. In terms of time complexity and success probability, our bounds are similar to those achieved by AM, which is to be expected. We differ from AM in that the success probability of our factoring algorithm ϵ_f depends only on ϵ_r, and not on $T_{1,r}, T_{2,r}$. We discuss additional differences between the time complexity and success probability of our ROM factoring algorithm and that of AM in Section 4. We believe using the ROM for the above result is reasonable since prior work on space/time tradeoffs (such as the seminal results of Hellman [19] and Fiat-Naor [17]) either required a random oracle or achieved simplified algorithms/improved parameters in the random oracle model. Nevertheless, it begs the question of whether the situation could change in the plain model.

A result in the plain model. Above we explained why it is difficult to remove the RI function while maintaining the desired parameters. Nevertheless, by developing new techniques for our setting we are finally able to show the following theorem statement, which is in the *plain model.*

▶ **Theorem 3** (Informal). *Suppose there is an RSA adverary in the GRM with preprocessing model with space S_r and running-time $(T_{1,r}, T_{2,r})$ that succeeds with probability ϵ_r. Then there is a factoring adversary in the* plain model *with space $S_f = O(S_r)$ and running-time $T_f = \mathsf{poly}(\kappa, T_{1,r}, T_{2,r}, 1/\epsilon_r)$ that succeeds with probability $\epsilon_f = \mathsf{poly}(\epsilon_r)$.*

The parameters are thus worse than what we obtain in the ROM, which is why we include the ROM result above. The main insight used for the plain-model result is to note that the online stage of the RI factoring algorithm we obtain has a particular form in which only a single uniformly random query is made to the forward direction of π, and a set of non-adaptive queries is made to the backward direction of π^{-1}. Combining a compression argument with a new argument based on a lemma of Drucker [16] (which to the best of our knowledge has not been previously used in a generic model setting) we are able to show that the online portion of the RIM factoring algorithm can be efficiently simulated in the plain model. Note, however, that we still include the ROM result above as it is obtained en route to our plain model result, illustrating many of our main techniques, and it enjoys a tighter reduction.

On Interpretation of our Results. To understand the significance of our results, it is helpful to recall the motivation for the GRM in the first place. AM explain that the GRM is an important computational model to consider for problems like RSA, where the adversary must output a function of its input that results in a ring element. As also mentioned below, there exist decisional problems (*e.g.*, Jacobi symbol computation) that are hard in the GRM but easy in $\mathbb{Z}_N$. Nevertheless, this does not seem to affect the question of the hardness of RSA. Indeed, an interesting question for future work would be to extend our results to hold relative to a Jacobi symbol oracle. Now, moving onto our GRM-with-preprocessing model, we stress that the intermediate stage, being non-generic, can run standard factoring algorithms on N such as NFS. Of course, if it simply factors N to break RSA, there is nothing to show. However, our results demonstrate that such non-generic computation cannot be fruitfully combined with *generic* computation on the RSA instance to obtain a significantly faster-than-factoring attack on RSA. In other words, essentially the entire computation by an RSA adversary needs to be non-generic for such a speed-up to be possible.

Finally, both our model and main theorem are very general in the sense that they show existence of a factoring algorithm with polynomially related parameters for *any setting of RSA parameters* $T_{1,r}, T_{2,r}, S_r, \epsilon_r$ and for a general class of algorithms. That is, our result does not restrict the relationship between $(T_{1,r}, T_{2,r}, S_r)$ (other than the requirement that $T_{1,r} \geq T_{2,r}$, which is implied by the model) and we show that generic RSA with preprocessing implies factoring with preprocessing, even for unconventional parameter settings (such as setting S_r to be larger than the time complexity of the best online factoring algorithm). We believe it is important to cover all parameter regimes, as this ensures that our result actually suggests a mathematical connection between the factoring and RSA problems themselves, rather than just showing that for the typical parameter settings used in practice the best factoring and RSA algorithms happen to have the same complexity.

On Using Bit-Fixing Instead of Compression. Another question is whether it is possible to rely on bit-fixing as alternative to our use of the compression technique (cf. [8]). That is, one would first show that an RSA algorithm of the form (A_0, A_1, A_2) with advice of size S_r, making at most T_r number of queries, and achieving success probability ϵ_r, implies the existence of an RSA algorithm of the form (A'_1, A'_2) making at most T'_r number of queries, and achieving success probability ϵ'_r in the bit fixing model, which fixes the labelling function π in q locations. It is possible that the AM reduction could then be applied more directly to (A'_1, A'_2) to obtain a factoring algorithm without going through a compression argument.

Unfortunately, similarly to the discussion above, this approach requires the number of fixed locations q to be at least $S_r T_r/(\epsilon_r)^2$. Since A'_1 cannot itself make oracle queries, for it to be able to choose A'_2 adaptively in the bit-fixing model, the information about the q fixed locations would need to be given to A'_1 as non-uniform advice. This would mean that the space of the RSA algorithm, and hence the resulting factoring algorithm, be at least $S_r T_r/(\epsilon_r)^2$, leading to trivial parameter settings.

On our use of information-theoretic techniques. We leverage information-theoretic techniques in three points in our proofs. First, in Section 4, we use a compression argument along with a theorem of De, Trevisan, Tulsiani [13] to show that the output of a successful RSA algorithm in the GRM with preprocessing model will satisfy a certain condition with high probability (see technical overview for more details). The condition being satisfied with high probability will then imply that the Aggarwal-Maurer factoring algorithm can be efficiently instantiated *in the ideal cipher or random oracle model*.

In Section 4.2, we use a different compression argument along with a theorem of Drucker [16] to show that our factoring algorithm (which invokes the Aggarwal-Maurer algorithm) in the ideal-cipher model can be efficiently simulated in the standard model by augmenting the preprocessing advice with a small table consisting of oracle query/response pairs. Note that this argument is not straightforward, since the online part of the factoring algorithm must respond to queries consistently with the pre-processing information, and it is not clear that this can be done efficiently (e.g. sampling the responses of the ideal cipher "on-the-fly" can lead to inconsistencies with the pre-processing information which may depend on global properties of the oracle).

Finally, in the full version [11], we use a variant of the Fiat-Naor/Hellman tables [19, 17] to obtain a factoring algorithm with the required space, time complexity for a range of parameters that is not covered by our factoring algorithm (which invokes the Aggarwal-Maurer algorithm) from above.

1.2 Related Work

There is an extensive body of literature on the hardness of the RSA problem and is relationship to factoring. Boneh and Venkatesan [5] gave the first among these results. Their result shows that reducing low-exponent RSA from factoring using a straight-line reduction is as hard as factoring itself. A similar result by Joux *et al.* [20] shows that when given access to an oracle computing eth roots modulo N of integers $x + c$ (where c is fixed and x varies), computing eth roots modulo N of arbitrary numbers becomes easier than factoring.

A more closely related line of work initiated by Brown [6] shows that for generic adversaries, computing RSA (or variants thereof), is as hard as factoring the modulus N. Brown's initial work considered only the case of SLPs without division and was subsequently extended by Leander and Rupp [22] to the case of GRAs without division.The work of Aggarwal and Maurer [1] finally showed that the problems are equivalent even for GRAs with division. A subsequent result of Jager and Schwenk showed that computing Jacobi symbols is equivalent to factoring for GRAs. Their result puts into question the soundness of the generic ring model (GRM), as it shows that there are problems which are hard in the GRM, but easy in the plain model. On the other hand, this result has no immediate implication for other computational problems like the RSA problems, which may still be meaningful to consider in the GRM. A recent work by Rotem and Segev also showed how the GRM can been used to analyze the security of verifiable delay functions [29].

The Generic Group Model (with Preprocessing). Starting with Nechaev [25], a long line of work has studied the complexity of group algorithm in the generic group model (GGM) [30, 24]. Algorithms in this model of Maurer [24] are restricted to accessing the group using handles and cannot compute on group elements directly. This makes it possible to prove information theoretic lower bounds on the running times and success probabilities of generic group algorithms for classic problems in cyclic groups (e.g., DLP, CDH, DDH). To the best of our knowledge, only two works have considered the RSA problem in idealized group models. The first of these work is due to Damgard and Koprowski [12] who ported Shoup's generic group model [30] to the setting of groups with unknown order and showed the generic hardness of computing eth roots in this model. The second work is that of Dodis et al. [15] who considered the instantiability of the hash function in FDH-RSA. On the one hand, unlike the GRA model that we use for online adversary, they only model the *multiplicative group* $\mathbb{Z}_N^*$ as generic. In other words, they do not allow the adversary to take advantage of the full ring structure of $\mathbb{Z}_N$. On the other, their model allows the online adversary to perform arbitrary side computations. Recall that we do not allow such computations in our model, as the online adversary is a GRA. We face many additional technical issues due to this point as well as preprocessing. More recently, the work of Corrigan-Gibbs and Kogan [10] initiated the study of preprocessing algorithms in the GGM. They considered generic upper and lower bounds for the discrete logarithm problem and associated problems. Their modelling approach is very similar to our own, in that the algorithm in the offline phase has access to the labelling oracle π and can pass an advice string of bounded size to the online phase of the algorithm. In addition to modeling differences mentioned above, they also consider adversaries who, in the online phase, may perform arbitrary side computations. Finally, the distinction between Maurer's and Shoup's GGM (*i.e.*, whether or not a labeling function is used) was studied in detail by Zhandry [32]; to our knowledge, the analogous issue has not been studied for GRAs.

The Algebraic Group Model. More recently, a series of works has explored the algebraic group model [18] as a means to abstract the properties of the groups $\mathbb{QR}_N$ and the multiplicative group $\mathbb{Z}_N^*$ more faithfully. The work of Katz et al. [21] introduced a quantitative version of the algebraic group model called the strong algebraic group model to relate the RSW assumption [27] over $\mathbb{QR}_N$ to the hardness of factoring (given that N is a product of safe primes p, q). Their model and ideas were extended to $\mathbb{Z}_N^*$ by Stevens and van Baarsen [31] who gave a general framework for computational reductions in the (strong) algebraic group model over $\mathbb{Z}_N^*$. Additionally, Rotem [28] reduces RSA to factoring in the algebraic group model over $\mathbb{Z}_N^*$.

2 Technical Overview

Our main result shows that any generic attack on RSA with preprocessing gives rise to a factoring algorithms with preprocessing in the random oracle model and plain models with polynomially related parameters. We begin by recapping the subclass of RSA algorithms we consider, and then discuss the high level approach of our proof of equivalence.

The RSA algorithm. Recall that we consider RSA adversaries that are split into two "fixed" parts (A_0^π, A_1) and a third part G^π that is adaptively chosen by A_1 upon seeing the RSA instance. In more detail, A_0^π gets oracle access to $\pi : \{0,1\}^\kappa \to \{0,1\}^m$ and is completely unbounded *both* in terms of computation and number of queries to π. A_0^π finally outputs a state *state* of size S_r (called A's *space*). A_1 takes as input *state* and the RSA instance $(N, e, \pi(y) = \pi(x^e))$, runs in time $T_{1,r}$, and outputs a GRA G^π of size (and hence running-time) $T_{2,r}$. The GRA G^π is an oracle-aided program that computes relative to π. In other words, each multiplication (resp. division, addition) step of G^π with inputs y_1, y_2 outputs $\pi(\pi^{-1}(y_1) \cdot \pi^{-1}(y_2) \pmod N)$ (resp. $\pi(\pi^{-1}(y_1) \cdot (\pi^{-1}(y_2))^{-1} \pmod N)$, $\pi(\pi^{-1}(y_1) + \pi^{-1}(y_2) \pmod N)$). A_1 is computationally bounded but may run for superpolynomial time. However, it may not make any queries to the oracle π. Finally, G^π takes as input $\pi(y)$ and evaluates $G^\pi(\pi(y))$. In the following, we fix π, a state *state* of some bounded size S_r output by A_0^π, as well as a modulus N and value e with $\gcd(e, \phi(N)) = 1$. We consider the success probability ϵ_r on input $\pi(y)$ of A_1 relative to these fixed values in outputting G^π such that $G^\pi(\pi(y)) = \pi(x)$ and $x^e = y \pmod N$. Here, the success probability is taken over random choice of $y \leftarrow \mathbb{Z}_N$ and coins of A_1. Fixing π, *state*, N, e simplifies our discussion and can easily be justified by an averaging argument. Our final analysis, however, considers these values drawn from an appropriate distribution. Our goal is to construct a factoring algorithm *with preprocessing* and with parameters S_f, T_f, ϵ_f (space, time, and success probability) that are polynomially related to $S_r, T_{1,r}, T_{2,r}, \epsilon_r$. Specifically, we require that $S_f = S_r + O(1)$, $T_f = \mathsf{poly}(\kappa, T_{1,r}, T_{2,r}, 1/\epsilon_r)$ and $\epsilon_f = \mathsf{poly}(\epsilon_r)$, where $\kappa = \log(N)$ is security parameter. We consider algorithms with unbounded preprocessing. Moreover, the algorithm A_1 does not

have access to π, but can perform arbitrary (and superpolynomially many) operations after learning the modulus N and the RSA instance $\pi(x^e)$. Only then does it hand over the remaining computation to the fully generic program G^{π}. In order for this to be possible, we must do several case analyses. To simplify this technical overview, we will henceforth conflate the online portion of algorithm's running times by setting $T_r = T_{r,1} + T_{r,2}$.
In the following, we first restrict our attention to the special case where A_1 outputs a straight-line program (SLP) with addition/multiplication *only* (i.e., without equality checks). This special case already requires most of the key ideas of our proof. We then briefly explain how to extend our result to the case where A_1 may output a generic ring algorithm (GRA).

First case analysis: Fiat-Naor argument. In the case that $T_r \cdot S_r \geq \epsilon_r \cdot 2^{\kappa}/4$, we will completely ignore the RSA algorithm, and construct a different Factoring algorithm in the RO model "from scratch." The idea is to use a theorem of Fiat and Naor [17], which extends Hellman's seminal result on space/time tradeoffs for inversion of a *random* function [19], to obtain space/time tradeoffs for inversion of *any* function f. Specifically, Fiat and Naor consider an arbitrary function $f : D \to D$ and show that f can be inverted with probability $1 - 1/|D|$ in the random oracle (RO) model with space S and time T, as long as $S^2 \cdot T \geq |D|^3 \cdot q(f)$, where $q(f)$ is the probability that two random elements in D collide under f.[1] We apply Fiat-Naor to the factoring problem by viewing f as the function that takes two $\kappa/2$ bit strings and multiplies them to obtain a κ-bit string, where $\kappa = \log(N)$. By carefully setting parameters and using properties of the second moment of the divisor function, to bound $q(f)$ as $q(f) \in O(\frac{\kappa^3}{2^{\kappa}})$, we obtain a factoring algorithm $S_f = S_r$, $T_f = \mathsf{poly}(\kappa) \cdot T_{2,r}^2$ and inversion probability $O(\epsilon_r)$. Note that all parameters are polynomial in the parameters of the RSA algorithm. See [11] for more details.

Factoring from RSA. We now consider the main parameter regime of interest, where $T_r \cdot S_r < \epsilon_r \cdot 2^{\kappa}/4$. In this parameter regime, we will show how to use the RSA algorithm to construct a factoring algorithm. However, before we can do that, we need to eliminate a crucial case in which the RSA algorithm is unhelpful for constructing a factoring algorithm. Let us first consider when and why the RSA algorithm is useful for factoring. Then we will show how to eliminate the remaining case.

Note that if A is successful with probability ϵ_r, then with probability ϵ_r the SLP S output by A_1 is such that on a randomly chosen $y = x^e$, $S^{\pi}(\pi(y)) = \pi(x)$. We begin by defining an "inversion procedure" on SLP's that, given S^{π} *with oracle access* to π and such that $S^{\pi}(\pi(y)) = \pi(x)$, outputs an SLP $\tilde{S}$ *with no oracle access* such that $\tilde{S}(y) = x$. (Crucially, the inversion procedure itself requires oracle access to π.) This, in turn, means that y is a root of the SLP $\tilde{S}(Y)^e - Y$, with respect to formal variable Y. In AM's analysis, they were able to conclude that if A is successful, then $\tilde{S}(Y)^e - Y$ must have many roots. Then, they showed an algorithm that successfully factors, given as input a non-zero SLP $\tilde{S}(Y)^e - Y$ with a sufficiently large fraction of roots. In our setting, however, we cannot necessarily conclude this. This is because we allow A_1 to output a different SLP $S^{\pi}_{\pi(y)}$ *after seeing* input $\pi(y)$ (we use the notation $S^{\pi}_{\pi(y)}$ to emphasize that the chosen SLP may depend on $\pi(y)$). This means that the SLP $S^{\pi}_{\pi(y)}$ output by A_1 can be tailored to succeed on $\pi(y)$ and on only *few* other inputs. Note that it is possible for A_1 to maintain an overall high success probability with this strategy. So while w.h.p. y itself must still be a root of the "inverted SLP" $\tilde{S}_{\pi(y)}(Y)^e - Y$, we are not guaranteed that $\tilde{S}_{\pi(y)}(Y)^e - Y$ has many roots overall. In this case, factoring fails.

[1] Their final algorithm actually requires only k-wise independent hash functions instead of a RO. For this overview, we assume a RO with $O(1)$ evaluation time.

The above reasoning leads to the second and third cases considered in our proof: The second case is that w.h.p. y is a root of $\tilde{S}_{\pi(y)}(Y)^e - Y$, *and* $\tilde{S}_{\pi(y)}(Y)^e - Y$ has *at most* J roots. The third case is that w.h.p. the SLP $\tilde{S}_{\pi(y)}(Y)^e - Y$, has *at least* J roots. The second case will lead to contradiction due to a compression argument. We will therefore be left with a (comparatively simple) third case which will imply existence of a factoring algorithm using the arguments of AM.

Second case analysis: Compression. For this case, we show how to construct an encoding routine that compresses the function table of a random injection π. Our main leverage to achieve this is the following idea. Suppose that y is a root of $\tilde{S}_{\pi(y)}(Y)^e - Y$, *and* $\tilde{S}_{\pi(y)}(Y)^e - Y$ has *at most* J roots. Then there is a space-efficient way for an encoding routine E^π (with oracle access to π) to transmit y to a decoder D (without oracle access to π) who knows only $\tilde{S}_{\pi(y)}$: Simply output the index of y among the J roots of $\tilde{S}_{\pi(y)}(Y)^e - Y$. (This takes $\log(J)$ bits.) Intuitively, we save space when $\log(J)$ is small compared to the trivial encoding of y, which specifies the index of y among all pre-images that are not yet mapped to an image in the encoding which is being constructed by E^π. Making this intuition rigorous, however, is quite challenging.

First, we must show how the encoder can efficiently transmit the description of $\tilde{S}_{\pi(y)}$ to the decoder. We may assume that A_1 and *state* will be known to the decoder (we can include *state* in the encoding). However, to obtain the correct SLP $\tilde{S}_{\pi(y)}$, the decoder must run A_1 on the correct random coins ρ and on the correct input $\pi(y)$. Furthermore, A_1 is only guaranteed to output an SLP $S^\pi_{\pi(y)}$ that is successful on $\pi(y)$ w.h.p., when $\pi(y) = \pi(x^e)$ and ρ are chosen *uniformly at random.* But we cannot afford to transmit the value of a random $\pi(y)$, nor the value of random coins ρ of A_1, while still achieving compression. To solve both of these problems, we rely, as prior work of Corrigan-Gibbs and Kogan [10] did, on a lemma of De, Trevisan, and Tulsiani [13]. This lemma proves incompressibility of an element x from a sufficiently large set $\mathcal{X}$ in a setting that allows the encoder and decoder to pre-share a random string of arbitrary length. For our purposes, this random string will allow us to both (1) select a random $\pi(y)$ from the set of images whose preimages are not yet known and (2) select the random tape ρ for A_1 to use together with input $\pi(y)$. Thus, the successful randomness can simply be encoded by its index within the shared random string, thus saving space. We mention that Corrigan-Gibbs and Kogan avoided encoding successful $\pi(y)$ values by using the random self-reducibility property of the discrete log problem to obtain an adversary that succeeds w.h.p. on *every* input. Unlike Corrigan-Gibbs and Kogan, our argument does *not* require random self-reducibility, and rather uses the random tape to select a random image $\pi(y)$ instead. Thus, while RSA also enjoys random self-reducibility, our proof does not make use of it, potentially making our techniques applicable to broader settings.

The third challenge is that in order to obtain $\tilde{S}_{\pi(y)}$ from $S_{\pi(y)}$, the decoder must run the SLP inversion procedure, which requires access to π. Therefore, our encoder E^π includes all the responses of queries to π during evaluation of the SLP inversion procedure in the encoding, replacing any query to $\pi^{-1}(\pi(y))$ itself with the formal variable Y. The final challenge is the delicate setting of parameters needed for the result to go through. We must set the value J (the number of roots in the SLP $\tilde{S}_{\pi(y)}(Y)^e - Y$) such that compression is achieved when the number of roots is at most J and, looking ahead, such that efficient factoring (with parameters S_f, T_f, ϵ_f that are polynomially related to S_r, T_r, ϵ_r) is possible when the number of roots is at least J. We note that our techniques for analyzing the encoding length are significantly different from those used by Corrigan-Gibbs and Kogan and may be of independent interest. (See the full version [11] for more details.)

Factoring and Extending to the GRA case. Once we have ensured that the the SLP $S_{\pi(y)}(Y)^e - Y$ has *at least* J roots w.h.p., we can directly apply a theorem of AM to obtain a factoring algorithm. Our final step will then be to extend the above discussion to a slightly broader setting in which A_1 outputs a GRA G^π rather than an SLP S^π. Here, we once again build on arguments of AM, although we need to put in some additional effort to make them work in our setting with preprocessing. In particular, the final factoring algorithm (with preprocessing) that we obtain is in the random injection (RI) model, where the algorithm requires access to *both* π and π^{-1}. This is because our factoring algorithm requires access to such a random injection in order to consistently simulate the oracle π to the RSA adversary over the preprocessing phase and the online phase in a space efficient manner. Thus, it remains to show how this oracle can be simulated in order to obtain a factoring algorithm in the plain model. For simplicity, we omit our intermediate result in the Random Oracle Model from this technical overview.

Obtaining our plain model result. In the following, we denote the random injective function by H and we denote by π the GRM oracle interface expected by the RSA adversary. We note that using backwards and forwards access to H, one can easily simulate queries made to π. We show that with some additional work one can dispense with the RI in our result and obtain a result in the plain model. To do so, we first observe that the online portion of our factoring algorithm in the RI model makes only a *single* query to H in the forward direction (on a *uniformly random* input modulo N), and makes a series of non-adaptive queries to H^{-1}. We will first show that we can simulate all the responses to the queries to H^{-1} while adding only a small overhead to the non-uniform advice. We will then show that the single query to the forward direction of H can be simulated as well.
Simulating queries to H^{-1}. Recall that A_1 receives the non-uniform advice *state* and the input $(N, e, \pi(x^e \mod N))$ and outputs a GRA. The factoring algorithm will run the GRA inversion algorithm by evaluating π^{-1} on hardcoded labels in the GRA that are *not equal* to the input label $\pi(x^e \mod N)$. Intuitively, since π is expanding, and since A_1 may not query the oracle, the only way A_1 can hardcode a valid label into the GRA is if this label is somehow stored in *state*. To formalize this intuition, for a fixed π, we consider the set S_π of valid images of π that are hardcoded into a GRA outputted by A_1 with sufficiently high probability over choice of input $(N, e, \pi(x^e \mod N))$ and the random coins of A_1.
We use a compression argument to show that for most choices of π, the set S_π is sufficiently small such that it can be added to A_1's advice *state*. By definition, for a fixed π, it is unlikely for A_1 to hardcode images of π into its outputted GRA if these images are not part of S_π. Thus, queries to π^{-1} can be simulated *without making a corresponding query to H* by using *state* as a lookup table.
Simulating the query to H. There is still a single query to the forward direction of H that must be taken care of. This is the query made by the factoring algorithm when generating the input to A_1. Specifically, it is a query with input $y = x^e \mod N$ and output $\pi(y) = \tilde{y}$. To simulate this query without accessing H, we construct a simulated plain model factoring algorithm as follows: In the preprocessing phase, the plain model algorithm internally samples a random injective function H, and the output of A_0 in the preprocessing stage is computed relative to this chosen H. Note that we can view A_0's input in the preprocessing stage as the entire oracle, and in particular, this will include the input/output pair $(y, \tilde{y}')$, where $y = x^e \mod N$ corresponds to the input value that will be given to A_1 in the online phase. In the online phase, our plain model factoring algorithm will actually resample the output value of H on input y and replace it with a uniform random string $\tilde{y}$. This resampled value

$\tilde{y}$ will then be given to A_1 in the online phase as the supposed value of $\pi(y)$. A lemma of Drucker [16] implies that (on average) the output distribution of a compressing algorithm A_0, which outputs *state*, does not change much when a single input in a randomly chosen location (location y) is switched from a fixed value to a randomly resampled value. This implies that the RI factoring algorithm will behave roughly the same when π is simulated in this manner. See the full version [11] for further details.

3 Preliminaries

3.1 Notation and Conventions

We denote the sampling of a uniformly random element x from a set S as $x \leftarrow S$. Similarly, we denote the output y of a randomized algorithm A on input x as $y \leftarrow \mathsf{A}(x)$. We sometimes also write $y := \mathsf{A}(x;\omega)$ to denote that A deterministically computes y on input x and random coins ω. To denote that an algorithm A has access to an oracle O during runtime, we write A^{O}. We denote as $\mathbb{Z}_N$ the ring of integers modulo N, and as $[N]$ the set $\{1,...,N\}$. We write $\nu_N(f)$ to denote the fraction of roots of a polynomial f over $\mathbb{Z}_N$, i.e.,

$$\nu_N(f) := \frac{|\{a \in \mathbb{Z}_N \mid f(a) = 0\}|}{N}.$$

Throughout, we denote the security parameter as κ. For $k, m \in \mathbb{N}$ we denote by $\mathsf{Func}[k,m]$ the set of *functions* $F\colon \{0,1\}^k \to \{0,1\}^m$. Denote by $\mathsf{Perm}[m]$ the set of *permutations* on $\{0,1\}^m$. We denote by $\mathsf{FuncInj}[k,m]$ the set of *injective functions* $I\colon \{0,1\}^k \to \{0,1\}^m$.

3.2 Incompressibility Lemmas

We use the following lemma by De et al. [13].

▶ **Lemma 4** (De, Trevisan, Tulsiani [13]). *Let* $\mathsf{E} : \mathcal{X} \times \{0,1\}^\rho \to \{0,1\}^m$ *and* $\mathsf{D} : \{0,1\}^m \to \mathcal{X} \times \{0,1\}^\rho$ *be randomized encoding and decoding procedures such that, for every* $x \in \mathcal{X}, \Pr_{r \leftarrow \{0,1\}^\rho}[\mathsf{D}(\mathsf{E}(x,r),r) = x] \geq \gamma$. *Then,* $m \geq \log|\mathcal{X}| - \log 1/\gamma$.

▶ Remark 5. As noted by [10], this lemma also holds when the encoding and decoding algorithms have access to a common random oracle.

The following lemma is from Drucker [16].

▶ **Lemma 6** (Drucker [16]). *Let* $N, S, m \geq 1$ *be integers. Given a possibly-randomized mapping* $A_0(\tilde{y}_0, \ldots, \tilde{y}_{N-1}) : \{0,1\}^{N\times m} \to \{0,1\}^S$, *and a collection* $\mathcal{D}_0, \ldots, \mathcal{D}_{N-1}$ *of mutually independent distributions over* $\{0,1\}^m$, *for* $y \in \mathbb{Z}_N$, *let*

$$\gamma_y := \mathbb{E}_{\tilde{y}\sim\mathcal{D}_y}[||A_0(\mathcal{D}_0, \ldots, \mathcal{D}_{y-1}, \tilde{y}, \mathcal{D}_{y+1}, \ldots, \mathcal{D}_{N-1}) - A_0(\mathcal{D}_0, \ldots, \mathcal{D}_{N-1})||_{\mathsf{stat}}],$$

where the notation $||\cdot - \cdot||_{\mathsf{stat}}$ *denotes the statistical distance between two distributions.*

We have that

$$\frac{1}{N}\sum_{y\in\mathbb{Z}_N} \gamma_y \leq \sqrt{\frac{\ln 2}{2} \cdot \frac{S+1}{N}}.$$

4 Main Results

We begin by stating two theorems that will be used to obtain both our plain model and RO model results.

▶ **Theorem 7.** *Let $S_r := S_r(\kappa)$, $T_{1,r} := T_{1,r}(\kappa), T_{2,r} := T_{2,r}(\kappa), \epsilon = \epsilon(\kappa)$ such that for sufficiently large κ, $S_r \cdot T_{2,r} \leq \epsilon/162^\kappa$. Let $\mathsf{A} = (\mathsf{A}_0^\pi, \mathsf{A}_1)$ be an $(S_r, T_{1,r}, T_{2,r})$-GP-RSA algorithm relative to* RSAGen*, and let $\mathsf{Adv}^{\text{ag-rsa}}_{\mathsf{RSAGen}}(\mathsf{A}) = \epsilon$.*

Then there exists a (S_f, T_f)-factoring algorithm B *in the random injective function model relative to* RSAGen *such that*

$$\mathbf{Adv}^{\mathbf{fac}}_{\mathsf{RSAGen}}(\mathsf{B}) \in \Omega(\epsilon^3),$$

such that $T_f := \mathsf{poly}(\kappa) \cdot (T_{1,r} + T_{2,r}^5 + \frac{T_{2,r}^{7/2}}{\epsilon^{3/2}})$, and such that $S_f := S_r$.

This theorem is proved in the full version [11].

▶ Remark 8. We give a comparison here of the bounds we achieve in Theorem 7 versus those achieved by AM's factoring algorithm. First, we consider our runtime of $T_f := \mathsf{poly}(\kappa) \cdot (T_{1,r} + T_{2,r}^5 + \frac{T_{2,r}^{7/2}}{\epsilon^{3/2}})$, and focus on the $(T_{1,r} + T_{2,r}^5 + \frac{T_{2,r}^{7/2}}{\epsilon^{3/2}})$ part. The first term's dependence on $T_{1,r}$ is unavoidable, since the factoring algorithm must run the RSA algorithm at least once. The second term of $T_{2,r}^5$ comes from running the SLPFAC^π algorithm with $M' := \mathsf{poly}(\kappa) \cdot (T_{2,r})^2$. This corresponds exactly to running AM's Algorithm 1 M' number of times, whereas they only run it once. The reason for one of the $T_{2,r}$ factors in M' is that the success probability of AM's Algorithm 1 depended linearly on $1/T_{2,r}$ (the size of the SLP) and we wanted to remove the dependence on $T_{2,r}$ from our factoring algorithm's success probability. The reason for the second $T_{2,r}$ factor is that the success probability of AM's Algorithm 1 also depends linearly on the fraction of roots in the SLP. For them, this is essentially equivalent to the RSA algorithm's success probability. But for us, due to our compression argument, the fraction of roots in the SLP is only guaranteed to be at least $J/N \approx \epsilon/T_{2,r}$. Since we want to remove the dependence on $T_{2,r}$ from the success probability of the factoring algorithm, this accounts for the second factor of $T_{2,r}$ in our runtime. Moving to the third term of $\frac{T_{2,r}^{7/2}}{\epsilon^{3/2}}$, this comes from the runtime of Alg2AM which is essentially the same as Algorithm 2 of AM. We are able to reduce from $\epsilon^{3/2}$ to $\epsilon^{5/2}$ in the denominator, since we assume that $\epsilon > 1/N$ and since we ignore $\mathsf{polylog}(N) = \mathsf{poly}(\kappa)$ factors in our analysis.

Next we move on to our success probability. We have ϵ^3 compared to linear dependence on ϵ in AM because we only provide a factoring algorithm when a certain event occurs. The event that we consider is only guaranteed to occur with probability ϵ with respect to ϵ-fraction of oracles.

▶ Remark 9. Note that achieving the desired factoring algorithm when $T_{r,2} \geq 2^{\kappa/10}$ or $\epsilon' \leq 1/2^{\kappa/6}$ is trivial since there is a trivial factoring algorithm that runs in time $T_f = O\left((2^{\kappa/10})^5\right) = O\left(2^{\kappa/2}\right)$, with zero pre-processing and success probability 1, as well as a trivial factoring algorithm that achieves success probability $\Omega\left((2^{-\kappa/6})^3\right) = \Omega\left(2^{-\kappa/2}\right)$ with zero pre-processing and $\mathsf{poly}(\kappa)$ time (which just guesses a random number in $[2^{\kappa/2}]$ as one of the factors of N). We therefore assume WLOG that $T_{r,2} < 2^{\kappa/10}$ and $\epsilon' > 2^{-\kappa/6}$.

The following theorem instantiates the algorithm of Fiat-Naor in the setting of factoring-with-preprocessing.

▶ **Theorem 10.** *Let $\tilde{S} = \tilde{S}(\kappa)$, $\tilde{T} = \tilde{T}(\kappa), \tilde{\epsilon} = \tilde{\epsilon}(\kappa)$ such that for sufficiently large κ, $\tilde{S} \cdot \tilde{T} \geq \tilde{\epsilon} 2^\kappa$. Then there exists a plain-model (S_f, T_f)-factoring-with-preprocessing algorithm A such that for $\kappa \in \mathbb{N}$, we have*

$$\mathbf{Adv}^{\mathbf{fac}}_{\mathsf{RSAGen}}(A) \in \Omega(\tilde{\epsilon}),$$

we further have that $S_f = \tilde{S}$, and $T_f = \mathsf{poly}(\kappa) \cdot \tilde{T}^2$.

This theorem is proved in the full version [11].

In Section 4.1 and 4.2 we formally state our results for the RO and plain model and explain how Theorems 7 and 10 are used to obtain those results.

4.1 The RO model result

▶ **Theorem 11.** *Let* $\mathsf{A} = (\mathsf{A}_0^\pi, \mathsf{A}_1)$ *be an* $(S_r, T_{1,r}, T_{2,r})$*-GP-RSA algorithm relative to* RSAGen*, and let* $\epsilon := \mathsf{Adv}^{\text{ag-rsa}}_{\mathsf{RSAGen}}(\mathsf{A})$*.*

Then there exists a (S_f, T_f)*-factoring algorithm* B *in the random injective function model relative to* RSAGen *such that*

$$\mathbf{Adv}^{\mathbf{fac}}_{\mathsf{RSAGen}}(\mathsf{B}) \in \Omega(\epsilon^3),$$

such that $T_f := \mathsf{poly}(\kappa) \cdot (T_{1,r} + T_{2,r}^5 + \frac{T_{2,r}^{7/2}}{\epsilon^{3/2}})$*, and such that* $S_f := S_r + O(1)$*.*

To prove Theorem 11 we first show that the RI-model factoring algorithm from Theorem 7 (which gets backwards and forwards access to the random injective function), can be converted into a factoring algorithm in the Random Oracle model with the same parameters.

Specifically, in Proposition 21 we take A to be our final factoring algorithm FAC^π (see [11]) and $q = 2^\kappa$. Now set L such that

$$2^{2\kappa}/L \in O(N^2/L) \le 1/(2N) \ .$$

As $\epsilon_f \in \Omega(1/N)$ where ϵ_f is the advantage FAC^π relative to a random injection on $[L]$, we have

$$\epsilon'_f \ge \epsilon_f/2$$

where ϵ'_f is the advantage of the factoring algorithm in RO model that runs FAC^π, answering its queries via Luby-Rackoff. This RO-model factoring algorithm is for the case that for sufficiently large κ, $S_r \cdot T_{2,r} \le \epsilon/162^\kappa$.

Setting $\tilde{S} = S_r$, $\tilde{T} = T_{2,r}, \tilde{\epsilon} = \epsilon/16$ and applying Theorem 10, we obtain a plain model factoring algorithm with parameters $S_f = S_r$, $T_f = \mathsf{poly}(\kappa) \cdot T_{2,r}^2$, and advantage $\epsilon_f \in \Omega(\epsilon)$. This plain-model factoring algorithm is for the case that for sufficiently large κ, $S_r \cdot T_{2,r} \ge \epsilon/162^\kappa$.

Note that it is also possible that neither of the above cases is satisfied and that, rather, for *infinitely many* κ, $S_r(\kappa) \cdot T_{2,r}(\kappa) \ge \epsilon(\kappa) \cdot 2^\kappa/16$, and *simultaneously*, for *infinitely many* κ, $S_r(\kappa) \cdot T_{2,r}(\kappa) < \epsilon(\kappa) \cdot 2^\kappa/16$. If this occurs, we obtain our factoring algorithm by having the unbounded pre-processing stage of the factoring algorithm do the following: On fixed input κ, it will run the GP-RSA algorithm exhaustively on all possible random coins and inputs to determine the exact constants $S_r(\kappa), T_{r,2}(\kappa), \epsilon(\kappa)$. It will then check whether $S_r(\kappa) \cdot T_{r,2}(\kappa) \ge \epsilon(\kappa) \cdot 2^\kappa/16$ or $S_r(\kappa) \cdot T_{r,2}(\kappa) < \epsilon(\kappa) \cdot 2^\kappa/16$. If the former is true, it will append a "0" bit to the preprocessing advice *state* to tell the online portion of the factoring algorithm to run the factoring algorithm for the first case. If the latter is true, it will append a "1" bit to the preprocessing advice to tell the online portion of the factoring algorithm to run the factoring algorithm for the second case. Thus, the preprocessing advice increases by a single bit (so it still satisfies $S_f = S_r + O(1)$) and the other parameters $T_f, \mathbf{Adv}^{\mathbf{fac}}_{\mathsf{RSAGen}}(\mathsf{B})$ remain the same and therefore satisfy the required constraints of Theorem 11.

4.2 The plain model result

▶ **Theorem 12.** *Let* $\mathsf{A} = (\mathsf{A}_0^\pi, \mathsf{A}_1)$ *be an* $(S_r, T_{1,r}, T_{2,r})$*-GP-RSA algorithm relative to* RSAGen*, and let* $\epsilon := \mathsf{Adv}^{\text{ag-rsa}}_{\mathsf{RSAGen}}(\mathsf{A})$*.*

Then there exists a (S_f, T_f)*-factoring algorithm* B *in the plain model relative to* RSAGen *such that*

$$\mathbf{Adv}^{\mathbf{fac}}_{\mathsf{RSAGen}}(\mathsf{B}) \in \Omega(\epsilon^6),$$

such that $T_f := \mathsf{poly}(\kappa) \cdot (T_{1,r} + T_{2,r}^5 + \frac{T_{2,r}^{7/2}}{\epsilon^{3/2}})$*, and such that* $S_f := O(S_r)$*.*

To prove Theorem 12 we start from the RI model factoring algorithm obtained in Theorem 7 and prove the following theorem:

▶ **Theorem 13.** *Let* $S_r := S_r(\kappa)$*,* $T_{1,r} := T_{1,r}(\kappa), T_{2,r} := T_{2,r}(\kappa), \epsilon = \epsilon(\kappa)$*. Let* $\mathsf{A} = (\mathsf{A}_0^\pi, \mathsf{A}_1)$ *be an* $(S_r, T_{1,r}, T_{2,r})$*-GP-RSA algorithm relative to* RSAGen*, and let* $\epsilon := \mathsf{Adv}^{\text{ag-rsa}}_{\mathsf{RSAGen}}(\mathsf{A})$*.*

There exists a constant c *such that, if for sufficiently large* κ*,* $S_r \cdot T_{2,r} \leq c \cdot \epsilon^6 2^\kappa$*, then there exists a* (S_f, T_f)*-factoring algorithm* B *in the plain model relative to* RSAGen *such that*

$$\mathbf{Adv}^{\mathbf{fac}}_{\mathsf{RSAGen}}(\mathsf{B}) \in \Omega(\epsilon^6),$$

such that $T_f := \mathsf{poly}(\kappa) \cdot (T_{1,r} + T_{2,r}^5 + \frac{T_{2,r}^{7/2}}{\epsilon^{3/2}})$*, and such that* $S_f := S_r$*.*

The proof of Theorem 13 appears in the full version [11]. To obtain an algorithm for the case that for sufficiently large κ, $S_r \cdot T_{2,r} \geq c \cdot \epsilon^6 2^\kappa$, we set $\tilde{S} = S_r$, $\tilde{T} = T_{2,r}, \tilde{\epsilon} = c \cdot \epsilon^6$ and apply Theorem 10. This yields a plain model factoring algorithm with parameters $S_f = S_r$, $T_f = \mathsf{poly}(\kappa) \cdot T_{2,r}^2$, and advantage $\epsilon_f \in \Omega(\epsilon^6)$. Using the same argument as in Section 4.1, we obtain a single factoring algorithm that covers all cases in the plain model with the parameters of Theorem 12.

Two Events. Fix A, N, e, π and *state* as in Lemma 5 in [11]. We consider the probability of two events over the randomness of ComGRA and choice of $y \leftarrow \mathbb{Z}_N$. Set $J := \frac{(1-\epsilon'/2)\epsilon' \cdot N}{8 \log NT_{2,r}} = \frac{(1-\epsilon'/2) \cdot N}{4R_1 T_{2,r}} = N \cdot \delta$, where $\delta := J/N$.

- Event $E[N, e, \mathit{state}, \pi]_1$: $\mathsf{ComGRA}[\mathsf{A}]^\pi$ on input $\pi(y)$ returns a list of polynomials $\{R_1, \ldots, R_{\psi+1}\}$ s.t. y is *negatively oriented* with respect to one of $\{R_1, \ldots, R_{\psi+1}\}$.
- Event $E[N, e, \mathit{state}, \pi]_2$: $\mathsf{ComGRA}[\mathsf{A}]^\pi$ on input $\pi(y)$ returns a list of polynomials $\{R_1, \ldots, R_{\psi+1}\}$ s.t. $\nu_N(R_{\psi+1}) \in (\delta, 1-\delta)$.

▶ **Corollary 14** (of Lemma 5 in [11]). *Suppose that the conditions of Lemma 5 in [11] hold. Then at least one of the events* $E[N, e, \mathit{state}, \pi]_1$ *or* $E[N, e, \mathit{state}, \pi]_2$ *occurs with probability at least* $\epsilon/4$*.*

Looking ahead, if $E[N, e, \mathit{state}, \pi]_1$ occurs, then A will be useless for factoring. Our task, therefore, is to prove that $E[N, e, \mathit{state}, \pi]_1$ occurs with probability less than $\epsilon' = \epsilon/4$ (which we do in the full version [11] via a compression argument). We therefore conclude that $E[N, e, \mathit{state}, \pi]_2$ occurs with probability at least $\epsilon' = \epsilon/4$.

References

1 Divesh Aggarwal and Ueli Maurer. Breaking RSA generically is equivalent to factoring. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 36–53. Springer, Heidelberg, April 2009. doi:10.1007/978-3-642-01001-9_2.

2 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.

3 Daniel J. Bernstein and Tanja Lange. Non-uniform cracks in the concrete: The power of free precomputation. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2013. doi:10.1007/978-3-642-42045-0_17.

4 Dan Boneh. Twenty years of attacks on the rsa cryptosystem. *Notices of the American Mathematical Society (AMS)*, 46(2):203–213, 1999.

5 Dan Boneh and Ramarathnam Venkatesan. Breaking RSA may not be equivalent to factoring. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 59–71. Springer, Heidelberg, May / June 1998. doi:10.1007/BFb0054117.

6 D. R. L. Brown. Breaking rsa may be as difficult as factoring. *Eprint Cryptology Archive*, 2006.

7 Don Coppersmith. Modifications to the number field sieve. *J. Cryptol.*, 6(3):169–180, 1993.

8 Sandro Coretti, Yevgeniy Dodis, and Siyao Guo. Non-uniform bounds in the random-permutation, ideal-cipher, and generic-group models. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 693–721. Springer, Heidelberg, August 2018. doi:10.1007/978-3-319-96884-1_23.

9 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, Heidelberg, April / May 2018. doi:10.1007/978-3-319-78381-9_9.

10 Henry Corrigan-Gibbs and Dmitry Kogan. The discrete-logarithm problem with preprocessing. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 415–447. Springer, Heidelberg, April / May 2018. doi:10.1007/978-3-319-78375-8_14.

11 Dana Dachman-Soled, Julian Loss, and Adam O'Neill. Breaking rsa generically is equivalent to factoring, with preprocessing. Cryptology ePrint Archive, Paper 2022/1261, 2022. URL: https://eprint.iacr.org/2022/1261.

12 Ivan Damgård and Maciej Koprowski. Generic lower bounds for root extraction and signature schemes in general groups. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 256–271. Springer, Heidelberg, April / May 2002. doi:10.1007/3-540-46035-7_17.

13 Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and prgs. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, pages 649–665, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

14 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, Heidelberg, April / May 2017. doi:10.1007/978-3-319-56614-6_16.

15 Yevgeniy Dodis, Iftach Haitner, and Aris Tentes. On the instantiability of hash-and-sign RSA signatures. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 112–132. Springer, Heidelberg, March 2012. doi:10.1007/978-3-642-28914-9_7.

16 Andrew Drucker. New limits to classical and quantum instance compression. In *53rd FOCS*, pages 609–618. IEEE Computer Society Press, October 2012. doi:10.1109/FOCS.2012.71.

17 Amos Fiat and Moni Naor. Rigorous time/space tradeoffs for inverting functions. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 534–541, January 1991. doi:10.1145/103418.103473.

18 Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018. doi:10.1007/978-3-319-96881-0_2.

19 Martin E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Trans. Inf. Theory*, 26(4):401–406, 1980.

20 Antoine Joux, David Naccache, and Emmanuel Thomé. When e-th roots become easier than factoring. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 13–28. Springer, Heidelberg, December 2007. doi:10.1007/978-3-540-76900-2_2.

21 Jonathan Katz, Julian Loss, and Jiayu Xu. On the security of time-lock puzzles and timed commitments. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 390–413. Springer, Heidelberg, November 2020. doi:10.1007/978-3-030-64381-2_14.

22 Gregor Leander and Andy Rupp. On the equivalence of RSA and factoring regarding generic ring algorithms. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 241–251. Springer, Heidelberg, December 2006. doi:10.1007/11935230_16.

23 Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2), 1988.

24 Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Heidelberg, December 2005.

25 V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994.

26 Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978.

27 Ronald L. Rivest, Adi Shamir, and David A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, MIT, 1996.

28 Lior Rotem. Revisiting the uber assumption in the algebraic group model: Fine-grained bounds in hidden-order groups and improved reductions in bilinear groups. 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 13:1–13:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

29 Lior Rotem and Gil Segev. Generically speeding-up repeated squaring is equivalent to factoring: Sharp thresholds for all generic-ring delay functions. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 481–509. Springer, Heidelberg, August 2020. doi:10.1007/978-3-030-56877-1_17.

30 Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997. doi:10.1007/3-540-69053-0_18.

31 Aron van Baarsen and Marc Stevens. On time-lock cryptographic assumptions in abelian hidden-order groups. In *ASIACRYPT*, pages 367–397, 2021.

32 Mark Zhandry. To label, or not to label (in generic groups). In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part III*, volume 13509 of *Lecture Notes in Computer Science*, pages 66–96. Springer, 2022.

A Relevant Problems

In this section, we introduce the main relevant problems: the RSA Problem, the Factoring Problem, and the general Function Inversion Problem (all with preprocessing). Algorithm RSAGen on input 1^κ generates (N, e, d, p, q) where $N = pq$ and p, q are primes of bit-length $\kappa/2$ with leading bit 1. Finally, $ed = 1 \bmod \phi(N)$.

▶ **Definition 15** (Factoring with Preprocessing). *Let* $\mathsf{F} = (\mathsf{F}_0, \mathsf{F}_1)$ *be an algorithm and* RSAGen *be an RSA generator. Consider the factoring-with-preprocessing game* $\mathbf{fac}^{\mathsf{F}}_{\mathsf{RSAGen}}$*:*

- **Offline Phase.** *Run* F_0 *on input* 1^κ *to obtain an* advice string *state.*
- **Online Phase.** *Run* RSAGen *on input* 1^κ *to obtain*(N, e, d, p, q)*. Then run* F_1 *on input* $(N, state)$*.*
- **Output Determination.** *When* F_1 *returns* p'*, the experiment returns* 1 *if* $p = p'$ *or* $q = p'$*. It returns* 0 *otherwise.*

Define F*'s advantage in the above experiment as*

$$\mathbf{Adv}^{\mathbf{fac}}_{\mathsf{RSAGen}}(\mathsf{F}) = \Pr[\mathbf{fac}^{\mathsf{F}}_{\mathsf{RSAGen}} = 1] \, .$$

We call F *an* (S, T)-factoring algorithm relative to RSAGen *if* F_0 *outputs advice strings of size at most* S *and* F_1 *runs in time at most* T*.*

▶ **Definition 16** (RSA with Preprocessing). *Let* $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$ *be an adversary. Consider the RSA-with-preprocessing game* $\mathbf{rsa}^{A}_{\mathsf{RSAGen}}$*:*

- **Offline Phase.** *Run* A_0 *on input* 1^κ *to obtain an* advice string *state.*
- **Online Phase.** *Run* RSAGen *on input* 1^κ *to obtain*(N, e, d, p, q)*. Sample* $x \leftarrow \mathbb{Z}_N$ *and run* A_1 *on input* $(N, e, state, x^e \bmod N)$*.*
- **Output Determination.** *When* A_1 *returns* x'*, the experiment returns* 1 *if* $x = x' \pmod{N}$*. It returns* 0 *otherwise.*

Define A's advantage in the above experiment as

$$\mathbf{Adv}^{\mathbf{rsa}}_{\mathsf{RSAGen}}(\mathsf{A}) = \Pr[\mathbf{rsa}^{\mathsf{A}}_{\mathsf{RSAGen}} = 1] \, .$$

We call A *an* (S, T)-RSA algorithm relative to RSAGen *if* A_0 *outputs advice strings of size at most* S *and* A_1 *runs in time at most* T*.*

In the following, we consider a domain D of finite size along with a randomized point generator G that outputs points in D.

▶ **Definition 17** (Function Inversion with Preprocessing). *Let* D *be a finite set and let* $f \colon D \to D$ *be a function. Let* $\mathsf{I} = (\mathsf{I}_0, \mathsf{I}_1)$ *be an adversary and* Gen *a point generator. Consider the function-inversion-with-preprocessing game* $\delta^{\mathsf{I}}_{f,\mathsf{Gen}}$*:*

- **Offline Phase.** *Run* I_0 *on input* 1^κ *to obtain an* advice string *state.*
- **Online Phase.** *Run* Gen *on input* 1^κ *to obtain a point* $y \in D$*. Run* I_1 *on input* $(y, state)$
- **Output Determination.** *When* I_1 *returns* x'*, the experiment returns* 1 *if* $f(x') = y$*. It returns* 0 *otherwise.*

Define I*'s advantage in the above experiment as*

$$\mathbf{Adv}^{\delta}_{f,\mathsf{Gen}}(\mathsf{I}) = \Pr[\delta^{\mathsf{I}}_{f,\mathsf{Gen}} = 1] \, .$$

We call I *an* (S, T)-function-inversion algorithm relative to Gen *if* I_0 *outputs advice strings of size at most* S *and* I_1 *runs in time at most* T*.*

▶ **Definition 18** (Collision Probability). *Let* D *be a finite set and let* $f \colon D \to D$ *be a function. For* $z \in D$*,* $I_f(z)$ *denotes the number of preimages for* z *under* f*, i.e.*

$$I_f(z) := |\{u \in D : f(u) = z\}| \, .$$

The collision probability of $f \colon D \to D$*, denoted by* $q(f)$ *is defined as follows:*

$$q(f) := \frac{\sum_{z \in D} I_f^2(z)}{|D|^2} .$$

▶ **Theorem 19** (Fiat-Naor [17]). *For any D, f, Gen as in Definition 17 and any S, T such that $T \cdot S^2 = |D|^3 \cdot q(f)$, there is an (S,T)-function-inversion algorithm* I *such that* $\mathbf{Adv}^{\delta}_{f,\mathsf{Gen}}(\mathsf{I}) \geq 1 - 1/|D|$.[2]

B Computational Models

In this section, we review some idealized models that will be relevant in our analyses and discuss their relationships to each other.

Random Oracle Model (ROM). In the random oracle model [2] all algorithms have oracle access to a uniformly random function from $\mathsf{Func}[m_1, m_2]$ for some $m_1, m_2 \in \mathbb{N}$ specified by the model.

Random Injection Model (RIM). In the random injection model all algorithms have *forwards and backwards* oracle access to a uniformly random function from $\mathsf{FuncInj}[n, m]$ for some $n \leq m$ specified by the model.

Random Permutation Model (RPM). In the random permutation model all algorithms have *forwards and backwards* oracle access to a uniformly random function from $\mathsf{Perm}[m]$ for some $m \in \mathbb{N}$ specified by the model.

B.1 Switching from RIM to ROM

To switch from the RIM to the ROM, we need to show how to simulate oracle access to a random injection (forward and backward), given oracle access to a random function. We implement the random injection by padding the input and using Luby-Rackoff's strong pseudorandom permutation construction [23].

Luby-Rackoff. We first recall the Luby-Rackoff construction [23], which we view as a construction of a random permutation oracle from a random oracle. Formally, suppose ρ is a RO from $\{0,1\}^{m/2}$ to $\{0,1\}^{m/2}$ for $m \in \mathbb{N}$. Define oracle $\mathsf{LubRac}[\rho]$ on $\{0,1\}^m$ as follows:

- Parse x as $x_1 \| x_2$ with $|x_1| = |x_2| = m/2$ and apply a 4-round balanced Feistel network with h as the round function to obtain y. Output y.

Oracle $\mathsf{LubRack}^{-1}[\rho]$ is defined accordingly.

▶ **Theorem 20** (Luby-Rackoff [23]). *For any (even unbounded) adversary* A *making at most q queries it holds that*

$$\Big| \Pr_{\rho \leftarrow \mathsf{Func}[m/2, m/2]}[\mathsf{A}^{\mathsf{LubRack}[\rho](\cdot), \mathsf{LubRack}^{-1}[\rho](\cdot)} \textit{ outputs } 1] - \Pr_{\pi \leftarrow \mathsf{Perm}[m]}[\mathsf{A}^{\pi(\cdot), \pi^{-1}(\cdot)} \textit{ outputs } 1] \Big| \in \mathcal{O}(q^2/2^{m/2}).$$

Random Injection from Random Permutation. We next show a construction of a random injection oracle π from a random permutation oracle ψ. Suppose ψ is a random permutation oracle on m bits and ψ^{-1} is its inverse. For $n \leq m$, define $\pi[\psi] \colon \{0,1\}^n \to \{0,1\}^m$ as $\pi[\psi](x) := \psi(\mathsf{pad}(x))$ where $\mathsf{pad}(x)$ is the function that pads the LSBs of x with $m - n$ zeros. Define $\pi[\psi]^{-1}$ accordingly. It should be clear that $\pi[\psi]$ is a random injection oracle.

[2] This statement is weaker than the one proven in [17] but is sufficient for our purpose.

Now, composing the above constructions gives a construction of a random injection oracle from a random oracle. Namely, suppose $\rho\colon \{0,1\}^{m/2} \to \{0,1\}^{m/2}$ is a RO. Define the random injection oracle $\pi[\rho]\colon \{0,1\}^n \to \{0,1\}^m$ as $\pi[\rho](x) = \mathsf{LubRac}[\rho](\mathsf{pad}(x))$ and $\pi[\rho]^{-1}$ accordingly. By a simple hybrid argument we have:

▶ **Proposition 21** (RIM-to-ROM). *For any (even unbounded) adversary* A *making at most q queries it holds that*

$$|\Pr_{\rho \leftarrow \mathsf{Func}[m/2,m/2]}[A^{\pi[\rho](\cdot),\pi[\rho]^{-1}(\cdot)} \textit{ outputs } 1] - \Pr_{\pi \leftarrow \mathsf{FuncInj}[n,m]}[A^{\pi(\cdot),\pi^{-1}(\cdot)} \textit{ outputs } 1]| \in \mathcal{O}(q^2/2^{m/2}).$$

B.2 Straight-Line Programs and Generic Ring Algorithms

Let $N \in \mathbb{N}$ and assume that $m \geq \kappa$, where κ is the bit length of N. Below, we define two types of programs (aka. algorithms) that use oracles, namely generic-ring algorithms (GRAs) and straight-line programs (SLPs).

Program Graphs and Their Execution. The below is based on [1]. We consider deterministic programs that perform arithmetic operations (mod N) on indeterminate Y.

We associate a program on a single input with its *program graph over* $\mathbb{Z}_N$, a labelled graph where a label of a node represents a (binary) operation and the program implicitly stores all intermediate results. We only consider programs whose graphs are binary trees. Vertices can be either *branching* or *non-branching.*

Execution of a program corresponds to traversing a labelled path in its program graph over $\mathbb{Z}_N$. *Non-branching vertices* are used to execute arithmetic operations (mod N) or to load inputs and constants into the program. They are accordingly labelled with elements $a \in \mathbb{Z}_N$ corresponding to constants in the program, with a (unique) indeterminate Y corresponding the programs input, or with an arithmetic operation label $(i, j, \circ, b)$ which applies the arithmetic ring operation $\circ$ (mod N) to operands at indices i and j that the program previously stored. (The flag $b \in \{-1, 1\}$ indicates inversion of the second operand.) *Branching vertices* are used to test two values i, j previously computed by the program for equality (mod N). A branching vertex has two outgoing edges that are labelled 0 (for left) and 1 (for right).

The program applies the operations indicated by the labels of the vertices and edges it encounters in the order of traversal as follows:

- The first three vertices are a path and are always labelled 0, 1, and Y. That is, they are used to load the constants 0 and 1, and the single input y of the program. The program stores the intermediate results $y_0 = 0, y_1 = 1, y_2 = y$ for these vertices, respectively, and continues execution along this path.
- For $k \geq 4$:
  - If the kth vertex v_k is labelled with $a \in \mathbb{Z}_N$, the program stores $y_k \leftarrow a$ as the intermediate result for this vertex. It continues execution along this path.
  - If the kth vertex v_k is labelled with $(i, j, \circ, b)$ then the program does as follows. Here $\circ \in \{\cdot, +\}, b \in \{-1, 1\}$, and $i, j < k$ correspond to the ith and jth vertices on the path of traversal, which must be non-branching. The program computes $y_k := y_i \circ y_j^b$ (mod N) and stores the intermediate result y_k for vertex v. In case $\circ = +$ and $b = -1$, then $y_j^b = -y_j$ (mod N). In case $\circ = \cdot$, $b = -1$, and $y_j = 0$ (mod N), $y_k := \perp$. In case $y_i = \perp$ or $y_j = \perp$, $y_k := \perp$. It continues execution along this path.

  - If the kth vertex v_k is labelled (i,j) where $i,j < k$ correspond to the ith and jth vertices on the path of traversal, which must be non-branching, the program makes an *equality test* whether $y_i = y_j \pmod N$. If the result is 1, the program continues its execution along the right edge; otherwise, along the left.
- Whenever v_k is the last vertex on the path, the program computes y_k and outputs it, terminating execution.

Oracle-Aided Programs. Apart from the types of programs we have discussed above, we are also interested in programs that can perform arithmetic operations via oracle access (as opposed to directly).

Hence, we define oracles π, eq, and op_π as follows. Oracle π initially samples a random function $\pi \in \mathsf{FuncInj}[\kappa, m]$ and on query $x \in \mathbb{Z}_N$ returns $y = \pi(x) \in \{0,1\}^m$. Here we refer to $y \in \{0,1\}^m$ as a *label*. We slightly abuse notation by referring to the oracle π and the internally sampled function indiscriminately. We also make the convention of parsing $x \in \mathbb{Z}_N$ as a κ-bit binary string. Given $\pi \in \mathsf{FuncInj}[\kappa, m]$, we first consider an oracle eq for testing equality. On input $y_1, y_2 \in \{0,1\}^m$, eq returns 1 iff $\pi^{-1}(y_1) = \pi^{-1}(y_2) \pmod N$, and 0 otherwise. Now, we define the behavior of the *ring oracle* op_π on input as $y_1, y_2 \in \{0,1\}^m$ as

$$\mathsf{op}_\pi(y_1, y_2, \circ, b) := \pi\left(\pi^{-1}(y_1) \circ \left(\pi^{-1}(y_2)\right)^b \bmod N\right)$$

for all $y_1, y_2 \in \{0,1\}^m$, $\circ \in \{+, \cdot\}$, $b \in \{1, -1\}$, where the inverse is additive in case $\circ = +, b = -1$. We implicitly assume that in case $\circ = \cdot, b = -1$, op_π internally queries $\mathsf{eq}(y_2, 0)$. op_π returns $\perp$ in case either of the operands is $\perp$ or the call to eq returns 1, i.e., if $\pi^{-1}(y_2) = 0 \pmod N$.

▶ Remark 22. Throughout the paper, when there is no possibility of confusion we abbreviate oracles op_π and eq_π by π. That is, for an oracle-aided program P we abbreviate $P^{\mathsf{op}_\pi, \mathsf{eq}_\pi}$ by P^π.

Oracle-aided program graphs over $\mathbb{Z}_N$ are labelled very similarly to plain program graphs over $\mathbb{Z}_N$. Roughly speaking, all values in $\mathbb{Z}_N$ are now replaced with their labels, according to π. Thus, a non-branching vertex is now labelled in one of two ways. Either it is labelled with $(i, j, \circ, b)$ where i and j correspond to the ith and jth non-branching vertex among the vertices previously encountered on the path and $\circ \in \{+, \cdot\}, b \in \{1, -1\}$. Otherwise, it is labelled with some m-bit label σ in the image of π.

As before, a branching vertex is labeled with (i, j), where i and j correspond to the ith and jth non-branching vertex among the vertices previously encountered on the path. It has two outgoing edges labelled 0 (for left edge) and 1 (for right edge). The only difference is that the program now has to invoke eq on the intermediate values y_i and y_j so as to test their equality (rather simply testing whether they are equal).

Execution of an oracle-aided program corresponds to its program graph by adapting the above correspondence in the straight forward manner:

- The first two nodes on a path are always labelled as $\pi(0), \pi(1)$, respectively; that is, they are used to load the constants 0 and 1. The third node on a path is used to load the (single) input $\pi(y)$ to the program. It is labelled with a special label ϕ. The program stores the intermediate results $y_0 = 0, y_1 = 1, y_3 = \pi(y)$ for these vertices, respectively, and continues execution along this path.
- When the program encounters a non-branching vertex v:
  - If v is labelled with $(i, j, \circ, b)$, where i, j are indices and $b \in \{0, 1\}$, and this is the kth non-branching vertex on the path of traversal for some $k \geq 4$, and, then the program invokes the oracle op_π on input $(y_i, y_j, \circ, b)$. It stores the output of op_π as y_k.

  - If v is labelled with σ and this is the kth non-branching index on the path of traversal for some $k \geq 4$, store $y_k \leftarrow \sigma$ and continue the execution of the program along this path.
- If the program encounters a branching vertex v: if v is labelled (i, j), the program invokes the oracle eq on input (y_i, y_j). If the result is 1, the program continues its execution along the right edge; otherwise, along the left.
- If k is the last vertex on the path, the program outputs y_k and terminates.

Types of Programs. We define two types of programs:

▶ **Definition 23.** *A T-step (possibly oracle-aided)* straight line program (SLP) S over $\mathbb{Z}_N$ *is a program whose program graph over $\mathbb{Z}_N$ is a labelled path $v_0, \ldots, v_{T+3}$.*

A deterministic generic ring algorithm (GRA) is a generalization of SLPs that allows equality tests. As explained above, such queries are represented as branching vertices in our graph representation of a GRA. Thus, an SLP can be seen as special case GRA, where an SLP is a GRA that contains no branching vertices.

▶ **Definition 24.** *A T-depth deterministic (possibly oracle-aided)* generic ring algorithm (GRA) G over $\mathbb{Z}_N$ *is a program whose program graph over $\mathbb{Z}_N$ is a depth-$(T+3)$ vertex-labelled and partially edge-labelled binary tree.*

To keep the distinction between oracle aided vs. regular programs clear, we will always make the dependency on π explicit by superscripting oracle-aided programs with π, i.e., G^π.

The following definition applies only to non-oracle aided programs. It inductively defines the polynomial corresponding to an execution of a program on input $x \in \mathbb{Z}_N$. Essentially, if the program encounters a non-branching vertex v and v corresponds to an arithmetic operation, then we associate the resulting tuple $(P_v^G(x), Q_v^G(x))$ with vertex v. Here, $P_v^G(x)$ and $Q_v^G(x)$ are interpreted as the numerator and denominator of a rational function $P_v^G(x)/Q_v^G(x)$ that is the result of applying the arithmetic operation to the rational functions associated with prior vertices w, u.

▶ **Definition 25.** *For a GRA G (or SLP S) over $\mathbb{Z}_N$ of size T and non-branching vertex v in its execution graph, the pair $(P_v^G(x), Q_v^G(x))$ of polynomials in $\mathbb{Z}_N[x]$* associated with v *is defined inductively, as follows:*

1. *The root has associated the pair $(0, 1)$, the child of the root the pair $(1, 1)$, and the child of that child has the pair $(x, 1)$.*
2. *A vertex v labelled with $a \in \mathbb{Z}_N$ is associated with $(a, 1)$.*
3. *For each non-branching vertex v, labelled with operation $(u, w, +, b)$, we have:*

$$(P_v^G(x), Q_v^G(x)) :=$$
$$\begin{cases} (P_u^G(x) \cdot Q_w^G(x) + P_w^G(x) \cdot Q_u^G(x), Q_u^G(x) \cdot Q_w^G(x)) & b = 1 \\ (P_u^G(x) \cdot Q_w^G(x) - P_w^G(x) \cdot Q_u^G(x), Q_u^G(x) \cdot Q_w^G(x)) & b = -1 \end{cases}$$

4. *For each non-branching vertex v, labelled with operation $(u, w, \cdot, b)$, we have:*

$$(P_v^G(x), Q_v^G(x)) :=$$
$$\begin{cases} (P_u^G(x) \cdot P_w^G(x), Q_u^G(x) \cdot Q_w^G(x)) & b = 1 \\ (P_u^G(x) \cdot Q_w^G(x), Q_u^G(x) \cdot P_w^G(x)) & b = -1, Q_u^G(x) \neq 0 \pmod{N} \\ \perp & b = -1, Q_u^G(x) = 0 \pmod{N} \end{cases}$$

Note that $P_v^G(x)$ and $Q_v^G(x))$ can each be represented as an SLP of size at most T.

▶ **Definition 26.** *For an SLP S over $\mathbb{Z}_N$ of size T, we denote by $(P^S(x), Q^S(x))$ the pair of polynomials in $\mathbb{Z}_N[x]$* associated with the final vertex on the evaluation path. *If $Q^S(x) \equiv 1$, we denote by f_S the polynomial $P^S(x)$. Note that $P^S(x)$ and $Q^S(x))$ can each be represented as an SLP of size at most T.*

▶ **Definition 27.** *For each non-branching vertex v in the program graph over $\mathbb{Z}_N$ of an ℓ-step GRA G with corresponding pair of polynomials $(P_v^G(a), Q_v^G(a))$, we associate the function*

$$f_v^G : \mathbb{Z}_N \to \mathbb{Z}_N \cup \{\perp\} : a \mapsto \frac{P_v^G(a)}{Q_v^G(a)}$$

where the function is undefined if $Q_v^G(a) = 0$, which is denoted as $f_v^G(a) = \perp$, and where $P_v^G(a)$ and $Q_v^G(a)$ are evaluated over $\mathbb{Z}_N$. Moreover, for an argument $a \in \mathbb{Z}_N$, the computation path from the root v_0 to a leaf $v_{\ell+3}(a)$ is defined by taking, for each equality test of the form (u, w), the edge labeled 0 if $f_v^G(a) = f_w^G(a)$, and the edge labeled 1 if $f_u^G(a) \neq f_w^G(a)$. The partial function f^G computed by G is defined as

$$f^G : \mathbb{Z}_N \to \mathbb{Z}_N \cup \{\perp\} : a \mapsto f^G_{v_{\ell+3}(a)}.$$

We define the output of G on input $x \in \mathbb{Z}_N$ as $G(x) := f^G(x)$.

B.3 Model Specific Versions of the RSA Assumption

We introduce a new variant of the RSA game with preprocessing model specifically tailored to the oracle-aided computational models from the previous section. In the following, we fix the security parameter κ and an integer $m \in \mathbb{Z}, m \geq \kappa$.

▶ **Definition 28** (Generic RSA Problem with Preprocessing). *For a tuple of algorithms $\mathsf{A} = (\mathsf{A}_0^\pi, \mathsf{A}_1)$ and an RSA instance generator RSAGen, define experiment $\text{ag-rsa}^{\mathsf{A}}_{\mathsf{RSAGen}}$ as follows:*

- **Offline Phase.** *Sample $\pi \leftarrow \mathsf{FuncInj}[\kappa, m]$. Run A_0^π on input 1^κ. Let state denote the return value of A_0^π.*
- **Online Phase.** *Compute $(N, e, d) \leftarrow \mathsf{RSAGen}(1^\kappa)$ and sample $x \leftarrow \mathbb{Z}_N$. Run A_1 on input $(N, e, \pi(x^e), state)$ and let G^π denote the output. If G^π does not correspond to the description of a GRA, abort. Note that A_1 does* not *get access to oracle π.*
- **Output Determination.** *Run G^π on input $(N, e, \pi(x^e))$. When G^π outputs $z \in \{0,1\}^m$, the experiment evaluates to 1 iff $z = \pi(x)$.*

Define A's advantage relative to RSAGen as

$$\mathbf{Adv}^{\text{ag-rsa}}_{\mathsf{RSAGen}}(\mathsf{A}) = \Pr[\text{ag-rsa}^{\mathsf{A}}_{\mathsf{RSAGen}} = 1] .$$

We call $\mathsf{A} = (\mathsf{A}_0^\pi, \mathsf{A}_1)$ an (S, T_1, T_2)-generic-RSA-with-preprocessing algorithm (GP-RSA) relative to RSAGen *if A_0^π outputs advice strings state of size at most S, A_1 runs in time at most T_1, and and any program G^π in the output of A_1 runs in time at most T_2. Note that we require that $T_1 \geq T_2$.*

We also give an alternative version of this game in which $\pi \in \mathsf{FuncInj}[\kappa, m]$ and $(N, e, d) \in \mathsf{RSAGen}(1^\kappa)$ are a fixed.

▶ **Definition 29** (Fixed Generic RSA Problem with Preprocessing). *Fix integers $(N, e, d) \in \mathsf{RSAGen}(1^\kappa)$, let $\pi \in \mathsf{FuncInj}[\kappa, m]$, and let state be of size at most S. Define experiment $\mathbf{fcrsa}^{\mathsf{A}}_{(N,e,d,state,\pi)}$ as follows:*

- **Online Phase.** *Sample $x \leftarrow \mathbb{Z}_N$. Run A on input $(N, e, \pi(x^e), state)$ and let G^π denote the output. If G^π does not correspond to the description of a GRA, abort. Note that A does not have oracle access to π.*

- **Output Determination.** *Run G^π on input $(N, e, \pi(x^e))$. When G^π outputs $z \in \{0,1\}^m$, the experiment evaluates to 1 iff $z = \pi(x)$.*

Define A's advantage relative to $(N, e, d, state, \pi)$ as

$$\mathbf{Adv}^{\mathbf{fcrsa}}_{(N,e,d,state,\pi)}(\mathsf{A}) = \Pr[\mathbf{fcrsa}^{\mathsf{A}}_{(N,e,d,state,\pi)}(1^\kappa) = 1],$$

*We call A an (S, T_1, T_2)-*fixed-generic-RSA-with-preprocessing (FGP-RSA) algorithm relative to *$(N, e, d, state, \pi)$ if A runs in time at most T_1, and and any program G^π in the output of A runs in time at most T_2.*

Note that in the above definition we do not require the advice string *state* to be output by a preprocessor A_0^π. However, by a standard averaging argument, we obtain the following lemma:

▶ **Lemma 30.** *Let $\mathsf{A} = (\mathsf{A}_0^\pi, \mathsf{A}_1)$ be an (S, T_1, T_2)-GP-RSA algorithm and suppose that $\mathbf{Adv}^{\text{ag-rsa}}_{\mathsf{RSAGen}}(\mathsf{A}) \geq \epsilon$. Then with probability at least $\epsilon/2$ over the coins of RSAGen, the choice of π, and coins of A_0^π, A_0^π outputs state and RSAGen outputs (N, e, d) s.t. $\mathbf{Adv}^{\mathbf{fcrsa}}_{(N,e,d,state,\pi)}(\mathsf{A}_1) \geq \epsilon/2$.*

Time-Space Tradeoffs for Finding Multi-Collisions in Merkle-Damgård Hash Functions

Akshima
NYU Shanghai, China

Abstract

We analyze the multi-collision resistance of Merkle-Damgård hash function construction in the auxiliary input random oracle model. Finding multi-collisions or m-way collisions, for some parameter m, in a hash function consists of m distinct input that have the same output under the hash function. This is a natural generalization of the collision finding problem in hash functions, which is basically finding 2-way collisions. Hardness of finding collisions, or collision resistance, is an important security assumption in cryptography. While the time-space trade-offs for collision resistance of hash functions has received considerable attention, this is the first work that studies time-space trade-offs for the multi-collision resistance property of hash functions based on the popular and widely used Merkle-Damgård (MD) constructions.

In this work, we study how the advantage of finding m-way collisions depends on the parameter m. We believe understanding whether multi-collision resistance is a strictly easier property than collision resistance is a fundamental problem and our work facilitates this for adversaries with auxiliary information against MD based hash functions. Furthermore, in this work we study how the advantage varies with the bound on length of the m colliding inputs. Prior works [1, 19, 3] have shown that finding "longer" collisions with auxiliary input in MD based hash functions becomes easier. More precisely, the advantage of finding collisions linearly depends on the bound on the length of colliding inputs. In this work, we show similar dependence for m-way collision finding, for any $m \geq 2$.

We show a simple attack for finding 1-block m-way collisions which achieves an advantage of $\tilde{\Omega}(S/mN)$. For $2 \leq B < \log m$, we give the best known attack for finding B-blocks m-way collision which achieves an advantage of $\tilde{\Omega}(ST/m^{1/(B-1)}N)$ when $m^{1/(B-1)}$-way collisions exist on every salt. For $B > \log m$, our attack achieves an advantage of $\tilde{\Omega}(STB/N)$ which is optimal when $SB \geq T$ and $ST^2 \leq N$. The main results of this work is showing that our attacks are optimal for $B = 1$ and $B = 2$. This implies that in the auxiliary-input random oracle model, the advantage decreases by a multiplicative factor of m for finding 1-block and 2-block m-way collisions (compared to collision finding) in Merkle-Damgård based hash functions.

2012 ACM Subject Classification Security and privacy → Cryptography; Security and privacy → Information-theoretic techniques; Mathematics of computing → Probability and statistics

Keywords and phrases Collision, hash functions, multi-collisions, Merkle-Damgård, pre-computation, auxiliary input

Digital Object Identifier 10.4230/LIPIcs.ITC.2024.9

Funding Most of this work was done while Akshima was a PhD student at University of Chicago and was supported in part by NSF Grant No. 1928767. Akshima is supported by Ministry of Education of P.R.C., through an Overseas Postdoc Special Program (2023), National Natural Science Foundation of China Grant No. 6210226 and NYTP Grant No. 2012120. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding institutions or the governments.

Acknowledgements We thank the anonymous reviewers for their constructive comments on earlier drafts of the work. We thank David Cash and Siyao Guo for useful discussions.

1 Introduction

In this work we study multi-collision finding in Merkle-Damgård (MD) hash functions with auxiliary input generated during a pre-computation. Several recent works [11, 12, 1, 19, 3] have rigorously studied collision finding in MD hash functions with pre-computation but currently nothing is known about the upper and lower bounds for m-way collision finding with pre-computation, where an m-way collision consists of m distinct messages, for some parameter m, whose hash values are identical.

Auxiliary-Input Random Oracle Model

We will study the problem in the auxiliary-input(AI) random oracle (RO) model in order to obtain the time-space trade-offs. We note that AI-RO model is strong enough to capture pre-computing as well as non-uniform adversaries. We informally describe the model next.

An adversary $\mathcal{A}$ in this model can be thought of as a pair of algorithms $(\mathcal{A}_1, \mathcal{A}_2)$ each of which corresponds to a stage of $\mathcal{A}$:

1. Offline/pre-computation stage: $\mathcal{A}_1$ gets computationally unbounded access to the oracle (i.e., there is no bound on the number of oracle queries and computation made by $\mathcal{A}_1$) and outputs a fixed-size advice/information about the oracle. Let's denote this pre-computed information by σ.
2. $\mathcal{A}_2$ takes σ outputted by $\mathcal{A}_1$ as input along with challenge, makes a bounded number of queries to the oracle.

In this work, we will use S to denote the length of σ in bits output by $\mathcal{A}_1$ and T to denote the bound on the number of queries made by $\mathcal{A}_2$. We will refer to such an adversary $\mathcal{A}$ as (S, T)-AI adversary.

Specifically for a m-way collision finding (S, T)-AI adversary, the input challenge to $\mathcal{A}_2$ consists of a salt for the hash function and $\mathcal{A}_2$ needs to output m colliding messages on the salt to win. There could be an additional restriction that the collisions found are at most B blocks long.

Why Study Multi-collisions

Multi-collision resistance is a natural generalization of a fundamental and widely used security property in cryptography, namely collision resistance. Multi-collision resistance property is seen as a relaxation of the collision resistance and it is a meaningful exercise to understand whether former is strictly an easier property than the standard collision resistance. In a recent work [29], Rothblum and Vasudevan studied this and gave a non-black box and non-constructive transformation from a multi-collision resistant hash function to collision resistant hash function. In this work, we understand how the complexity of the m-way collision finding in MD hash functions varies with m for any (S, T)-AI adversary. If the complexity grows linearly or exponentially in m, then it could be better to reduce the security of protocols to m-way collisions for larger m's. Reducing the security to m- way collision resistance for larger and larger m could allow applications to reduce the length of the output of the hash function and hence make them more efficient.

For a random hash function (i.e., a hash function modelled as a random oracle) it was proven that any adversary making T queries can achieve the advantage of the order $(T/m)^m/N^{m-1}$ for finding m-way collision. Note that the advantage decreases exponentially in the size of m. Works dating as early as 1990s had identified that finding m-way collisions gets harder as m becomes larger and inspired by this observation, the notion of Multi-collision resistance in hash functions had been used in several existing constructions of applications such as digital signatures, commitment scheme and more.

The multi-collision resistance primitive has been used in signature schemes [8], identification schemes [20], micropayment scheme [28] and commitment scheme [26]. Bitansky et al. in [7], formally introduced multi-collision resistance as a cryptographic primitive, used it to construct secure keyless hash functions and study its applications. Berman et al., too, in [5] used the notion of multi-collision resistance to find secure keyless hash functions.

Liu and Zhandry in [27] analyzed multi-collisions in the quantum setting. As Grover's algorithm speeds up the classical birthday attack for finding collisions from $O(2^{n/2})$ to $O(2^{n/3})$, Liu and Zhandry study the speedup on multi-collisions in random hash function with quantum query. They conclude that it could be better to base the security of constructions on the multi-collision resistance in the quantum setting as they prove that $\Omega(2^{n/2\cdot(1-1/(2^m-1))})$ queries are required on average to find m-way collision for any constant m.

While it can be provably shown for random functions that finding m-way collisions becomes exponentially unlikely in m (the probability of finding m-way collisions is $\theta(T^m/N^{m-1})$), same cannot be said to hold true for iterated hash function constructions such as Merkle-Damgård. In fact Joux in [25], showed an attack that requires just $\log m \cdot \sqrt{N}$ queries in expectation to find m-way collision in Merkle-Damgård based hashing algorithms. Note that this means that the number of queries required to find m-way collisions in Merkle-Damgård based hashing algorithms increases by just a multiplicative factor of $\log m$ with m (unlike for random functions).

Here we note that Joux's result does not take into account two things: 1) What if there is a restriction on the length of permitted collisions? 2) What if adversaries allowed to perform pre-computation or learn some auxiliary information about the hash function? To the best of our knowledge, our work is the first to investigate on both of these fronts simultaneously.

From our investigation of the problem with pre-computation, we learn that finding m-way collisions in MD based hash functions do not get harder in the Auxiliary input random oracle (AI-RO) model as far as there is no restriction on the length of m-way collisions found. But what about when there is a restriction on the length of the m-way collisions found with pre-computation? The biggest question this work investigates is about the hardness of finding "short" m-way collisions. In fact, for some restricted parameter ranges we manage to show that any pre-computing adversary's advantage provably takes a linear hit as m grows larger.

It is worth-noting that m-way collision finding is somewhat related to the fundamental m-distinctness problem. The difference is that our problem studies collisions on the same salt whereas m-distinctness applies even when there is an m-way collision with different salts. Nevertheless both the problems are looking for m-way collisions. In fact, solving m-distinctness is a trivial problem in the AI-RO model as long as the advice σ is $\Omega(m \log N)$ bits long. This is because $\mathcal{A}_1$ can simply find the m-way collision on the hash function and store it as advice for $\mathcal{A}_2$ to win certainly. Thus, it is important that $\mathcal{A}_2$ gets a random salt as challenge and the outputted m messages should collide with this salt under the hash function for $\mathcal{A}$ to win.

1.1 Our Contributions

1.1.1 Our Results

First, we summarize our results for m-way collision-finding in table 1. As stated before, let S denote the size of pre-computed information in bits, T be the number of the queries made to oracle implementing a random function h in $[N] \times [M] \to [N]$ and B be the number of blocks accepted in the multi-collisions.

Table 1 Asymptotic security bounds on the security of finding B-block m-way collisions in Merkle-Damgård Hash Functions constructed from a random function $h : [N] \times [M] \mapsto [N]$ against (S, T)-algorithms. The attacks assume M is "sufficiently" larger than N for the required collisions to exist in h. For simplicity, logarithmic terms and constant factors are omitted.

B	Best attacks	Security bounds
$B = 1$	$\frac{S}{mN} + \frac{(T/m)^m}{N^{m-1}}$ [Thm 10]	$\frac{S}{mN} + \frac{(2T/m)^m}{N^{m-1}}$ [Thm 10]
$B = 2$	$\frac{ST}{mN} + \frac{(T/m)^m}{N^{m-1}}$ [Thm 8]	$\frac{ST}{mN} + \frac{(eT/m)^m}{N^{m-1}}$ [Thm 11]
$B < \log m$	$\frac{ST}{m^{1/(B-1)}N} + \frac{(T/m)^m}{N^{m-1}}$ [Thm 8]	$\frac{STB}{N} \cdot \max\{1, \frac{ST^2}{N}\} + \frac{T^2}{N}$ [3]
$[\log m, 2\log m)$	$\frac{ST}{N} + \left(\frac{T^2}{N \log^2 m}\right)^{\log m}$ [Thm 8]	$\frac{STB}{N} \cdot \max\{1, \frac{ST^2}{N}\} + \frac{T^2}{N}$ [3]
$[2\log m, T]$	$\frac{STB}{N} + \left(\frac{T^2}{N \log^2 m}\right)^{\log m}$ [Thm 8]	$\frac{STB}{N} \cdot \max\{1, \frac{ST^2}{N}\} + \frac{T^2}{N}$ [3]
Unbounded	$\frac{ST^2}{N}$ [Thm 7]	$\frac{ST^2}{N}$ [11]

We elaborate our results next.

1. For unbounded B (or $B \geq T$), a reduction to the collision-finding problem upper bounds the advantage to $\tilde{O}(ST^2/N)$ using a result from [12]. Our interesting finding is a matching attack based on the attacks of Joux in [25] and Coretti et al. in [12].
   Finding an attack that matches this trivial security bound up to poly-log factors is surprising because this attack shows that finding m-way collision in the AI model requires about the same effort for any $m \geq 2$. To put this into more perspective, this attack shows that an AI adversary can find an $\sqrt{N}$-way collision in about the same effort as required for finding a 2-way collision.
2. For any other B, the best we manage to do is again bound the security via reduction to the collision-finding problem as for unbounded B. The bounded-length version of our attack from (1) matches the bound when $B \geq \log m, SB \geq T$ and $ST^2 \leq N$.
   For $B < \log m$, the best attack we can find has a gap of a factor of $m^{1/(B-1)}$. To elaborate, we give attack that achieves:
   - advantage $\tilde{\Omega}\left(\frac{ST}{m^{1/(B-1)}N}\right)$ for any $B > 1$ when $M > (m^{1/(B-1)} - 1) \cdot N$ and $m = O(N^c)$ for some constant c.
     We restrict to $M > (m^{1/(B-1)} - 1) \cdot N$, so that an $m^{1/(B-1)}$-way collision is guaranteed to exist on every salt in $[N]$ under h by the pigeonhole principle. However, $M > (m^{1/(B-1)} - 1) \cdot N$ is not a necessary condition for such collisions to exist. It is the case that our attack succeeds with advantage $\tilde{\Omega}\left(\frac{ST}{m^{1/(B-1)}N}\right)$ when $m^{1/(B-1)}$-way collisions exist on every salt in $[N]$.
   - advantage $\tilde{\Omega}\left(\frac{STB}{N}\right)$ when $B \geq (d+1)\log m$ and $M^d > N$ for some $d \geq 1$.
     Let's first understand this for $d = 1$. It means we restrict to $M > N$ but again this is not a necessary condition. Having 2-way collisions on every salt is sufficient for our attack to achieve $\tilde{\Omega}\left(\frac{STB}{N}\right)$ advantage. To get rid of the $M > N$ requirement, we can set d to be the smallest value such that $M^d > N$ and replace h with h^d. (Hence, $B \geq (d+1)\log m$.) This is so that the offline adversary will be able to find 2-way collisions on every salt under h^d by the pigeonhole principle. Note that the smallest d which satisfies this requirement will be at most $\log N + 1$ for any $M \geq 2$.

3. For $B = 1$, we show that advantage for any (S,T)-AI adversary is bounded by $\tilde{O}\left(\frac{S}{mN} + \frac{T^m}{N^{m-1}}\right)$. Note that this bound shows a loss of factor of m in advantage when S/mN is the dominating term. The trivial attack stated in section 5 matches this bound. The proof for this result is via generalizing a technique used in [16], namely "global" compression.
4. Our result for $B = 2$ is the most important contribution of this work in terms of techniques used. We show that no (S,T)-AI adversary can achieve better advantage than $\tilde{O}\left(\frac{ST}{mN}\right)$. This bound again shows a loss of factor of m in advantage and matches our attack from (2).
   Several prior works have used the relation between the advantage of solving a problem in the AI model and another model, namely the multi-instance model (MI), so that they analyze the MI-security and obtain bounds for the AI-security. However, we reduce the problem to finding m-way collision in a variant of the "parallel" multi-instance model (MI). It is the way we relate the security in the two models which is new and we believe could be of independent interest.

Our results suggest that finding m-way collisions doesn't get harder (as m increases) in the AI-RO model if the collisions are allowed to be sufficiently long or m is really "small" (for instance $O(\log^c N)$ for some constant c because we ignore poly-log factors in our results). From the practice standpoint, this could mean that if the constructors of applications want to gain in complexity by reducing their security to larger collision finding, it is imperative to make sure the relative size of the parameters N, m and B in their application actually makes that plausible.

Discussion

As in several prior works [24, 1, 10, 19, 3], we also use that the security in the auxiliary input RO model is closely related to the security in another model that is easier to analyze, namely the Multi-instance (MI) model. How we relate these two models in this work is different from the prior works. This allows our analysis of the Multi-instance model to be significantly different and easier compared to the prior works.

We informally describe the "Parallel" MI model next and follow up with a high level idea of our technique. We note that there is a "Sequential" variation of the MI model that has been extensively used in several prior works but this work only uses the Parallel variation. So whenever we refer to MI model in this paper, we mean the Parallel MI model.

Informally, an adversary in the MI model gets multiple instances of challenges and gets to make bounded number of queries to the oracle for each challenge instance. The adversary gets no advice and it is required to succeed on every challenge instance in order to succeed. Specifically, our parameters in the MI model will be S, T where S will denote the number of challenge instances given to the adversary (note the parameter S here corresponds to the size the advice from AI model for optimal reduction) and T will denote the number of queries the adversary can make for each instance. Note that the adversary will get to make a total of ST queries.

Our high level approach would be to identify all the types of 2-block m-way collisions. For 2-block m-way collision finding, there are several types of collisions which an adversary could find in order to win. We depict the types in fig. 4. (Note that an adversary could find a collision that is a mix of these types but there will exist an $(c \cdot m)$-way collision of one type where c is a constant.) For a straight-forward AI to MI security reduction used in prior works, we need to analyze adversaries that could find any one of these five types of m-way collisions for each of the S challenge salts in MI model and win. However, analyzing such adversaries seems very hard.

We will relate the AI-security to the MI-security for each of these type of collisions. This allows us to analyze adversary restricted to finding a certain type of collision for all the S challenge instances in the MI model. This is the idea that makes the analysis in the MI model possible. We would like to note here that we are able to do per collision type of relation of AI-security to MI-security because our analysis in the MI model is insensitive to the order in which the blocks of colliding messages are found. For full details refer to section 6.

1.2 Related Works

We would like to first note that time-space lower bounds have been studied for several cryptographic primitives including collision resistance. Some other primitives are function inversion, discrete log, one-way functions, pseudo-random generators. Refer [11, 15, 9, 14, 13, 22, 21] for further details.

Here we will focus on the prior works for collision resistance in the auxiliary input model. As far as we know, all the prior works have focused on time-space trade-offs for 2-way collisions. Ours is the first work that studies the more general primitive, m-way collisions in hash functions.

Before discussing the prior works, we recall our notations. The adversary in the AI model works in two stages. The output of the first (offline) stage adversary is bounded length advice σ. We will denote this length by S-bit, i.e., $|\sigma| = S$. The adversary in the second (online) stage makes bounded number of queries to the oracle, denoted by T. The adversary could be required to output collisions that have bounded length. We will denote this bound on the length by B.

Dodis et al. in [16] were the first to study 2-way collision-resistance in AI-RO model. Dodis et al. studied it for random functions and proved the security bound of $\tilde{\theta}(S/N + T^2/N)$. They presented an attack and proved the matching security bound via “global” compression. To elaborate further, the authors showed that any pre-computing adversary that finds collisions on “too many” salts can be used to compress the random function. Since random functions cannot be compressed, this can be used to bound the size of the set of winning salts for any adversary. Our proof of security bound for 1-block m-way collisions is based on this technique of Dodis et al..

Coretti et al. in [12] were the first to study (2-way) collision-resistance for Merkle-Damgård based hash function in the AI-RO model. Coretti et al. proved a bound of $O(ST^2/N)$ on finding collisions when there is no restriction on the length of the collisions. Coretti et al. reduced the security of collision finding in the AI-RO model to another model, namely Bit-fixing random oracle (BF-RO) model which was inspired by the work of Unruh [30]. The matching attack presented in [12] was inspired by the Hellman's attack presented in the seminal work [23] and found collisions that could be order of T blocks long.

The follow-up work of Akshima et al. [1] realized that when accepted collisions were restricted to B-blocks in length, the best attack they could find achieved an advantage of $\Omega(STB/N)$ and they conjectured it to be the optimal attack. However, Akshima et al. could prove the optimality of their attack only for a restricted class of pre-computing adversaries and not any pre-computing adversary. For any pre-computing adversary, they could show their attack to be optimal only when $B = 2$. The proof in [1] reduced to Sequential multi-instance game and proved the required bound via compression. In addition, Akshima et al. presented an attack that showed it was impossible to achieve the desired bound (bound that would give optimal bound in AI-RO model) in the parallel multi-instance game, effectively proving a gap between the two versions of the multi-instance game for collision-finding.

The follow-up work of Ghoshal and Komargodski [19] proved the conjecture in [1] for any constant B. Ghoshal et al. used similar techniques to those in [1] together with their observation that it is "unlikely" to find $\geq \log N$-way collision in a random function to obtain better results.

Inspired by the idea of [10] to analyze the sequential multi-instance game stage-wise instead of simultaneously analyzing all the S stages, Akshima et al. in [3] proved the conjecture of [1] for any $B \in [2, T)$ when $ST^2 \leq N$. They proved a bound of $STB/N \cdot \max\{1, ST^2/N\}$. Our bounds for "long" m-way collisions is based on this result. Our attack for $B \geq \log m$ blocks long m-way collision, as in [3], matches the bound (up to poly-log factors) when $ST^2 \leq N$.

Freitag et al. in [17] studied 2-way collision-finding for Sponge based hash functions in the AI random permutation model. They presented an attack for $B = 1$ which uses inverse queries of permutation to beat the trivial attack for some parameter ranges. They also prove loose security bounds for $B = 1$ and $B = 2$ block collision finding. Their bound for $B = 1$ was improved in the follow-up work [2]. In [2], Akshima et al. also showed that relation between AI security and multi-instance security cannot be used to improve the security bounds any further for Sponge based constructions.

A recent work [18] by Freitag et al. uses Merkle trees to get a provably improved hash function construction that gives optimal collision resistance against adversaries with pre-computation.

1.3 Open Problems

1. An obvious open question is proving a tighter security bound on B-block m-way collision finding for $B < \log m$ or alternatively finding a better attack. We conjecture that the attack we have presented in this work for that parameter range is optimal.
2. We are aware of few works that have studied multi-collisions in sponge constructions without pre-processing. In the famous work [6], Bertoni et al. presented the sponge construction and briefly talked about realizing multi-collisions in sponges. Another work [4] from 2013 by AlAhmad, Alshaikhli and Nandi formally studied Joux's attack for sponges.

   As far as we are aware there is no work on security bounds of multi-collision resistance with pre-processing for Sponge constructions. As Sponge constructions use a permutation instead of a function, our lower bounds do not extend trivially to Sponge (however, our attacks do) and it is an interesting problem to consider for future work.

2 Preliminaries

2.1 Notation

For non-negative integers N, k, $[N] := \{1, \ldots, N\}$ and $\binom{[N]}{k}$ denotes the set of all k-sized subsets of $[N]$. For non-negative integers a, b such that $a < b$, (a, b) is the set of values between a and b excluding a and b themselves, i.e., $(a, b) := \{a+1, a+2, \ldots, b-1\}$. For a set X, $x \leftarrow_\$ X$ denotes x is a uniform random variable on X. X^+ denotes a list of one or more elements from X.

O, Ω and Θ are the standard asymptotic notations. The asymptotic notations with $\tilde{}$, e.g. $\tilde{O}$, ignore the poly-logarithmic terms.

2.2 Merkle Damgård Hash Function

A cryptographic hash function is a function that maps inputs of varied length to a fixed length output. Merkle Damgård (MD) is one of the popular classical construction for hash functions. It uses fixed length compression functions. MD5, SHA1, SHA2 and many more standard hashing algorithms are based on this construction.

In this work, we study an abstraction of the MD construction that takes a salt and an arbitrary length message as input, breaks the message into blocks of fixed length and iteratively applies the compression function, denoted h and modelled as a random oracle. For the remaining of the paper, unless specified otherwise, we will always think of h as a random function from $[N] \times [M] \to [N]$.

For any salt $a \in [N]$ and message $m \in [M]$, we define $MD_h(a, m) = h(a, m)$. For message $m \in [M]^+$ such that m can be written as $m = m_1||\dots||m_\ell$ where $m_1, \dots, m_\ell \in [M]$, we define MD_h on a and m as follows:

$$MD_h(a, m) := h(MD_h(a, m_1||\dots||m_{\ell-1}), m_\ell).$$

We refer to $m_1, \dots, m_\ell$ as blocks.

2.3 Definitions

Next, we formally define the m-way multi-collision resistance game for MD hash functions in the auxiliary input (AI) RO model.

▶ **Definition 1** ((S, T, m)-AI adversary)**.** *A pair of algorithms $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is an (S, T, m)-AI adversary against m-way collision resistance in MD_h if*

- *$\mathcal{A}_1^h$ gets unbounded access to h (i.e., gets to make unbounded number of oracle queries to h) and outputs S bits of advice σ;*
- *$\mathcal{A}_2^h$ takes σ and a salt $a \in [N]$ as input, issues T queries to h and outputs $\mathsf{msg}_1, \dots, \mathsf{msg}_m$*

where h is a function in $[N] \times [M] \to [N]$.

The m-way multi-collision resistance security game, namely m-AICR, is defined next.

▶ **Definition 2.** *For a function h in $[N] \times [M] \to [N]$ and salt $a \in [N]$, the game* m-AICR *is defined in fig. 1.*

```
Game m-AICR$_{h,a}(\mathcal{A})$
    $\sigma \leftarrow \mathcal{A}_1^h$
    $\mathsf{msg}_1, \dots, \mathsf{msg}_m \leftarrow \mathcal{A}_2^h(\sigma, a)$
    If $\mathsf{msg}_i \neq \mathsf{msg}_j$ and $\mathrm{MD}_h(a, \mathsf{msg}_i) = \mathrm{MD}_h(a, \mathsf{msg}_j)$  $\forall i \neq j \in [m]$
        Then Return 1
    Else Return 0
```

Figure 1 Security game m-AICR$_{h,a}(\mathcal{A})$.

For an (S, T, m)-AI adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage of $\mathcal{A}$, denoted as $\mathsf{Adv}^{\text{m-AICR}}(\mathcal{A})$*, is defined as the probability of* m-AICR$_{h,a}(\mathcal{A})$ *returning* 1 *when h is a random function in $[N] \times [M] \to [N]$ and a is a random salt in $[N]$ drawn independently. We define the (S, T, m)-AI multi-collision resistance, denoted by* $\mathsf{Adv}^{\text{m-AICR}}(S, T, m)$*, as the maximum advantage taken over all (S, T, m)-AI adversaries in the game* m-AICR.

```
Game $\textsf{m-AICR}^B_{h,a}(\mathcal{A})$
    $\sigma \leftarrow \mathcal{A}_1^h$
    $\textsf{msg}_1, \ldots, \textsf{msg}_m \leftarrow \mathcal{A}_2^h(\sigma, a)$
    If any of $\textsf{msg}_1, \ldots, \textsf{msg}_m$ have more than $B$ blocks
        Then Return 0
    If $\textsf{msg}_i \neq \textsf{msg}_j$ and $\mathrm{MD}_h(a, \textsf{msg}_i) = \mathrm{MD}_h(a, \textsf{msg}_j)$  $\forall i \neq j \in [m]$
        Then Return 1
    Else Return 0
```

Figure 2 Security game $\textsf{m-AICR}^B_{h,a}(\mathcal{A})$.

▶ **Definition 3.** *For a function h in $[N] \times [M] \to [N]$ and salt $a \in [N]$, the game $\textsf{m-AICR}^B$ is defined in fig. 2.*

For an (S, T, m)-AI adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage of $\mathcal{A}$ in the game $\textsf{m-AICR}^B$, denoted as $\textsf{Adv}_{\textsf{B}}^{\text{m-AICR}}(\mathcal{A})$, is defined as the probability of $\textsf{m-AICR}^B_{h,a}(\mathcal{A})$ returning 1 when h is a random function in $[N] \times [M] \to [N]$ and a is a random salt in $[N]$ drawn independently. We define the (S, T, m)-AI B-length multi-collision resistance, denoted by $\textsf{Adv}_{\textsf{B}}^{\text{m-AICR}}(S, T, m)$, as the maximum advantage taken over all (S, T, m)-AI adversaries in the game $\textsf{m-AICR}^B$.

2.4 Relevant Results

Compression arguments

▶ **Lemma 4** ([15], restated in [16])**.** *Say $\textsf{Enc}$ is an encoding function in $\{0,1\}^{\ell_1} \to \{0,1\}^{\ell_2}$ and $\textsf{Dec}$ is a decoding function in $\{0,1\}^{\ell_2} \to \{0,1\}^{\ell_1}$ such that for any ℓ_1-bit x, $\textsf{Dec}(\textsf{Enc}(x)) = x$ with probability at least ϵ, then $\ell_2 \geq \ell_1 - \log(1/\epsilon)$ holds true.*

2-way collision results

▶ **Lemma 5** (from [12])**.** *For any S, T, N*

$$\textsf{Adv}^{\text{2-AICR}}(S, T, 2) = \tilde{O}\left(\frac{ST^2}{N}\right).$$

▶ **Lemma 6** (from [3])**.** *For any S, T, N and $2 < B < T$*

$$\textsf{Adv}_B^{\text{2-AICR}}(S, T, 2) = \tilde{O}\left(\frac{STB}{N} \cdot \max\left\{1, \frac{ST^2}{N}\right\}\right).$$

3 Unbounded Length Multi-Collisions

▶ **Theorem 7.** *For any S, T, m, N and M such that $m = O(N^c)$ for some constant c,*

$$\textsf{Adv}^{\text{m-AICR}}(S, T, m) = \tilde{\Theta}\left(\frac{ST^2}{N}\right).$$

Proof. Note that the security bound is unsurprising and follows from lemma 5 via trivial reduction. We present the reduction in the full version of the paper for completeness.

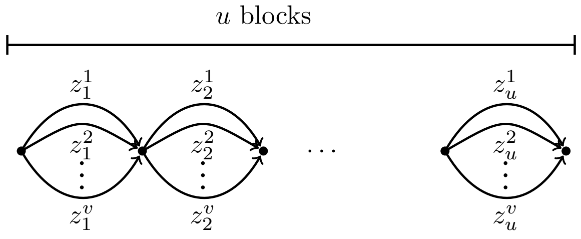


Figure 3 u-length chain of v-way collisions.

3.1 Matching Attack

Next we present the matching attack achieving the bound in the theorem. This multi-collision finding attack is based on the attack of Joux in [25] and the pre-computing attacks given in [23, 12, 1]. We assume $M > N$ and thus 2-way collisions exist for every salt in $[N]$ under h (that is because even if $M = N + 1$ there should exist a 2-way collision on every salt by pigeonhole principle). Note that it is possible to get rid of this assumption by replacing h with h^c for some c such that $M^c > N$. Then our (S, T, m)- AI adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is as follows:

1. In *Offline stage*, $\mathcal{A}_1$ picks $s = S/(3 \log m \cdot \log M)$ random salts, denoted $\{a_1, \dots, a_s\}$ and iteratively computes $a_j^i = h(a_j^{i-1}, 0)$ for $i \in [T/2]$ and $j \in [s]$ to obtain, say $\{a_1', \dots, a_s'\}$ set of salts. Then for each of the salt a_i' in this set, $\mathcal{A}_1$ finds a $(\log m)$-length chain of 2-way collisions (refer fig. 3 for a pictorial depiction of u-length chain of v-way collisions) and store these along with a_i' as advice.
2. In *Online stage*, $\mathcal{A}_2$ takes the advice output by $\mathcal{A}_1$ and random salt a and iteratively queries $h(*, 0)$ up to T times. If output of any of these queries is some $a' \in \{a_1', \dots, a_s'\}$, then $\mathcal{A}_2$ succeeds in outputting an m-way collision.

Let $\mathcal{G}$ be the subset of salts that are output of some $h(*, 0)$ query made in step 1 (Offline stage). We can show that $\mathbb{E}[|\mathcal{G}|] = \tilde{\Omega}(ST)$ in exactly the same manner as in lemma 14 of [12].

Then,

$$\mathsf{Adv}^{\text{m-AICR}}(\mathcal{A}) = \tilde{\Omega}\left(\frac{ST^2}{N}\right).$$

using that the adversary wins if output of any of its first $T/2$ queries in the online stage is in $\mathcal{G}$ and $\mathbb{E}[|\mathcal{G}|] = \tilde{\Omega}(ST)$. ◀

4 B-Block Multi-Collisions

▶ **Theorem 8.** *For any S, T, m, B, N and Msuch that $m = O(N^c)$ for some constant c and $B > 1$,*

$$\mathsf{Adv}_\mathsf{B}^{\text{m-AICR}}(S, T, m) = \tilde{\Omega}\left(\frac{ST}{m^{1/(B-1)}N}\right).$$

when $M > (m^{1/(B-1)} - 1) \cdot N$ and

$$\mathsf{Adv}_\mathsf{B}^{\text{m-AICR}}(S, T, m) = \tilde{\Omega}\left(\frac{STB}{N}\right)$$

when $B \geq (d+1) \log m$ and $M^d > N$ for the some $d \geq 1$.

Proof. We give an attack that achieves the bound in the theorem. In the attack and its analysis we assume $d = 1$ for simplicity. However, it is straightforward to extend it for other d.

We also assume $m = O(N^c)$ for some constant c. Then our proposed (S, T, m)-AI adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is a variation of our attack proposed in the proof of theorem 7. Refer to Appendix A for more details. ◀

▶ **Theorem 9.** *For any S, T, m, N and M*

$$\mathsf{Adv}_{\mathsf{B}}^{\text{m-AICR}}(S, T, m) = \tilde{O}\left(\max\left\{1, \frac{ST^2}{N}\right\} \cdot \frac{STB}{N} + \frac{T^2}{N}\right).$$

Proof. The proof of the theorem follows from the reduction of 2-way collision finding to m-way collision finding and lemma 6. ◀

Note that there is a gap between the best known attack and the bounds proved in Thm. 9 for several parameter ranges. In the following sections, we prove optimal security bounds for some of these parameter ranges, specifically $B = 1$ and $B = 2$ block m-way collisions.

5 1-Block Multi-Collisions

▶ **Theorem 10.** *For any S, T, m, N, M and $B = 1$,*

$$\mathsf{Adv}_{\mathsf{B}}^{\text{m-AICR}}(S, T, m) = \tilde{\Theta}\left(\frac{S}{mN} + \frac{(T/m)^m}{N^{m-1}}\right).$$

We refer the readers to Appendix B for the proof.

6 2-Block Multi-Collisions

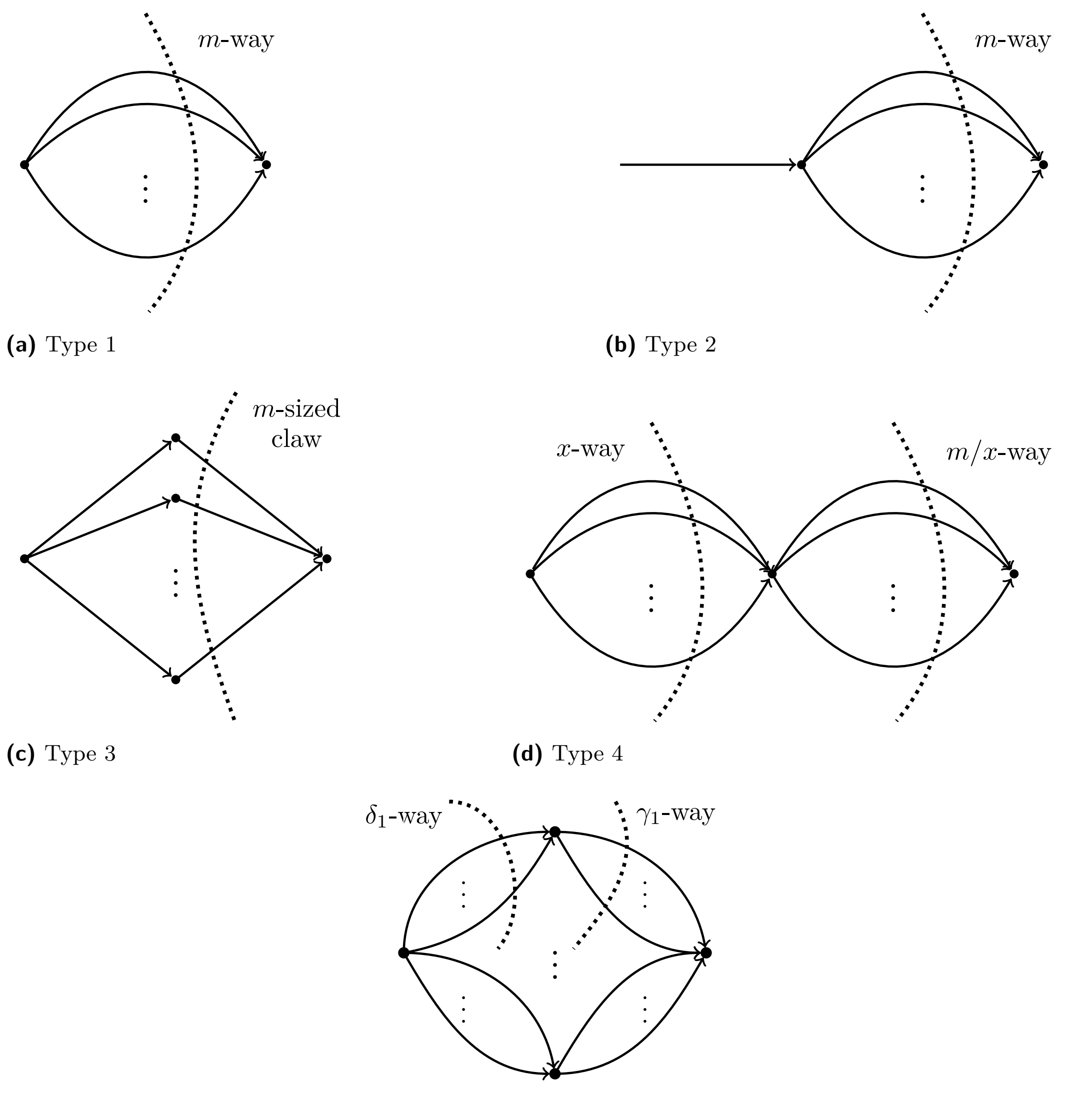
▶ **Theorem 11.** *For any S, T, m, N, M and $B = 2$ such that $m \le T$,*

$$\mathsf{Adv}_{\mathsf{B}}^{\text{m-AICR}}(S, T, m) = \tilde{O}\left(\frac{ST}{mN} + \frac{(eT/m)^m}{N^{m-1}}\right).$$

Proof. Let's fix an (S, T, m)-AI adversary $\mathcal{A}$. Next, we identify all the types of 2-block m-way collisions. See fig. 4 for a pictorial depiction of the types. Note that collision type 5 in fig. 4e can be thought of as a generalization of collision types in fig. 4cand fig. 4d but we treat them separately for simplicity.

Moreover it is possible that an adversary finds an m-way collision which is a mix of two types of collisions given in fig. 4. For some ℓ in $(0, m)$, consider an m-way collision on salt a that comprises of $\mathsf{msg}_1, \ldots, \mathsf{msg}_\ell, (\mathsf{msg}^0_{\ell+1}, \mathsf{msg}^1_{\ell+1}), \ldots,$
$(\mathsf{msg}^0_m, \mathsf{msg}^1_m)$ where for every $i \in [\ell]$, $h(a, \mathsf{msg}_i) = a'$ and for every $i \in [m] \setminus [\ell]$, $h(h(a, \mathsf{msg}^0_i), \mathsf{msg}^1_i) = a'$. Such an m-way collision can be thought of as ℓ-way collision of type 1 and $(m - \ell)$-way collision of type 3. Note that any m-way collision which is a mix of two types of collisions contains an m'-way pure collision (i.e., collision of a unique type) such that $m' \ge m/2$. Thus, we can reduce bounding the security of such a (mixed) m-way collision-finding adversary to bounding security of an $(m/2)$-way (pure) collision finding adversary (adding a multiplicative factor of 2 to our bound for pure collision-finding).

▷ Claim 12. Any m-way collision contains an $(m/2)$-way collision of exactly one of the types given in fig. 4.



(a) Type 1 **(b)** Type 2 **(c)** Type 3 **(d)** Type 4 **(e)** Type 5

Figure 4 Depiction of types of 2-block m-way collisions using function graph of h. The vertices denote the salt (in $[N]$) and the arrows correspond to a value in $[M]$.

Proof. For an m-way 2-block collision on an arbitrary salt a, the colliding messages can be denoted by $(\mathsf{msg}_i^0, \mathsf{msg}_i^1)_{i=1}^m$ such that $\mathsf{msg}_i^0 \in [M]$ and $\mathsf{msg}_i^1 \in [M] \cup \{\perp\}$ where $\mathsf{msg}_i^1 = \perp$ denotes that the colliding message is just 1-block, msg_i^0. Let's denote the output salt on which the messages collide by a', i.e., $h(h(a, \mathsf{msg}_i^0), \mathsf{msg}_i^1) = a'$ for all $i \in [m]$.

One possibility is that for all $i \in [m]$, $\mathsf{msg}_i^1 = \perp$. Then $\mathsf{msg}_1^0, \dots, \mathsf{msg}_m^0$ will have to be distinct and satisfy $h(a, \mathsf{msg}_i^0) = a'$ for every i. This is type 1 collision shown in fig. 4a.

Other possibility is that $\mathsf{msg}_i^1 \neq \perp$ for all $i \in [m]$. Then exactly one of the following will hold:

- $\mathsf{msg}_1^0 = \cdots = \mathsf{msg}_m^0$, then $\mathsf{msg}_1^1, \dots, \mathsf{msg}_m^1$ will have to be distinct and the messages will form type 2 collision shown in fig. 4b.
- $\mathsf{msg}_1^0, \dots, \mathsf{msg}_m^0$ are distinct and $h(a, \mathsf{msg}_1^0), \dots, h(a, \mathsf{msg}_m^0)$ are distinct, then the messages will form type 3 collision shown in fig. 4c.

- $\mathsf{msg}_1^0, \ldots, \mathsf{msg}_m^0$ are neither all equal nor all distinct but $h(a, \mathsf{msg}_1^0) = \cdots = h(a, \mathsf{msg}_m^0)$, then the colliding messages will form collision of types 4 shown in fig. 4d.
- $\mathsf{msg}_1^0, \ldots, \mathsf{msg}_m^0$ are not all equal (may or may not be all distinct) and $h(a, \mathsf{msg})$ for $m \in \{\mathsf{msg}_1^0, \ldots, \mathsf{msg}_m^0\}$ are neither all equal nor all distinct, then the colliding messages will form collision of types 5 shown in fig. 4e.

Finally, consider the possibility that $\mathsf{msg}_i^1 \neq \perp$ for some (not all) $i \in [m]$. Then the messages with $\mathsf{msg}_i^1 = \perp$ will be forming type 1 m'-way collision (where m' is the number of messages with $\mathsf{msg}_i^1 = \perp$) and the remaining messages will satisfy one of the cases listed above for $(m - m')$-way collision. Then there will exist $m/2$ colliding messages that satisfy one of the types of collisions depicted in fig. 4. $\triangleleft$

Let E_t be the event that $\mathcal{A}$ wins game $\mathsf{m\text{-}AICR}^2$ by outputting $(m/2)$-way collision of type t in fig. 4. Then

$$\mathsf{Adv}_2^{\text{m-AICR}}(\mathcal{A}) \leq \sum_{t=1}^{5} \Pr\left[\mathsf{m\text{-}AICR}_{h,a}^2(\mathcal{A}) = 1 \wedge E_t\right]$$

Next, we formally define multi-collision resistance game for MD hash functions in the Multi-instance model. We first define the adversary in the model.

▶ **Definition 13** ((S, T, m)-MI adversary). *An algorithm $\mathcal{A}$ is an (S, T, m)-MI adversary against m-way collision resistance in MD_h if*
- *$\mathcal{A}^h$ receives S salts from $[N]$ as input*
- *$\mathcal{A}^h$ gets to make ST queries to h and outputs $(\mathsf{msg}_1^i, \ldots, \mathsf{msg}_m^i)_{i=1}^S$*

where h is a function in $[N] \times [M] \to [N]$.

▶ **Definition 14.** *For a function h in $[N] \times [M] \to [N]$, fixed function S, and any $t \in [5]$, the game $\mathsf{m\text{-}MICR}^{S,t}$ is defined in fig. 5.*

```
Game $\mathsf{m\text{-}MICR}_h^{S,t}(\mathcal{A})$
    Sample $\{a_1, \ldots, a_S\} \leftarrow_\$ \left[\binom{N}{S}\right]$
    $(\mathsf{msg}_1^i, \ldots, \mathsf{msg}_m^i)_{i=1}^S \leftarrow \mathcal{A}^h(\{a_1, \ldots, a_S\})$
    If for any $i \in [S]$ and any of $\mathsf{msg}_1^i, \ldots, \mathsf{msg}_m^i$ have more than 2 blocks
        Then Return 0
    If $\forall i \in [S]$: $(\mathsf{msg}_1^i, \ldots, \mathsf{msg}_m^i)$ is $m$-way collision of type $t$ on $a_i$
        Then Return 1
    Else Return 0
```

Figure 5 Security game $\mathsf{m\text{-}MICR}_h^{S,t}(\mathcal{A})$.

For an (S, T, m)-MI adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in the game $\mathsf{m\text{-}MICR}^{S,t}$, denoted as $\mathsf{Adv}^{(\text{m,t})\text{-MICR}}(\mathcal{A})$, is defined as the probability of $\mathsf{m\text{-}MICR}_h^{S,t}(\mathcal{A})$ returning 1 when h is a random function in $[N] \times [M] \to [N]$. We define the (S, T, m)-MI 2-block multi-collision resistance, denoted by $\mathsf{Adv}^{(\text{m,t})\text{-MICR}}(S, T, m)$, as the maximum advantage taken over all (S, T, m)-MI adversaries.

Next, we define a modified version of the $\mathsf{m\text{-}MICR}$ game, which we refer to as $\mathsf{m\text{-}mod\text{-}MICR}$ game. It is different from $\mathsf{m\text{-}MICR}$ game only in the way the challenge salts are sampled.

▶ **Definition 15.** *For a function h in $[N] \times [M] \to [N]$, fixed function S, and any $t \in [5]$, the game $\mathsf{m\text{-}mod\text{-}MICR}^{S,t}$ is defined in fig. 6.*

The advantages in the game $\mathsf{m\text{-}mod\text{-}MICR}$ is defined similarly to that in the game $\mathsf{m\text{-}MICR}$.

```
Game $\mathsf{m\text{-}mod\text{-}MICR}_h^{S,t}(\mathcal{A})$
    Sample $a_i \leftarrow_\$ [N]$ for $i \in [S]$
    $(\mathsf{msg}_1^i, \ldots, \mathsf{msg}_m^i)_{i=1}^S \leftarrow \mathcal{A}^h(\{a_1, \ldots, a_S\})$
    If for any $i \in [S]$ and any of $\mathsf{msg}_1^i, \ldots, \mathsf{msg}_m^i$ have more than 2 blocks
        Then Return 0
    If $\forall i \in [S]$: $(\mathsf{msg}_1^i, \ldots, \mathsf{msg}_m^i)$ is $m$-way collision of type $t$ on $a_i$
        Then Return 1
    Else Return 0
```

Figure 6 Security game $\mathsf{m\text{-}mod\text{-}MICR}_h^{S,t}(\mathcal{A})$.

Next, we present a lemma that establishes a relation between the advantage in the games m-MICR and m-mod-MICR for a MI adversary.

▶ **Lemma 16.** *For any S, T, u, m, N, any $t \in [5]$ and δ such that $\delta \geq S/N$, if $\mathsf{Adv}^{(\mathrm{m,t})\text{-}\mathrm{MICR}}(u, T) \leq \delta^u$ for every $u \leq S$ then $\mathsf{Adv}^{(\mathrm{m,t})\text{-}\mathrm{mod}\text{-}\mathrm{MICR}}(S, T) \leq (2\delta)^S$.*

Proof.

$$\begin{aligned}
&\mathsf{Adv}^{(\mathrm{m,t})\text{-}\mathrm{mod}\text{-}\mathrm{MICR}}(S, T) \\
&\leq \sum_{u=1}^{S} \Pr[u \text{ distinct salts among } S \text{ random salts}] * \mathsf{Adv}^{(\mathrm{m,t})\text{-}\mathrm{MICR}}(u, T) \\
&\leq \sum_{u=1}^{S} \binom{S}{u} \cdot \left(\frac{u}{N}\right)^{S-u} \cdot \delta^u \\
&\leq \left(\frac{S}{N} + \delta\right)^S \leq (2\delta)^S
\end{aligned}$$

where the third inequality follows from the binomial theorem and the last inequality uses that $\delta \geq S/N$. ◀

Now we present the theorem that allows bounding the security in the auxiliary input model by analyzing the multi-instance game. This theorem is a variation of Theorem 4.1 in [10] and Theorem 3 in [3].

▶ **Theorem 17.** *For any S, T, m, $t \in [5]$ and $\delta \in [0, 1]$, if $\mathsf{Adv}^{(\mathrm{m/2,t})\text{-}\mathrm{MICR}}(S, T, m/2) \leq \delta^S$, then $\Pr[\mathsf{m\text{-}AICR}_{h,a}^2(\mathcal{A}) = 1 \wedge E_t] \leq 4\delta$.*

Proof. It is given that $\mathsf{Adv}^{(\mathrm{m/2,t})\text{-}\mathrm{MICR}}(S, T, m/2) \leq \delta^S$. Using lemma 16 we know that $\mathsf{Adv}^{(\mathrm{m/2,t})\text{-}\mathrm{mod}\text{-}\mathrm{MICR}}(S, T, m/2) \leq (2\delta)^S$ when $ST/mN \geq S/N$ (in other words $m \leq T$).

Say for some $t \in [5]$ there exists an (S, T, m)-AI adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ such that $\Pr[\mathsf{m\text{-}AICR}_{h,a}^2(\mathcal{A}) = 1 \wedge E_t] > 4\delta$. Then we can give an (S, T, m)-MI adversary $\mathcal{A}'$ that uses $\mathcal{A}$ and achieves $\mathsf{Adv}^{(\mathrm{m/2,t})\text{-}\mathrm{mod}\text{-}\mathrm{MICR}}(\mathcal{A}') > (2\delta)^S$ which is a contradiction. We describe $\mathcal{A}'$ next.

1. $\mathcal{A}'$ randomly chooses S-bit advice, σ.
2. For each element a in the input set of salts $\{a_1, \ldots, a_S\}$, $\mathcal{A}'$ runs $\mathcal{A}_2(\sigma, a)$ to obtain m messages that collide under MD_h and outputs the $m/2$ messages that form $(m/2)$-way collision of type t.

Let $\delta_h := \Pr_a[\mathsf{m\text{-}AICR}^2_{h,a}(\mathcal{A}) = 1 \wedge E_t]$ for a fixed choice of h. Then we know $\mathbb{E}_h[\delta_h] = \Pr_{h,a}[\mathsf{m\text{-}AICR}^2_{h,a}(\mathcal{A})] > 4\delta$. Then

$$\begin{aligned}\mathsf{Adv}^{(\mathrm{m/2,t})\text{-}\mathrm{MICR}}(S,T,m/2) &= \Pr_{h,a_1,\dots,a_S}[(\mathsf{m/2})\text{-}\mathsf{MICR}^{S,t}_h(\mathcal{A}) = 1] \\ &= \mathbb{E}_h\left[\Pr_{a_1,\dots,a_S}[(\mathsf{m/2})\text{-}\mathsf{MICR}^{S,t}_h(\mathcal{A}) = 1]\right] \\ &= \mathbb{E}_h\left[\Pr_{a_1,\dots,a_S}[(\mathsf{m/2})\text{-}\mathsf{MICR}^{S,t}_h(\mathcal{A}) = 1|\sigma = \mathcal{A}^h_1] \cdot \Pr[\sigma = \mathcal{A}^h_1]\right] \\ &= \mathbb{E}_h\left[\delta_h^S \cdot \frac{1}{2^S}\right] \geq \frac{\mathbb{E}_h[\delta_h]^S}{2^S} > (2\delta)^S,\end{aligned}$$

where the second to last inequality is by Jensen's inequality. ◀

▶ **Lemma 18.** *For any S, T, N, m and $t \in [5]$ such that $m = \omega(\log^2 N)$ and $m \leq ST$,*

$$\mathsf{Adv}^{(\mathrm{m,t})\text{-}\mathrm{MICR}}(S,T,m) = \left(\tilde{O}\left(\frac{ST}{mN} + \frac{(eT/m)^m}{N^{m-1}}\right)\right)^S.$$

It is worth noting that in the lemma we make an additional assumption that $m = \omega(\log^2 N)$ which does not contradict with the statement of Thm 11. And that is because for $m = O(\log^c N)$, the theorem holds trivially from the reduction of 2-way collision finding to m-way collision finding and the result of [1].

Now proof of theorem 11 follows from theorem 17 and lemma 18 as follows:

$$\begin{aligned}&\forall t \in [5] : \Pr[\mathsf{m\text{-}AICR}^2_{h,a}(\mathcal{A}) = 1 \wedge E_t] \leq 4 \cdot \left(\mathsf{Adv}^{(\mathrm{m,t})\text{-}\mathrm{MICR}}(S,T,m)\right)^{1/S} \\ \implies & \mathsf{Adv}_2^{\mathrm{m\text{-}AICR}}(\mathcal{A}) \leq \sum_{t=1}^{5} \Pr[\mathsf{m\text{-}AICR}^2_{h,a}(\mathcal{A}) = 1 \wedge E_t] \\ &\leq 4 \cdot \sum_{t=1}^{5} (\mathsf{Adv}^{(\mathrm{m,t})\text{-}\mathrm{MICR}}(S,T,m))^{1/S} \\ &= \tilde{O}\left(\frac{ST}{mN} + \frac{(eT/m)^m}{N^{m-1}}\right)\end{aligned}$$

To complete the proof of theorem 11, it only remains to prove lemma 18.

Proof. Let's fix a (S,T,m)-MI adversary $\mathcal{A}$ and let h be a random oracle. We denote the input of random S-sized set of salts by $\{a_1, \dots, a_S\}$. Let $\mathbf{X}^t_i$ be the indicator variable that $\mathcal{A}$ wins on salt a_i, i.e., $\mathcal{A}$ finds m-way collision of type t on salt a_i for $t \in [5]$.

To prove the lemma , we need to show for each $t \in [5]$, that $\mathsf{Adv}^{(\mathrm{m,t})\text{-}\mathrm{MICR}}(S,T,m) = \left(\tilde{O}\left(\frac{ST}{mN} + \frac{(eT/m)^m}{N^{m-1}}\right)\right)^S$.

Type 1 collisions are intuitively the easiest to analyze. Consider the adversary gets S distinct salts, say $\{a_1, \dots, a_S\}$, as input and in order to win on a_i for any $i \in [S]$, $\mathcal{A}$ needs to make m queries of the form $(a_i, *)$ such that all their outputs are equal under h. If $\mathcal{A}$ wins on "too many" S-sized subsets of $[N]$, we give an encoder that could use this $\mathcal{A}$ to compress h but that is impossible. Hence, there exists no such $\mathcal{A}$.

The encoding algorithm will be as follows:

- Store the Sm queries among the ST queries that are involved in collisions for the S challenge salts in an unordered set, say W. This would require $\log \binom{ST}{Sm}$ bits.
- For all $i \in [S]$, delete the output of all the queries of the form $(a_i, *)$ except the first occurring query in the unordered set W from the table of h. This saves $S(m-1)\log N$ bits.

Say $\mathsf{Adv}^{(\mathrm{m,1})\text{-}\mathrm{MICR}}(\mathcal{A}) = \epsilon$. Then

$$\log \epsilon \le \log \binom{ST}{Sm} - S(m-1)\log N$$
$$\implies \epsilon \le \frac{\binom{ST}{Sm}}{N^{S(m-1)}} \le \left[\frac{(eT/m)^m}{N^{m-1}}\right]^S .$$

Analysis for collisions of type 2-5 are more evolved. We refer the readers to Appendix C for a careful analysis of type 2 collisions. Analysis for remaining collision types are based on similar ideas. However, due to lack of space we omit the complete analysis here. We refer the readers to the full version of the paper. ◀

◀

References

1 Akshima, David Cash, Andrew Drucker, and Hoeteck Wee. Time-Space Tradeoffs and Short Collisions in Merkle-Damgård Hash Functions. In *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 157–186. Springer, 2020.

2 Akshima, Xiaoqi Duan, Siyao Guo, and Qipeng Liu. On Time-Space Lower Bounds for Finding Short Collisions in Sponge Hash Functions. In *Theory of Cryptography - 21st International Conference, TCC 2023, Taipei, Taiwan, November 29 - December 2, 2023, Proceedings, Part III*, volume 14371 of *Lecture Notes in Computer Science*, pages 237–270. Springer, 2023. doi:10.1007/978-3-031-48621-0_9.

3 Akshima, Siyao Guo, and Qipeng Liu. Time-Space Lower Bounds for Finding Collisions in Merkle-Damgård Hash Functions. In *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part III*, volume 13509 of *Lecture Notes in Computer Science*, pages 192–221. Springer, 2022. doi:10.1007/978-3-031-15982-4_7.

4 Mohammad A AlAhmad, Imad Fakhri Alshaikhli, and Mridul Nandi. Joux Multicollisions Attack in Sponge Construction. In *Proceedings of the 6th International Conference on Security of Information and Networks*, pages 292–296, 2013.

5 Itay Berman, Akshay Degwekar, Ron D Rothblum, and Prashant Nalini Vasudevan. Multi-Collision Resistant Hash Functions and their Applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 133–161. Springer, 2018.

6 Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge Functions. In *ECRYPT hash workshop*, volume 2007, 2007.

7 Nir Bitansky, Yael Tauman Kalai, and Omer Paneth. Multi-Collision Resistance: a Paradigm for Keyless Hash Functions. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 671–684, 2018.

8 Ernest Brickell, David Pointcheval, Serge Vaudenay, and Moti Yung. Design Validations for Discrete Logarithm Based Signature Schemes. In *International Workshop on Public Key Cryptography*, pages 276–292. Springer, 2000.

9 Dror Chawin, Iftach Haitner, and Noam Mazor. Lower Bounds on the Time/Memory Tradeoff of Function Inversion. In *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part III*, pages 305–334, 2020.

10 Kai-Min Chung, Siyao Guo, Qipeng Liu, and Luowen Qian. Tight Quantum Time-Space Tradeoffs for Function Inversion. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 673–684. IEEE, 2020.

11 Sandro Coretti, Yevgeniy Dodis, and Siyao Guo. Non-Uniform Bounds in the Random-Permutation, Ideal-Cipher, and Generic-Group Models. In *Annual International Cryptology Conference*, pages 693–721. Springer, 2018.

12 Sandro Coretti, Yevgeniy Dodis, Siyao Guo, and John Steinberger. Random Oracles and Non-Uniformity. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 227–258. Springer, 2018.

13 Henry Corrigan-Gibbs and Dmitry Kogan. The Discrete-Logarithm Problem with Preprocessing. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 415–447. Springer, 2018.

14 Henry Corrigan-Gibbs and Dmitry Kogan. The Function-Inversion Problem: Barriers and Opportunities. In *TCC*, 2019. Also, Crypto ePrint 2019/1046.

15 Anindya De, Luca Trevisan, and Madhur Tulsiani. Time Space Tradeoffs for Attacks against One-Way Functions and PRGs. In *Annual Cryptology Conference*, pages 649–665. Springer, 2010.

16 Yevgeniy Dodis, Siyao Guo, and Jonathan Katz. Fixing Cracks in the Concrete: Random Oracles with Auxiliary Input, Revisited. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 473–495. Springer, 2017.

17 Cody Freitag, Ashrujit Ghoshal, and Ilan Komargodski. Time-Space Tradeoffs for Sponge Hashing: Attacks and Limitations for Short Collisions. In *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part III*, volume 13509 of *Lecture Notes in Computer Science*, pages 131–160. Springer, 2022. `doi:10.1007/978-3-031-15982-4_5`.

18 Cody Freitag, Ashrujit Ghoshal, and Ilan Komargodski. Optimal Security for Keyed Hash Functions: Avoiding Time-Space Tradeoffs for Finding Collisions. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV*, volume 14007 of *Lecture Notes in Computer Science*, pages 440–469. Springer, 2023. `doi:10.1007/978-3-031-30634-1_15`.

19 Ashrujit Ghoshal and Ilan Komargodski. On Time-Space Tradeoffs for Bounded-Length Collisions in Merkle-Damgård Hashing. In *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part III*, volume 13509 of *Lecture Notes in Computer Science*, pages 161–191. Springer, 2022. `doi:10.1007/978-3-031-15982-4_6`.

20 Marc Girault and Jacques Stern. On the Length of Cryptographic Hash-Values Used in Identification Schemes. In *Annual International Cryptology Conference*, pages 202–215. Springer, 1994.

21 Alexander Golovnev, Siyao Guo, Spencer Peters, and Noah Stephens-Davidowitz. Revisiting Time-Space Tradeoffs for Function Inversion. 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 II*, volume 14082 of *Lecture Notes in Computer Science*, pages 453–481. Springer, 2023. `doi:10.1007/978-3-031-38545-2_15`.

22 Nick Gravin, Siyao Guo, Tsz Chiu Kwok, and Pinyan Lu. Concentration Bounds for Almost k-wise Independence with Applications to Non-Uniform Security. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2404–2423. SIAM, 2021.

23 M. Hellman. A Cryptanalytic Time-memory Trade-off. *IEEE Trans. Inf. Theor.*, 26(4):401–406, July 1980.

24 Russell Impagliazzo and Valentine Kabanets. Constructive Proofs of Concentration Bounds. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 13th International Workshop, APPROX 2010, and 14th International Workshop, RANDOM 2010, Barcelona, Spain, September 1-3, 2010. Proceedings*, pages 617–631, 2010.

25 Antoine Joux. Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In *Annual International Cryptology Conference*, pages 306–316. Springer, 2004.

26 Ilan Komargodski, Moni Naor, and Eylon Yogev. Collision Resistant Hashing for Paranoids: Dealing with Multiple Collisions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 162–194. Springer, 2018.

27 Qipeng Liu and Mark Zhandry. On Finding Quantum Multi-Collisions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 189–218. Springer, 2019.

28 Ronald L Rivest and Adi Shamir. PayWord and MicroMint: Two Simple Micropayment Schemes. In *International workshop on security protocols*, pages 69–87. Springer, 1996.

29 Ron D. Rothblum and Prashant Nalini Vasudevan. Collision-Resistance from Multi-Collision-Resistance. In *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part III*, volume 13509 of *Lecture Notes in Computer Science*, pages 503–529. Springer, 2022. doi:10.1007/978-3-031-15982-4_17.

30 Dominique Unruh. Random Oracles and Auxiliary Input. In *Annual International Cryptology Conference*, pages 205–223. Springer, 2007.

A Bounded Length Multi-Collision Attack

The attack is as follows:

1. In the *Offline* stage,
   a. $\mathcal{A}_1$ randomly picks s salts, denoted $\{a_1^0, \dots, a_s^0\}$. Let's define $x := \max\{0, \lfloor (B - \log_2 m)/2 \rfloor\}$ and $y := \min\{\log_2 m, B-1\}$. Then for every $i \in [s]$ and $j \in [x]$, $\mathcal{A}_1$ first iteratively computes $a_i^j = h(a_i^{j-1}, 0)$ to obtain the set $\{a_1^x, \dots, a_s^x\}$.
   b. Next, $\mathcal{A}_1$ finds a y-length chain of $m^{1/y}$-way collision (recall fig. 3) on a_i^x for every $i \in [s]$.
   c. Finally, $\mathcal{A}_1$ stores the tuple of a_i^x and the corresponding y-length chain for every $i \in [s]$ as advice.
2. In the *Online* stage, $\mathcal{A}_2$ is given the advice from $\mathcal{A}_1$ and a random salt, denoted a, as challenge. $\mathcal{A}_2$ runs the following steps for $k \in [\lfloor T/2x \rfloor]$ when x is non-zero and $k \in [T]$ when $x = 0$:
   - Set $b = h(a, k)$ and $\mathsf{counter} = 1$
   - While $a, b \notin \{a_1^x, \dots, a_s^x\}$ and $\mathsf{counter} < 2x$:
     - query and set $b = h(b, 0)$
     - increment $\mathsf{counter}$ by 1.
   - Say $b = a_\ell^x$ for some ℓ, then $\mathcal{A}_2$ can learn the y-length chain of $m^{1/y}$-way collision on a_ℓ^x from the advice and output m colliding messages consisting of k concatenated with $(\mathsf{counter} - 1)$ 0's followed by m different combinations in the chain.

The analysis of this attack can be found in the full version of the paper.

B Proof of Theorem 10

We first present the attack for finding 1-block m-way collisions.

Note that the following trivial adversary, say $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, achieves $\tilde{\Omega}\left(\frac{S}{mN} + \frac{(T/m)^m}{N^{m-1}}\right)$ advantage.

1. In the *Offline* stage, $\mathcal{A}_1$ stores (1-block) m-way collisions for $\tilde{\Omega}(S/m)$ salts as part of the advice.
2. In the *Online* stage, if the input random salt, say a, is one of the salts for which the advice contains a m-way collision, adversary returns the corresponding m-way collision. Else the adversary tries to find m-way collision using it's T queries to the oracle h.

Next we prove the security bound from the theorem. We prove it via "global" compression which is based on a result in [16].

Fix an (S, T, m)-AI adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ having advantage ϵ. Let σ denote the S-bit advice output by $\mathcal{A}_1$ and let $\mathcal{G}$ be the set of salts in $[N]$ on which $\mathcal{A}_2$ succeeds in finding m-way collision when given σ and makes T queries to h.

We want to show that the set $\mathcal{G}$ cannot be large for any (S, T, m)-AI adversary. In other words, it is impossible for any (S, T, m)-AI adversary to succeed in finding m-way collisions on "too" many salts, irrespective of what information is contained in the S-bit advice σ. The idea is to give an encoding algorithm that uses such an adversary that succeeds on a "lot" of salts to compress h, which should be impossible as h is a random function.

Note that since $\mathcal{A}_2$ finds m-way collisions on each of the salts in $\mathcal{G}$, for every $a \in \mathcal{G}$ there should exist m queries of the type $(a, *) \in [N] \times [M]$ such that outputs of all the m queries are same under h. Our encoding algorithm uses this repetition in outputs to compress as follows:

1. Store the advice σ (requires S-bits)
2. Store the size of the set $\mathcal{G}$ (in other words, the number of elements in $\mathcal{G}$), denoted $|\mathcal{G}|$ (requires $\log N$ bits)
3. Store the set $\mathcal{G}$ (requires $\log \binom{N}{|\mathcal{G}|}$ bits)
4. Store the unordered set of $|\mathcal{G}| \cdot m$ queries corresponding to the m-way collisions for each salt in $\mathcal{G}$. Let's denote this set by $\mathcal{X}$. Note that $\mathcal{X}$ is a subset of the $|\mathcal{G}| \cdot T$ queries made by $\mathcal{A}_2$ using the assumption that $\mathcal{A}_2$ has to query it's outputs. Thus, storing $\mathcal{X}$ requires $\log \binom{|\mathcal{G}|T}{|\mathcal{G}|m}$ bits.
5. Delete the outputs of $\mathcal{G}|(m-1)$ queries in $\mathcal{X}$ from table of h. (Note that the table for h is stored as follows: table contains output of h on the queries made by $\mathcal{A}_2$ on the set $\mathcal{G}$ followed by the output of h on the remaining entries of queries in $[N] \times [M]$ in lexicographic order.) This saves $|\mathcal{G}|(m-1) \cdot \log N$ bits.

Via the compression argument of De et al. [15] (stated as lemma 4 in this paper), the following holds:

$$S + \log N + \log \binom{N}{|\mathcal{G}|} + \log \binom{|\mathcal{G}|T}{|\mathcal{G}|m} + NM \log N - |\mathcal{G}|(m-1) \cdot \log N \geq NM \log N$$

$$\implies S + \log N + |\mathcal{G}| \log \left(\frac{N}{|\mathcal{G}|}\right) + |\mathcal{G}| m \log \left(\frac{|\mathcal{G}|T}{|\mathcal{G}|m}\right) \geq |\mathcal{G}|(m-1) \log N$$

$$\implies S + \log N \geq |\mathcal{G}| \log \left(\frac{N^{m-1}|\mathcal{G}|}{N(T/m)^m}\right)$$

We take expectation on both sides of the above equation and use convexity of the function $x \log x$ along with $\mathbb{E}[|\mathcal{G}|] = \epsilon N$ as follows:

$$\mathbb{E}[S + \log N] \geq \mathbb{E}\left[|\mathcal{G}| \log\left(\frac{N^{m-1}|\mathcal{G}|}{N(T/m)^m}\right)\right]$$
$$\implies S + \log N \geq \mathbb{E}[|\mathcal{G}|] \log\left(\frac{N^{m-1}\mathbb{E}[|\mathcal{G}|]}{N(T/m)^m}\right)$$
$$\geq \epsilon N \cdot \log\left(\frac{N^{m-1} \cdot \epsilon N}{N(T/m)^m}\right) \geq \epsilon N \cdot \log\left(\frac{N^{m-1} \cdot \epsilon}{(T/m)^m}\right)$$

Consider the following two cases:

1. If
$$\frac{N^{m-1}\epsilon}{(T/m)^m} \leq 2^m$$
This simply implies
$$\epsilon \leq \frac{(2T/m)^m}{N^{m-1}}$$
2. Otherwise
$$\frac{N^{m-1}\epsilon}{(T/m)^m} > 2^m$$
which implies
$$S + \log N \geq \epsilon N \log 2^m \implies \epsilon \leq \frac{S + \log N}{mN}$$
This completes the proof for the security bound.

C Bounding MI-security for Type 2 collisions

The analysis would be trivial and same as for $t = 1$ if m-way collision for each salt used distinct queries. However, queries can be reused among collisions for different salts. Refer to fig. 4b for a visual depiction.

Say x salts use the same m-way collision structure. Refer fig. 7 for a visual depiction. Say the adversary finds y such structures where $x = \delta_i$ for the i^{th} structure.

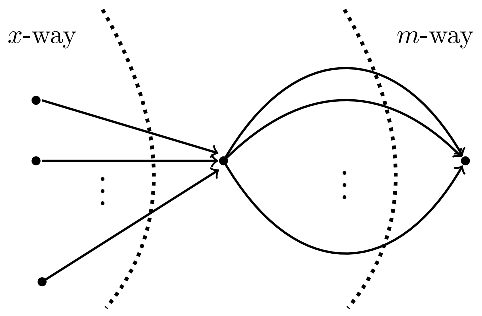


Figure 7 Depiction of x salts "sharing" a m-way 1-block collision.

Then it holds that

$$\delta_1 + \cdots + \delta_y \geq S$$

The encoding algorithm will be as follows:

```
1. Store $y$ which requires $\log S$ bits.
2. Store the subset of $ST$ queries forming $m$-way collisions in an unordered fashion. From
   the table of $h$, delete the entries corresponding to the queries in the set except the first
   query. There will be $y$ such $m$-sized sets.
3. Store the number of $i \in [y]$ such that $\delta_i \geq m$ which requires $\log S$-bits.
4. For $i \in [y]$: if $\delta_i > m$ do the following
   - Store $\delta_i$
   - Store the $\delta_i$-sized set of queries that have the same output. From the table of $h$,
     delete the entries corresponding to the queries in the set except the first query.
```

Say ϵ is the advantage. For decoding purposes, the encoder must store these sets of collisions but also store the values of $y, \{\delta_i | i \in [y], \delta_i > m\}$ and $|\{\delta_i | i \in [y], \delta_i > m\}|$. It holds by the compression argument that

$$\begin{aligned}\log \epsilon \leq \log S + y \log \binom{ST}{m} - y \cdot (m-1) \log N + \log S + \\ \sum_{i \in [y]: \delta_i \geq m} \left[\log S + \log \binom{ST}{\delta_i} - (\delta_i - 1) \log N \right] \\ = \log S + \sum_{i \in [y]: \delta_i < m} \left[\log \binom{ST}{m} - (m-1) \log N \right] + \log S + \\ \sum_{i \in [y]: \delta_i \geq m} \left[\log \binom{ST}{m} - (m-1) \log N + \log S + \log \binom{ST}{\delta_i} - (\delta_i - 1) \log N \right]\end{aligned}$$

In the analysis, we consider the following cases:

1. $x \geq m$
   It would suffice to take into account the probability of x of the ST queries having the same output, which is:

$$\frac{\binom{ST}{x}}{N^{x-1}} \leq \left(\frac{eST}{xN}\right)^x \cdot N \leq \left(\frac{2eST}{xN}\right)^x \cdot \frac{1}{2^x} \cdot N \leq \left(\frac{2eST}{mN}\right)^x$$

   where the last inequality uses that $\frac{1}{2^x} \leq \frac{1}{2^m} \leq N$ as $x \geq m \geq \log N$.
2. $x < m$
   Then the probability of an m-way collision is:

$$\leq \frac{\binom{ST}{m}}{N^{m-1}} \leq \left(\frac{eST}{mN}\right)^m \cdot N \leq \left(\frac{2eST}{mN}\right)^m \cdot \frac{1}{2^m} \cdot N \leq \left(\frac{2eST}{mN}\right)^x$$

   where the last inequality holds for $m \geq \log N$ and $2eST/mN \leq 1$.

Hence, the probability of finding the structure where x salts share a m-way 1-block collision such that we get type 2 collision on the x salts in the MI setting is at most $(2eST/mN)^x$ irrespective of whether $x < m$ or $x \geq m$. This shows that in either case, we can get the desired bound exponentially small in x.

Then, we know for each i such that $\delta_i < m$,

$$\log \binom{ST}{m} - (m-1) \log N \leq \log \left(\frac{2eST}{mN}\right)^{\delta_i}$$

and for each i such that $\delta_i \geq m$

$$\begin{aligned}
&\log \binom{ST}{m} - (m-1)\log N + \log S + \log \binom{ST}{\delta_i} - (\delta_i - 1)\log N \\
&\leq \log S + \log \binom{ST}{\delta_i} - (\delta_i - 1)\log N \leq \log \left[S \cdot \left(\frac{2eST}{mN} \right)^{\delta_i} \right]
\end{aligned}$$

using $2eST \leq mN$. Then,

$$\begin{aligned}
\implies \epsilon &\leq S^2 \cdot \left[\prod_{i\in[y]:\delta_i<m} \left(\frac{2eST}{mN} \right)^{\delta_i} \right] \cdot \left[\prod_{i\in[y]:\delta_i\geq m} S \cdot \left(\frac{2eST}{mN} \right)^{\delta_i} \right] \\
&\leq S^2 \cdot \left[\prod_{i\in[y]:\delta_i<m} \left(\frac{2eST}{mN} \right)^{\delta_i} \right] \cdot \left[\prod_{i\in[y]:\delta_i\geq m} S \cdot \frac{1}{2^m} \cdot \left(\frac{4eST}{mN} \right)^{\delta_i} \right] \\
&\leq 2^S \cdot \prod_{i\in[y]} \left(\frac{4eST}{mN} \right)^{\delta_i} = 2^S \cdot \left(\frac{4eST}{mN} \right)^{\delta_1+\cdots+\delta_y} \leq 2^S \cdot \left(\frac{4eST}{mN} \right)^{S} \\
&= \left(\frac{8eST}{mN} \right)^{S}
\end{aligned}$$

where the last inequality uses that $\delta_1 + \delta_2 + \delta_y >= S$ as these structures should give collisions on S distinct salts.

Secure Multiparty Computation of Symmetric Functions with Polylogarithmic Bottleneck Complexity and Correlated Randomness

Reo Eriguchi
National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

Abstract

Bottleneck complexity is an efficiency measure of secure multiparty computation (MPC) protocols introduced to achieve load-balancing in large-scale networks, which is defined as the maximum communication complexity required by any one player within the protocol execution. Towards the goal of achieving low bottleneck complexity, prior works proposed MPC protocols for computing symmetric functions in the correlated randomness model, where players are given input-independent correlated randomness in advance. However, the previous protocols with polylogarithmic bottleneck complexity in the number n of players require a large amount of correlated randomness that is linear in n, which limits the per-party efficiency as receiving and storing correlated randomness are the bottleneck for efficiency. In this work, we present for the first time MPC protocols for symmetric functions such that bottleneck complexity and the amount of correlated randomness are both polylogarithmic in n, assuming semi-honest adversaries colluding with at most $n - o(n)$ players. Furthermore, one of our protocols is even computationally efficient in that each player performs only polylog(n) arithmetic operations while the computational complexity of the previous protocols is $O(n)$. Technically, our efficiency improvements come from novel protocols based on ramp secret sharing to realize basic functionalities with low bottleneck complexity, which we believe may be of interest beyond their applications to secure computation of symmetric functions.

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

Keywords and phrases Secure multiparty computation, Bottleneck complexity, Secret sharing

Digital Object Identifier 10.4230/LIPIcs.ITC.2024.10

Related Version *Full Version*: `https://eprint.iacr.org/2024/1152`

Funding *Reo Eriguchi*: This work was supported in part by JST CREST Grant Number MJCR22M1 and JST AIP Acceleration Research JPMJCR22U5.

Acknowledgements We thank Keitaro Hiwatashi for his helpful discussions and suggestions.

1 Introduction

Secure multiparty computation (MPC) [48] is a fundamental cryptographic primitive which enables n players to jointly compute a function $f(x_1, \ldots, x_n)$ without revealing information on their private inputs x_i to adversaries corrupting at most t players. Due to many important applications, the asymptotic and concrete optimization of MPC protocols has been the subject of a large body of research. In this work, we consider the *dishonest-majority* setting, where the majority of players are corrupted, i.e., $t > n/2$.

MPC in the correlated randomness model. A popular approach to designing MPC protocols in the dishonest-majority setting is to employ *correlated randomness*. In this model, players receive correlated randomness from a trusted dealer before inputs are known (the offline phase) and then consume the randomness to perform input-dependent computation (the online phase). It was shown in [1] that the correlated randomness allows us to construct information-theoretically secure protocols in the dishonest-majority setting, while such protocols do not exist in the plain model. Subsequently, many optimizations have been proposed and several of them are even implemented [6, 19, 37, 18, 8, 9]. Two primary efficiency metrics for MPC in this model are the *online communication* cost and the amount of *correlated randomness* received from a trusted dealer [13, 9]. This is because as opposed to local computation, communication and storage costs are usually dominant in MPC protocols and minimizing both costs simultaneously leads to fast and scalable protocols.

Bottleneck complexity. Traditionally, the cost of online communication has been measured by the *total* amount of communication across all n players. On the other hand, for practical applications such as peer-to-peer computations between lightweight devices, the *per-party* cost is a more effective measure than the total cost. For example, several existing protocols (e.g., [17, 14, 29, 22]) require one player to communicate different messages with every other player. Then, while the total communication cost is possibly scalable, the player must bear communication proportional to n and his cost quickly becomes prohibitive in large-scale MPC involving many players. In this work, we focus on a more fine-grained efficiency measure capturing the load-balancing aspect of protocols, called *bottleneck complexity* [10], which is defined as the maximum communication required by any one player during the protocol execution.

To fit large-scale networks, we aim at designing MPC protocols whose bottleneck complexity scales polylogarithmically with n. Unfortunately, there is a negative result that we cannot achieve sublinear bottleneck complexity for *all* functions even without any security considerations [10]. Due to this result, a line of works [43, 39, 21] have studied the problem of constructing protocols with low bottleneck complexity for specific classes of functions. Above all, the class of *symmetric functions*, whose values are the same no matter the order of n inputs, is one of the most fundamental functions including majority, counting, and parity functions. Recently, the authors of [39, 21] constructed information-theoretic protocols for symmetric functions with $O(\log n)$ bottleneck complexity[1]. However, a main drawback of the protocols is that every player needs to receive a large amount of correlated randomness that is linear in n per party. This means that no matter how much bottleneck complexity in the online phase is improved, the protocols do not work efficiently as receiving and storing correlated randomness is the bottleneck for efficiency. Motivated by the above considerations, in this work, we ask:

Can we construct MPC protocols for symmetric functions keeping <u>both</u> bottleneck complexity and the amount of correlated randomness polylogarithmic in n?

1.1 Our Results

In this work, we answer the above question affirmatively by presenting two different constructions of MPC protocols for symmetric functions, assuming semi-honest adversaries colluding with at most $n - o(n)$ players. There is a trade-off between bottleneck complexity and the amount of correlated randomness.

▶ **Theorem 1** (Informal). *For a parameter ℓ, there exists an information-theoretic MPC protocol for computing a symmetric function $f : \{0,1\}^n \to \{0,1\}$ that has bottleneck complexity $O(\log n)$, per-party correlated randomness of size $O(\ell \log n)$, and tolerates up to $n - \Theta(n/\ell)$ semi-honest corruptions.*

[1] The authors of [39] considered a related class of functions called *abelian programs*. Their protocol can also compute symmetric functions by setting the underlying abelian group as the ring of integers modulo $n + 1$. See Remark 6 for more details.

Table 1 Information-theoretic MPC protocols for computing symmetric functions with sublinear bottleneck complexity in the dishonest-majority setting.

Reference	BC	CR	Corruption
[39, 21]	$O(\log n)$	$O(n)$	$n-1$
[21]	$O(\sqrt{n})$	$O(\sqrt{n})$	$n-1$
Ours (Corollary 8)	$O(\log n)$	$O((\log n)^2)$	$n-o(n)$
Ours (Corollary 15)	$O((\log n)^2)$	$O(\log n)$	$n-o(n)$
Ours (Corollary 9)	$O(\log n)$	$O(\log n)$	$(1-\epsilon)n$

"BC" stands for bottleneck complexity and "CR" stands for the amount of correlated randomness per party. ϵ is any constant with $0<\epsilon<1/2$.

▶ **Theorem 2** (Informal). *For a parameter ℓ, there exists an information-theoretic MPC protocol for computing a symmetric function $f:\{0,1\}^n \to \{0,1\}$ that has bottleneck complexity $O(\ell \log n)$, per-party correlated randomness of size $O(\log n)$, and tolerates up to $n-\Theta(n/\ell)$ semi-honest corruptions.*

A typical choice of the parameter ℓ is $\ell=\Theta(\log n)$. Theorem 1 then gives a protocol that has bottleneck complexity $O(\log n)$ and correlated randomness of size $O((\log n)^2)$. Theorem 2 gives a protocol that has bottleneck complexity $O((\log n)^2)$ and correlated randomness of size $O(\log n)$. Compared to the previous works, our protocols achieve for the first time polylog(n) bottleneck complexity and correlated randomness simultaneously (see Table 1). Furthermore, if we set $\ell \approx 1/\epsilon$ for a constant $0<\epsilon<1/2$, then both Theorems 1 and 2 give protocols such that both the bottleneck complexity and the amount of correlated randomness are $O(\log n)$ for a constant fraction of corrupted players (e.g., 99 percent of the parties are corrupted). Although the corruption threshold t is lower than the maximum $n-1$, our protocols are still secure in the dishonest majority setting $t>n/2$. A more detailed comparison is shown in Table 1.

Our first protocol even achieves polylog(n) computational complexity since the local computation of each player involves only $O(\log n)$ arithmetic operations in a field of size $O(n)$. As a comparison, the previous protocols in [39, 21] have $O(n)$ computational complexity since every player needs to process vectors or matrices of size $O(n)$.

Technically, we achieve polylogarithmic bottleneck complexity and correlated randomness with the help of ramp secret sharing [42, 7, 47, 23] (also known as packed secret sharing), a technique to distribute and operate on multiple secrets simultaneously only paying the cost of a single secret. This tool was used to realize certain functionalities in several previous works [14, 22], but they required every player to distribute fresh shares of their local secret, which leads to inefficient protocols in terms of bottleneck complexity. Our technical novelty is carefully designing correlated randomness to avoid such resharing processes and keep polylog(n) bottleneck complexity. See Section 2 for a detailed overview of our techniques.

1.2 Related Work

Boyle et al. [10] constructed a generic compiler from any possibly insecure protocol to a computationally secure protocol (without correlated randomness) preserving bottleneck complexity up to a polynomial factor in a security parameter. However, their compiler is based on fully homomorphic encryption, which can only be instantiated from a narrow class of cryptographic assumptions [25, 46, 26], and the concrete efficiency leaves much to be desired. Orlandi, Ravi, and Scholl [43] constructed a protocol for symmetric functions in the correlated randomness model assuming garbled circuits. However, in addition to not achieving information-theoretic security, players need to receive a garbled circuit with $O(\log n)$ input bits as correlated randomness. Since the minimum size of circuits computing a worst-case function with m input bits is $\Omega(2^m/m)$ [38], the correlated randomness of [43] is $\Omega(\lambda n/\log n)$ in the worst case, which is not polylogarithmic in n. There are maliciously secure protocols with sublinear bottleneck complexity for general tasks [20] and specific tasks [41, 24]. However, these protocols assume the strong honest-majority setting ($t < n/3$) and/or only achieve $\Omega(\sqrt{n})$ bottleneck complexity.

There is a rich line of works studying total communication complexity of MPC, e.g., [27, 4, 11, 44, 15, 35, 34, 17, 2, 16, 19, 5, 36, 12, 30, 31, 40, 29]. However, protocols in all of the above works require *full interaction* among players, that is, each player may send different messages to all the other players in each round of interaction. This feature necessarily results in high bottleneck complexity $\Omega(n)$.

The authors of [33, 32] initiated the study of the communication complexity of MPC with restricted interaction patterns. Halevi et al. [32] studied a chain-based interaction, in which players interact over a simple directed path traversing all players. Protocols on a chain-based interaction possibly achieve low bottleneck complexity since each player communicates with at most two players. However, since the last player on the chain is allowed to evaluate the function on every possible input of his choice, the constructions in [32] cannot achieve the standard security of MPC, which requires that corrupted players learn nothing but the output.

2 Technical Overview

In this section, we provide an overview of our techniques. More detailed descriptions and security proofs will be given in the following sections.

2.1 Our First Protocol for Symmetric Functions

To begin with, we recall the protocol computing symmetric functions with $O(\log n)$ bottleneck complexity in [39, 21]. Let $h : \{0,1\}^n \to \{0,1\}$ be a symmetric function. Since the value of $h(x_1, \dots, x_n)$ depends only on the number of 1's, which is equal to the sum $\sum_{i\in[n]} x_i$, there is the unique function $f : \{0,1,\dots,n\} \to \{0,1\}$ such that $h(x_1,\dots,x_n) = f(\sum_{i\in[n]} x_i)$. Roughly speaking, the protocol in [39, 21] proceeds as follows: In the setup, players receive an additive sharing of the truth-table $\mathbf{T}_r \in \{0,1\}^{n+1}$ of f permuted with a random shift $r \in \{0,1,\dots,n\}$. Simultaneously, they also receive an additive sharing $(r_i)_{i\in[n]}$ of the shift r. In the online phase, players compute $x_i + r_i$, open $y = \sum_{i\in[n]} x_i + r$, and then open the y-th component of the permuted truth-table $\mathbf{T}_r$, which is $f(y - r) = h(x_1,\dots,x_n)$. In this protocol, however, players need to receive additive shares of the $(n+1)$-dimensional vector $\mathbf{T}_r$, which results in correlated randomness of size $O(n)$ per party.

Our starting point to reduce this large correlated randomness is using a *ramp secret sharing scheme* to share the permuted truth-table $\mathbf{T}_r$ of f. Ramp secret sharing [42, 7, 47] is a variant of secret sharing which can share a secret vector of dimension k keeping the share size logarithmic in k and n. One may expect that a ramp secret sharing scheme can compress the $(n+1)$-dimensional vector $\mathbf{T}_r$ into shares each of size logarithmic in n. However, this falls short of achieving our goal since the efficiency of ramp secret sharing schemes comes at the cost of decreasing a privacy threshold t to $n-k$. In our setting, this means that when sharing the $(n+1)$-dimensional vector $\mathbf{T}_r$, we need to set a privacy threshold $t = n - (n+1) < 0$, which guarantees no privacy.

To overcome this, we decompose the permuted truth-table $\mathbf{T}_r$ into ℓ vectors $\mathbf{T}_r = (\mathbf{U}^{(0)}, \mathbf{U}^{(1)}, \dots, \mathbf{U}^{(\ell-1)})$ each of dimension $k = (n+1)/\ell$. We independently generate shares of each vector $\mathbf{U}^{(j)}$ using a ramp secret sharing scheme. Now, a privacy threshold is $t = n - (n+1)/\ell = n - o(n)$ instead of $t = n - (n+1)$. In the online phase, players write $y = x + r$ as $y = \sigma k + \tau$ for some $\sigma \in \{0, 1, \dots, \ell - 1\}$ and $\tau \in \{0, 1, \dots, k-1\}$, which implies that the y-th component of $\mathbf{T}_r$ corresponds to the τ-th component of $\mathbf{U}^{(\sigma)}$. Then all players can together reconstruct the output $f(y - r) = h(x_1, \dots, x_n)$ by opening the τ-th component of $\mathbf{U}^{(\sigma)}$. A typical choice of the parameter ℓ is $\ell = \Theta(\log n)$. Then a privacy threshold is $t = n - \Theta(n/\log n)$ and correlated randomness for each player consists of $O(\ell) = O(\log n)$ shares. Since a ramp secret sharing scheme requires the underlying field to contain $n + k = O(n)$ elements, the size of correlated randomness is $O((\log n)^2)$ in bits. Note that the bottleneck complexity is still $O(\log n)$ since players open only one share in the online phase. On the other hand, if we set $\ell \approx 1/\epsilon$ for a constant $0 < \epsilon < 1/2$, then both the bottleneck complexity and the amount of correlated randomness are $O(\log n)$ while the number of corrupted players should be at most $(1-\epsilon)n$.

2.2 Our Second Protocol for Symmetric Functions

Next, we show a protocol which reduces the amount of correlated randomness to $O(\log n)$ bits at the cost of increasing bottleneck complexity to $O((\log n)^2)$. Our starting point is a balancing approach in [21] of expressing the truth-table of $f : \{0, 1, \dots, n\} \to \{0, 1\}$ (induced by a symmetric function h) as a matrix $\mathbf{M}_f$ instead of an $(n+1)$-dimensional vector. More specifically, assume that there are two distinct primes ℓ and k such that $\ell k = n + 1$, and fix the one-to-one correspondence ϕ between $\mathbb{Z}_{n+1} = \{0, 1, \dots, n\}$ and $\mathbb{Z}_\ell \times \mathbb{Z}_k = \{(\sigma, \tau) \in \mathbb{Z}^2 : 0 \le \sigma < \ell,\ 0 \le \tau < k\}$ induced by the Chinese remainder theorem. Then there exists a matrix $\mathbf{M}_f \in \{0,1\}^{\ell \times k}$ such that the computation of $f(\sum_{i \in [n]} x_i)$ can be expressed as the following inner product

$$f(\sum_{i \in [n]} x_i) = \langle \mathbf{e}_\sigma, \mathbf{M}_f \cdot \mathbf{e}_\tau \rangle, \tag{1}$$

where $(\sigma, \tau) = \phi(\sum_{i \in [n]} x_i)$ and $\mathbf{e}_j$ denotes the vector with a 1 in the j-th coordinate and 0's elsewhere. The task is now reduced to secure computation of matrix-vector products of size at most $\max\{\ell, k\}$, which balances bottleneck complexity and the amount of correlated randomness. However, if we naively implement secure computation of the inner product (1) by sharing secret vectors $\mathbf{e}_\sigma$ and $\mathbf{e}_\tau$ in an element-wise way, then the best possible bottleneck complexity is $\Omega(\sqrt{n})$ since the primes ℓ, k should satisfy $\ell k = \Omega(n)$.

To achieve polylogarithmic bottleneck complexity, we use a ramp secret sharing scheme and encode secret vectors $\mathbf{e}_\sigma$ and $\mathbf{e}_\tau$ into small shares. This reduces the secure computation of (1) to constructing protocols for the following functionalities:

Linear transformation. Players obtain ramp shares of an ℓ-dimensional vector $\mathbf{M} \cdot \mathbf{w}$ from shares of a k-dimensional secret vector $\mathbf{w}$, where $\mathbf{M}$ is a public ℓ-by-k matrix.

Inner product. Players obtain $\langle \mathbf{v}, \mathbf{w} \rangle$ from ramp shares of two ℓ-dimensional vectors $\mathbf{v}$ and $\mathbf{w}$.

We note that a protocol for the first functionality was previously considered in [14] but it requires every player to reshare their local shares, which results in $\Omega(n)$ bottleneck complexity. Our technical novelty is carefully designing correlated randomness to avoid such resharing processes and keep bottleneck complexity polylogarithmic in n.

Linear transformation. Ramp secret sharing schemes considered in this paper have linear reconstruction, that is, a secret vector can be expressed as a linear combination of all shares over a field. This implies that given shares of $\mathbf{w}$, every player can locally compute an ℓ-dimensional vector $\mathbf{s}_i$ such that $\mathbf{s}_1 + \cdots + \mathbf{s}_n = \mathbf{M} \cdot \mathbf{w}$. If players were allowed to reshare all the $\mathbf{s}_i$'s, they could securely obtain shares of $\mathbf{M} \cdot \mathbf{w}$. However, the resharing of all the $\mathbf{s}_i$'s results in high bottleneck complexity $\Omega(n)$. Instead, we distribute shares of a randomly chosen ℓ-dimensional vector $\mathbf{r}$ in the offline phase. This enables players to locally compute $\mathbf{x}_i$ such that $\mathbf{x}_1 + \cdots + \mathbf{x}_n = \mathbf{M} \cdot \mathbf{w} + \mathbf{r}$ and jointly reconstruct $\mathbf{M} \cdot \mathbf{w} + \mathbf{r}$, which can be done by communicating $O(\ell)$ field elements. Note that since $\mathbf{r}$ is unknown to any player, $\mathbf{M} \cdot \mathbf{w} + \mathbf{r}$ is just a random vector. It can be done locally to obtain shares of $\mathbf{M} \cdot \mathbf{w} + \mathbf{r}$ from it. Players then convert these shares into the ones of $\mathbf{M} \cdot \mathbf{w}$ by subtracting the shares of $\mathbf{r}$. In our protocol, players communicate only $O(\ell)$ field elements in the online phase and receive a constant number of field elements in the offline phase.

Inner product. Distributing Beaver triples [1] in the offline phase is a common technique to compute the product vw from shares of two secrets v and w. Although this technique successfully works when computing the product of *scalars*, a naive generalization does not work if we compute the inner product of *vectors* shared by a ramp scheme. More specifically, a common template using Beaver triples is distributing fresh shares of three secrets a, b and c in the offline phase, where a and b are randomly chosen and $c = ab$. In the online phase, players reconstruct $v - a$ and $w - b$, and then compute shares of vw based on the equation

$$vw = (v - a)(w - b) + a(w - b) + b(v - a) + c.$$

This can be done locally since vw is a linear combination of secrets a, b and c with public coefficients $v - a$ and $w - b$. To generalize this template, we distribute shares of secret vectors $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$, where $\mathbf{a}$ and $\mathbf{b}$ are random and $\mathbf{c} = \mathbf{a} * \mathbf{b}$, where $*$ denotes the element-wise product. As above, players reconstruct $\mathbf{v} - \mathbf{a}$ and $\mathbf{w} - \mathbf{b}$. Naturally, we extends the above equation to vectors:

$$\mathbf{v} * \mathbf{w} = (\mathbf{v} - \mathbf{a}) * (\mathbf{w} - \mathbf{b}) + \mathbf{a} * (\mathbf{w} - \mathbf{b}) + \mathbf{b} * (\mathbf{v} - \mathbf{a}) + \mathbf{c}.$$

It is easy to compute shares of the first term since $\mathbf{v} - \mathbf{a}$ and $\mathbf{w} - \mathbf{b}$ are public. A technical difficulty lies in computing shares of the second and third terms. When we only deal with scalars, players can locally compute shares of $a(w - b)$ from shares of a and a public constant $w - b$ just by multiplying the shares by the constant. However, when a secret vector $\mathbf{a}$ is shared by a ramp scheme, multiplying shares of $\mathbf{a}$ by a constant d results in shares of a vector $d \cdot \mathbf{a}$, whose entries are *all* multiplied by d. To obtain shares of $\mathbf{a} * (\mathbf{w} - \mathbf{b})$, we need to multiply different entries of a secret vector $\mathbf{a}$ by different constants. For that, we rewrite $\mathbf{a} * (\mathbf{w} - \mathbf{b}) = \mathrm{diag}(\mathbf{w} - \mathbf{b}) \cdot \mathbf{a}$ and apply the above protocol for linear transformation with $\mathbf{M} = \mathrm{diag}(\mathbf{w} - \mathbf{b})$, where $\mathrm{diag}(\mathbf{w} - \mathbf{b})$ denotes a diagonal matrix whose (i, i)-th entry is the i-th entry of $\mathbf{w} - \mathbf{b}$. Finally, players obtain shares of $\mathbf{v} * \mathbf{w}$, jointly reconstruct it, and output $\langle \mathbf{1}, \mathbf{v} * \mathbf{w} \rangle = \langle \mathbf{v}, \mathbf{w} \rangle$, where $\mathbf{1}$ is the all-one vector. Since naively reconstructing $\mathbf{v} * \mathbf{w}$ leaks additional information, we let players add shares of a random secret $\mathbf{s}$ such that $\langle \mathbf{1}, \mathbf{s} \rangle = 0$, which does not affect correctness since $\langle \mathbf{1}, \mathbf{v} * \mathbf{w} + \mathbf{s} \rangle = \langle \mathbf{v}, \mathbf{w} \rangle$. In this protocol, players communicate only $O(\ell)$ field elements in the online phase and receive a constant number of field elements in the offline phase.

Putting it altogether. Similarly to our first protocol, in the offline phase, we distribute additive shares of a random mask $r \in \{0, 1, \dots, n\}$ and ramp shares of vectors $\mathbf{e}_{\sigma_r}$ and $\mathbf{e}_{\tau_r}$, where $\phi(r) = (\sigma_r, \tau_r) \in \mathbb{Z}_\ell \times \mathbb{Z}_k$. In the online phase, players open a masked sum $y =$

$\sum_{i\in[n]} x_i - r$ and compute $\phi(y) = (\sigma_y, \tau_y)$. Note that $(\sigma_y+\sigma_r, \tau_y+\tau_r) = \phi(\sum_{i\in[n]} x_i) = (\sigma, \tau)$. Then, players obtain ramp shares of $\mathbf{e}_\sigma$ by applying the protocol for linear transformation with $\mathbf{w} = \mathbf{e}_{\sigma_r}$ and $\mathbf{M}$ being the linear operation of shifting a vector by σ_y. Similarly, players run the linear transformation protocol on ramp shares of $\mathbf{e}_{\tau_r}$ to obtain shares of $\mathbf{e}_\tau$. Subsequently, they apply the linear transformation protocol setting $\mathbf{w} = \mathbf{e}_\tau$ and $\mathbf{M} = \mathbf{M}_f$ to obtain ramp shares of $\mathbf{M}_f \cdot \mathbf{e}_\tau$. Finally, they run the inner product protocol on input $\mathbf{e}_\sigma$ and $\mathbf{M}_f \cdot \mathbf{e}_\tau$, and obtain $\langle \mathbf{e}_\sigma, \mathbf{M}_f \cdot \mathbf{e}_\tau \rangle = f(\sum_{i\in[n]} x_i) = h(x_1, \dots, x_n)$. A typical choice of the primes ℓ and k is $\ell = \Theta(\log n)$ and $k = \Theta(n/\log n)$. Since a ramp secret sharing scheme requires a field of size $O(n)$, a field element can be described in $O(\log n)$ bits. Therefore, the bottleneck complexity of our final protocol is $O(\ell \log n) = O((\log n)^2)$ and the per-party correlated randomness is $O(\log n)$ bits. On the other hand, a privacy threshold is $t = n - \max\{\ell, k\} = n - \Theta(n/\log n)$ since ℓ-dimensional and k-dimensional secret vectors are shared by a ramp scheme.

3 Preliminaries

3.1 Notations

For $m \in \mathbb{N}$, define $[m] = \{1, \dots, m\}$. Define $\mathbb{Z}_m$ as the ring of integers modulo m. We identify $\mathbb{Z}_m$ (as a set) with $\{z \in \mathbb{Z} : 0 \le z \le m-1\}$. For a subset X of a set Y, we define $Y \setminus X = \{y \in Y : y \notin X\}$. We write $u \leftarrow_\$ Y$ if u is chosen uniformly at random from a set Y. For a vector $\mathbf{s} = (s_i)_{i\in\mathbb{Z}_m} \in X^m$ and $r \in \mathbb{Z}_m$, we define $\mathsf{Shift}_r(\mathbf{s})$ as the vector obtained by shifting elements by r. Formally, $\mathbf{u} = (u_i)_{i\in\mathbb{Z}_m} = \mathsf{Shift}_r(\mathbf{s})$ is defined by $u_i = s_{(i-r) \bmod m}$ for all $i \in \mathbb{Z}_m$. If X is a field $\mathbb{F}$, Shift_r can be expressed by a linear operation. Formally, define a permutation matrix $\mathbf{P}_r \in \mathbb{F}^{m\times m}$ as the one whose (i, j)-th entry is 1 if $j = (i-r) \bmod m$ and 0 otherwise, where we identify the sets indexing the rows and columns of the matrix as $\mathbb{Z}_m$. Then it holds that $\mathsf{Shift}_r(\mathbf{s}) = \mathbf{P}_r \cdot \mathbf{s}$. It also holds that $\mathbf{P}_r^{-1} \cdot \mathbf{s} = \mathbf{P}_r^\top \cdot \mathbf{s} = \mathbf{P}_{-r} \cdot \mathbf{s} = \mathsf{Shift}_{-r}(\mathbf{s})$ Let $\mathbf{0}_m$ be the zero vector of dimension m and $\mathbf{1}_m$ be the all-ones vector of dimension m. We simply write $\mathbf{0}$ or $\mathbf{1}$ if the dimension is clear from the context. Let $\mathbf{e}_i$ denote the i-th unit vector whose entry is 1 at position i, and 0 otherwise. For a vector $\mathbf{v}$ of dimension m, we define $\mathrm{diag}(\mathbf{v})$ as a diagonal matrix whose (i, j)-th entry is the i-th entry of $\mathbf{v}$ if $j = i$ and 0 otherwise. For two vectors $\mathbf{u}, \mathbf{v}$ over a ring, we denote the standard inner product of $\mathbf{u}$ and $\mathbf{v}$ by $\langle \mathbf{u}, \mathbf{v} \rangle$. Throughout the paper, we fix the following notations:

- n is the total number of players.
- t is the maximum number of corrupted players (see Section 3.2).
- $\mathbb{K}$ is the minimum finite field such that $|\mathbb{K}| \ge 2n$. Fix $2n$ pairwise distinct elements $\beta_0, \beta_1, \dots, \beta_{n-1}, \alpha_1, \dots, \alpha_n \in \mathbb{K}$.

3.2 Secure Multiparty Computation

We denote the set of n players by $\{\mathsf{P}_1, \dots, \mathsf{P}_n\}$, where P_i is called the i-th player. Assume that each player P_i has a private input x_i from a finite set D. Let $\mathcal{F}(x_1, \dots, x_n) = (y_1, \dots, y_n)$ be an n-input/n-output randomized functionality. We assume the *correlated randomness model*, in which there is a trusted dealer who samples $(r_1, \dots, r_n)$ according to a joint distribution $\mathcal{D}$ over the Cartesian product $R_1 \times \cdots \times R_n$ of n sets, and gives $r_i \in R_i$ to each player P_i before he decides his input. We assume computationally unbounded adversaries who passively corrupt up to t players. (We do not consider active adversaries whose corrupted players deviate from protocols arbitrarily.) Let Π be a protocol between n players in the correlated randomness model. For a subset $T \subseteq [n]$ of size at most t and any input $\mathbf{x} = (x_i)_{i\in[n]}$, consider the following two processes:

Ideal process. This process is defined with respect to a simulator Sim. Let $(y_1, \ldots, y_n) \leftarrow \mathcal{F}(\mathbf{x})$. The output of this process is $\mathsf{Ideal}_{\mathcal{F},\mathsf{Sim}}(T, \mathbf{x}) := (\mathsf{Sim}(T, (x_i, y_i)_{i \in T}), (y_i)_{i \in [n]})$.

Real process. Suppose that all players each holding an input x_i execute Π honestly. Let $\mathsf{View}_{\Pi,i}(\mathbf{x})$ denote the view of P_i at the end of the protocol execution (which consists of his private input x_i, correlated randomness r_i, and messages that he received or sent during the execution of Π), and let $\mathsf{Output}_{\Pi,i}(\mathbf{x})$ be the output of P_i. The output of this process is $\mathsf{Real}_{\Pi}(T, \mathbf{x}) := ((\mathsf{View}_{\Pi,i}(\mathbf{x}))_{i \in T}, (\mathsf{Output}_{\Pi,i}(\mathbf{x}))_{i \in [n]})$.

We say that Π is a *t-secure MPC protocol for $\mathcal{F}$* if for any subset $T \subseteq [n]$ of size at most t and any input $\mathbf{x} = (x_i)_{i \in [n]}$, the distributions $\mathsf{Ideal}_{\mathcal{F},\mathsf{Sim}}(T, \mathbf{x})$ and $\mathsf{Real}_{\Pi}(T, \mathbf{x})$ are perfectly identical to each other.

Let g be a deterministic function on D^n. We say that Π is a t-secure protocol computing g if it is a t-secure protocol for the functionality that takes $\mathbf{x}$ as input and gives $g(\mathbf{x})$ to every player. Then we have that Π is a t-secure MPC protocol computing g if and only if

Correctness. For any input $\mathbf{x}$ and any $i \in [n]$, it holds with probability 1 that $\mathsf{Output}_{\Pi,i}(\mathbf{x}) = g(\mathbf{x})$.

Privacy. For any set $T \subseteq [n]$ of size at most t and any pair of inputs $\mathbf{x} = (x_i)_{i \in [n]}$, $\mathbf{w} = (w_i)_{i \in [n]}$ such that $(x_i)_{i \in T} = (w_i)_{i \in T}$ and $g(\mathbf{x}) = g(\mathbf{w})$, the distributions $(\mathsf{View}_{\Pi,i}(\mathbf{x}))_{i \in T}$ and $(\mathsf{View}_{\Pi,i}(\mathbf{w}))_{i \in T}$ are perfectly identical to each other.

We denote by $\mathrm{Comm}_i(\Pi)$ the total number of bits sent or received by the i-th player P_i during the execution of a protocol Π with worst-case inputs. We define the bottleneck complexity of Π as $\mathrm{BC}(\Pi) = \max_{i \in [n]}\{\mathrm{Comm}_i(\Pi)\}$. We denote by $\mathrm{Rand}_i(\Pi)$ the size of correlated randomness for P_i, i.e., the total number of bits received by P_i in the setup of Π, and define $\mathrm{CR}(\Pi) = \max_{i \in [n]}\{\mathrm{Rand}_i(\Pi)\}$. We denote by $\mathrm{Round}(\Pi)$ the round complexity of Π, i.e., the number of sequential rounds of interaction.

Let $\mathcal{G}$ be a functionality. We say that a protocol Π is in the *$\mathcal{G}$-hybrid model* if players invoke $\mathcal{G}$ during the execution of Π, that is, a trusted third party receives messages from players and gives them the correct output of $\mathcal{G}$. The composition theorem [28] implies that if a protocol Π securely realizes a functionality $\mathcal{F}$ in the $\mathcal{G}$-hybrid model and a protocol $\Pi_{\mathcal{G}}$ securely realizes $\mathcal{G}$, then the composition of Π and $\Pi_{\mathcal{G}}$, i.e., the protocol obtained by replacing all invocations of $\mathcal{G}$ in Π with $\Pi_{\mathcal{G}}$, also securely realizes $\mathcal{F}$. While the above theorem assumes sequential composition, a set of protocols in the paper can be composed concurrently.

3.3 Basic Algorithms and Protocols

Let $\mathbb{G}$ be an abelian group (e.g., a finite field or a ring of integers modulo m). Define $\mathsf{Additive}_{\mathbb{G}}(s)$ as an algorithm to generate additive shares over $\mathbb{G}$ for a secret $s \in \mathbb{G}$. Formally, on input $s \in \mathbb{G}$, $\mathsf{Additive}_{\mathbb{G}}(s)$ chooses $(s_1, \ldots, s_n) \in \mathbb{G}^n$ uniformly at random conditioned on $s = \sum_{i \in [n]} s_i$, and outputs it.

Broadcast. Let $\mathcal{F}_{\mathsf{Broadcast},i}$ be the functionality which receives an input y from the i-th player and gives y to all players. Since all players are supposed to be semi-honest, a protocol $\Pi_{\mathsf{Broadcast},i}$ realizing $\mathcal{F}_{\mathsf{Broadcast},i}$ with low bottleneck complexity is straightforward (see [21] for a formal description). Roughly speaking, assume that the set of n players is represented by a binary tree whose height is $O(\log n)$ and root is P_i. Each player sends his two children the element that he received from his parent node. The complexity of $\Pi_{\mathsf{Broadcast},i}$ is $\mathrm{CR}(\Pi_{\mathsf{Broadcast},i}) = 0$, $\mathrm{BC}(\Pi_{\mathsf{Broadcast},i}) = O(\ell_y)$, and $\mathrm{Round}(\Pi_{\mathsf{Broadcast},i}) = O(\log n)$, where ℓ_y is the bit-length of y.

Functionality $\mathcal{F}_{\mathsf{Sum}}((x_i)_{i\in[n]})$

Upon receiving a group element $x_i \in \mathbb{G}$ from each player P_i, $\mathcal{F}_{\mathsf{Sum}}$ gives every player $s := \sum_{i\in[n]} x_i$.

Protocol Π_{Sum}

Input. Each player P_i has a group element $x_i \in \mathbb{G}$.
Output. Every player obtains $s = \sum_{i\in[n]} x_i$.
Protocol.

1. Each player P_i chooses $r_i \leftarrow_\$ \mathbb{G}$ and sets $y_i = x_i + r_i$.
2. P_1 sends y_1 to P_2.
3. For each $i = 2, 3, \dots, n-1$, P_i lets z_{i-1} be the message from P_{i-1}, computes $z_i = z_{i-1} + y_i$, and sends z_i to P_{i+1}.
4. P_n sends $z_n = z_{n-1} + y_n$ to P_1.
5. P_1 sends $w_1 = z_n - r_1$ to P_2.
6. For each $i = 2, 3, \dots, n-1$, P_i lets w_{i-1} be the message from P_{i-1}, computes $w_i = w_{i-1} - r_i$, and sends w_i to P_{i+1}.
7. P_n computes $s = w_{n-1} - r_n$ and invokes $\mathcal{F}_{\mathsf{Broadcast},n}$ with input s.
8. Each player P_i outputs s.

Figure 1 The functionality $\mathcal{F}_{\mathsf{Sum}}$ and a protocol Π_{Sum} implementing it.

Sum. In Fig. 1, we describe the functionality $\mathcal{F}_{\mathsf{Sum}}$ which receives group elements $x_1, \dots, x_n \in \mathbb{G}$, each from P_i, and gives $s := \sum_{i\in[n]} x_i$ to all players. We show a protocol Π_{Sum} for $\mathcal{F}_{\mathsf{Sum}}$ without any correlated randomness while the protocol in [43, 21] requires correlated randomness of size $O(\log|\mathbb{G}|)$ per party. In our protocol, each player P_i masks his input x_i with a random element r_i, players compute $s' := \sum_{i\in[n]}(x_i + r_i)$ in a round-table structure, and then unmask s' in the same round-table structure. The formal description of Π_{Sum} is shown in Fig. 1. The complexities are $\mathrm{CR}(\Pi_{\mathsf{Sum}}) = 0$, $\mathrm{BC}(\Pi_{\mathsf{Sum}}) = O(\log|\mathbb{G}|)$ and $\mathrm{Round}(\Pi_{\mathsf{Sum}}) = O(n)$.

3.4 Ramp Secret Sharing

Recall that $\mathbb{K}$ is the minimum finite field such that $|\mathbb{K}| \geq 2n$ and we fix $2n$ pairwise distinct elements $\beta_0, \beta_1, \dots, \beta_{n-1}, \alpha_1, \dots, \alpha_n \in \mathbb{K}$. Let ℓ be a positive integer such that $\ell \leq n$. Define $\mathsf{RSS}_\ell(\mathbf{s})$ as an algorithm to generate shares of the (t, ℓ, n)-ramp secret sharing scheme for a secret vector $\mathbf{s} \in \mathbb{K}^\ell$. Formally, for $\mathbf{s} \in \mathbb{K}^\ell$, we define a set $\mathcal{R}_{\mathbf{s}}$ of polynomials as

$$\mathcal{R}_{\mathbf{s}} := \{\varphi \in \mathbb{K}[X] : \deg\varphi \leq t + \ell,\ (\varphi(\beta_0), \dots, \varphi(\beta_{\ell-1})) = \mathbf{s}\}$$

On input $\mathbf{s} \in \mathbb{K}^\ell$, $\mathsf{RSS}_\ell(\mathbf{s})$ chooses a polynomial φ uniformly at random from $\mathcal{R}_{\mathbf{s}}$, and then outputs $(\varphi(\alpha_1), \dots, \varphi(\alpha_n))$.

Regarding RSS_ℓ, we recall basic mathematical facts that we will use to construct our protocols in the following lemmas. We defer the proofs to the full version.

▶ **Lemma 3.** *Let $T \subseteq [n]$ be any set of size at most t and $\mathbf{s} \in \mathbb{K}^\ell$. Then, there is a polynomial $\Delta_{\mathbf{s}} \in \mathcal{R}_{\mathbf{s}}$ such that $\Delta_{\mathbf{s}}(\alpha_i) = 0$ for all $i \in T$.*

▶ **Lemma 4.** *Let* $\mathbf{s}, \mathbf{u} \in \mathbb{K}^\ell$ *and* $\varphi_{\mathbf{s}} \in \mathcal{R}_{\mathbf{s}}$. *If* $\varphi_{\mathbf{u}}$ *is uniformly distributed over* $\mathcal{R}_{\mathbf{u}}$, *then* $\varphi_{\mathbf{s}} + \varphi_{\mathbf{u}}$ *is uniformly distributed over* $\mathcal{R}_{\mathbf{s}+\mathbf{u}}$.

▶ **Lemma 5.** *Let* $\mathbf{s} = (s_0, \ldots, s_{\ell-1}) \in \mathbb{K}^\ell$. *Then, there is an algorithm* $\mathsf{Reconst}_\ell$ *such that* $\sum_{i\in[n]} \mathsf{Reconst}_\ell(j, i; v_i) = s_j$ *for any* j *and any possible shares* $(v_1, \ldots, v_n) \leftarrow \mathsf{RSS}_\ell(\mathbf{s})$. *Furthermore,* $\mathsf{Reconst}_\ell$ *is linear in the sense that* $\mathsf{Reconst}_\ell(j, i; v) + \mathsf{Reconst}_\ell(j, i; v') = \mathsf{Reconst}_\ell(j, i; v + v')$ *for any* $v, v' \in \mathbb{K}$.

We introduce a deterministic algorithm $\mathsf{FixedShare}_\ell$ that outputs predetermined shares consistent with a given secret vector. Formally, we fix a deterministic algorithm $\mathsf{FixedSample}_\ell$ which on input $\mathbf{s} \in \mathbb{K}^\ell$, computes a polynomial $\psi_{\mathbf{s}} \in \mathcal{R}_{\mathbf{s}}$. It can be implemented efficiently, e.g., with Gaussian elimination. Define $\mathsf{FixedShare}_\ell$ as follows: On input $i \in [n]$ and $\mathbf{s} \in \mathbb{K}^\ell$, $\mathsf{FixedShare}_\ell(i, \mathbf{s})$ computes $\psi_{\mathbf{s}} = \mathsf{FixedSample}_\ell(\mathbf{s})$ and outputs $\psi_{\mathbf{s}}(\alpha_i)$. Note that $(\mathsf{FixedShare}_\ell(i, \mathbf{s}))_{i\in[n]}$ is a tuple of possible shares of a secret vector $\mathbf{s}$.

4 Our First Protocol for Symmetric Functions

We call a function $h : \{0,1\}^n \to \{0,1\}$ *symmetric* if $h(x_{\sigma(1)}, \ldots, x_{\sigma(n)}) = h(x_1, \ldots, x_n)$ for any input $(x_1, \ldots, x_n) \in \{0,1\}^n$ and any permutation $\sigma : [n] \to [n]$. By definition, the value of a symmetric function h is determined only by the Hamming weight w of the input, i.e., $w := |\{i \in [n] : x_i = 1\}| = \sum_{i\in[n]} x_i$. Thus, there is the unique function $f : \{0, 1, \ldots, n\} \to \{0,1\}$ such that $f(x_1 + \cdots + x_n) = h(x_1, \ldots, x_n)$ for all $(x_1, \ldots, x_n) \in \{0,1\}^n$.

▶ Remark 6. The authors of [43, 39] considered a related class of functions called *abelian programs*. Specifically, a function $\tilde{h} : \mathbb{G}^n \to \{0,1\}$ is called an abelian program over an abelian group $\mathbb{G}$ if there exists a function $f : \mathbb{G} \to \{0,1\}$ such that $\tilde{h}(\tilde{x}_1, \ldots, \tilde{x}_n) = f(\tilde{x}_1 + \cdots + \tilde{x}_n)$ for all $(\tilde{x}_1, \ldots, \tilde{x}_n) \in \mathbb{G}^n$, where addition is taken over $\mathbb{G}$. As pointed out in [3], abelian programs can compute a symmetric function $h : \{0,1\}^n \to \{0,1\}$ by setting $\mathbb{G} = \mathbb{Z}_{n+1}$ and viewing each input $x_i \in \{0,1\}$ as an element $\tilde{x}_i \in \mathbb{Z}_{n+1}$ (i.e., embed $\{0,1\}$ into $\mathbb{Z}_{n+1}$). The authors of [39] presented an information-theoretic MPC protocol Π for an abelian program $\tilde{h} : \mathbb{G}^n \to \{0,1\}$ such that $\mathrm{CR}(\Pi) = O(|\mathbb{G}|)$ and $\mathrm{BC}(\Pi) = O(\log |\mathbb{G}|)$. Based on the above correspondence, the protocol has $\mathrm{CR}(\Pi) = O(n)$ and $\mathrm{BC}(\Pi) = O(\log n)$ when computing a symmetric function $h : \{0,1\}^n \to \{0,1\}$.

First, for a parameter ℓ, we show an $(n - \Theta(n/\ell))$-secure protocol for any symmetric function h such that the bottleneck complexity is $O(\log n)$ and the amount of correlated randomness is $O(\ell \log n)$. If we set $\ell = \Theta(\log n)$, then we obtain an $(n - o(n))$-secure protocol such that the bottleneck complexity is $O(\log n)$ and the amount of correlated randomness is $O((\log n)^2)$.

▶ **Theorem 7.** *Let* $h : \{0,1\}^n \to \{0,1\}$ *be a symmetric function. Let* ℓ *be any integer such that* $\ell \le n+1$, *and suppose that* $t \le n - \lceil (n+1)/\ell \rceil$. *The protocol* Π_{Sym} *described in Fig. 2 is a* t*-secure MPC protocol computing* h *in the* $\mathcal{F}_{\mathsf{Sum}}$*-hybrid model. Implementing* $\mathcal{F}_{\mathsf{Sum}}$, *the protocol* Π_{Sym} *achieves* $\mathrm{CR}(\Pi_{\mathsf{Sym}}) = O(\ell \log n)$ *and* $\mathrm{BC}(\Pi_{\mathsf{Sym}}) = O(\log n)$.

Proof. First, we prove the correctness of Π_{Sym}. Let $\mathbf{x} \in \{0,1\}^n$ be any input. Since $r = \sum_{i\in[n]} r_i$, it holds that $y = r + \sum_{i\in[n]} x_i$. Since $(v_i^{(j)})_{i\in[n]}$ are shares of RSS_k for a secret vector $\mathbf{U}^{(j)}$, it also holds that

$$z = \sum_{i\in[n]} z_i = \sum_{i\in[n]} \mathsf{Reconst}_k(\tau, i; v_i^{(\sigma)}) = (\mathbf{U}^{(\sigma)})_\tau = (\mathbf{S})_{\sigma k+\tau} = (\mathbf{S})_y = F_{(y-r) \bmod m}$$

where $(\mathbf{U}^{(\sigma)})_\tau$ is the τ-th element of $\mathbf{U}^{(\sigma)}$ and $(\mathbf{S})_y$ is the y-th element of $\mathbf{S}$. Therefore, we have that $z = f(\sum_{i\in[n]} x_i) = h(x_1, \ldots, x_n)$.

Next, we prove the privacy of Π_{Sym}. Let $T \subseteq [n]$ be the set of corrupted players. Let $H = [n] \setminus T$ be the set of honest players and fix an honest player $j \in H$. Note that in the $\mathcal{F}_{\mathsf{Sum}}$-hybrid model, corrupted players' view can be simulated from the following elements since the other elements are locally computed from them:
Correlated randomness. $(r_i, v_i^{(0)}, \dots, v_i^{(\ell-1)})$ for all $i \in T$;
Online messages. $y = \sum_{i \in [n]} x_i + r$ and z.
Let $\mathbf{x} = (x_i)_{i \in [n]}, \widetilde{\mathbf{x}} = (\widetilde{x}_i)_{i \in [n]} \in \{0,1\}^n$ be any pair of inputs such that $x_i = \widetilde{x}_i$ $(\forall i \in T)$ and $h(x_1, \dots, x_n) = h(\widetilde{x}_1, \dots, \widetilde{x}_n)$. It is sufficient to prove that the distribution of the above elements during the execution of Π_{Sym} on input $\mathbf{x}$ is identical to that on input $\widetilde{\mathbf{x}}$. To show the equivalence of the distributions, we show a bijection between the random strings used by Π_{Sym} on input $\mathbf{x}$ and the random strings used by Π_{Sym} on input $\widetilde{\mathbf{x}}$ such that the correlated randomness and the online messages received by T are the same under this bijection. The set of all random strings is

$$\mathcal{R} = \left\{ \left((r_i)_{i \in [n]}, \phi^{(0)}, \dots, \phi^{(\ell-1)} \right) : r_i \in \mathbb{Z}_m,\ \phi^{(j)} \in \mathcal{R}_{\mathbf{U}^{(j)}} \right\},$$

where $r = \sum_{i \in [n]} r_i$ and $(\mathbf{U}^{(0)}, \dots, \mathbf{U}^{(\ell-1)}) = \mathsf{Shift}_r(\mathbf{F})$. We denote the randomness of Π_{Sym} on input $\mathbf{x}$ by $R = ((r_i)_{i \in [n]}, \phi^{(0)}, \dots, \phi^{(\ell-1)})$ and that on input $\widetilde{\mathbf{x}}$ by $\widetilde{R} = ((\widetilde{r}_i)_{i \in [n]}, \widetilde{\phi}^{(0)}, \dots, \widetilde{\phi}^{(\ell-1)})$. We consider a bijection that maps the randomness $R \in \mathcal{R}$ to $\widetilde{R} \in \mathcal{R}$ in such a way that

$$\widetilde{r}_i = \begin{cases} r_i, & \text{if } i \in T, \\ r_i + x_i - \widetilde{x}_i, & \text{if } i \in H, \end{cases} \quad \text{and } \widetilde{\phi}^{(j)} = \phi^{(j)} + \Delta_{\widetilde{\mathbf{U}}^{(j)} - \mathbf{U}^{(j)}}$$

where

$$r := \sum_{i \in [n]} r_i,\ (\mathbf{U}^{(0)}, \dots, \mathbf{U}^{(\ell-1)}) := \mathsf{Shift}_r(\mathbf{F}),\ \widetilde{r} := \sum_{i \in [n]} \widetilde{r}_i,\ (\widetilde{\mathbf{U}}^{(0)}, \dots, \widetilde{\mathbf{U}}^{(\ell-1)}) := \mathsf{Shift}_{\widetilde{r}}(\mathbf{F}),$$

and $\Delta_{\widetilde{\mathbf{U}}^{(j)} - \mathbf{U}^{(j)}} \in \mathcal{R}_{\widetilde{\mathbf{U}}^{(j)} - \mathbf{U}^{(j)}}$ is a polynomial such that $\Delta_{\widetilde{\mathbf{U}}^{(j)} - \mathbf{U}^{(j)}}(\alpha_i) = 0$ for all $i \in T$, whose existence is guaranteed by Lemma 3. The image is indeed a consistent random string, i.e., $((\widetilde{r}_i)_{i \in [n]}, \widetilde{\phi}^{(0)}, \dots, \widetilde{\phi}^{(\ell-1)}) \in \mathcal{R}$, since $\phi^{(j)} \in \mathcal{R}_{\mathbf{U}^{(j)}}$ implies that $\widetilde{\phi}^{(j)} = \phi^{(j)} + \Delta_{\widetilde{\mathbf{U}}^{(j)} - \mathbf{U}^{(j)}} \in \mathcal{R}_{\widetilde{\mathbf{U}}^{(j)}}$. The above map is indeed a bijection since it has the inverse

$$r_i = \begin{cases} \widetilde{r}_i, & \text{if } i \in T, \\ \widetilde{r}_i + \widetilde{x}_i - x_i, & \text{if } i \in H, \end{cases} \quad \text{and } \phi^{(j)} = \widetilde{\phi}^{(j)} - \Delta_{\widetilde{\mathbf{U}}^{(j)} - \mathbf{U}^{(j)}}.$$

This bijection does not change the correlated randomness $(r_i, v_i^{(0)}, \dots, v_i^{(\ell-1)})_{i \in T}$ of T since $\widetilde{v}_i^{(j)} = \widetilde{\phi}^{(j)}(\alpha_i) = \phi^{(j)}(\alpha_i) + \Delta_{\widetilde{\mathbf{U}}^{(j)} - \mathbf{U}^{(j)}}(\alpha_i) = \phi^{(j)}(\alpha_i) = v_i^{(j)}$ for all $i \in T$. It can be seen that $\widetilde{x}_i + \widetilde{r}_i = \widetilde{x}_i + (r_i + x_i - \widetilde{x}_i) = x_i + r_i$ for $i \in H$. In particular, the message y is the same in both executions. Since $h(x_1, \dots, x_n) = h(\widetilde{x}_1, \dots, \widetilde{x}_n)$, the message z is also the same in both executions, which implies that the bijection does not change online messages seen by corrupted players.

Finally, since players also need to receive correlated randomness for two executions of the protocol Π_{Sum} implementing $\mathcal{F}_{\mathsf{Sum}}$, we have $\mathrm{CR}(\Pi_{\mathsf{Sym}}) = O(\log m + \ell \log |\mathbb{K}|) + O(\log m + \log |\mathbb{K}|) = O(\ell \log n)$ and $\mathrm{BC}(\Pi_{\mathsf{Sym}}) = O(\log m + \log |\mathbb{K}|) = O(\log n)$. ◀

Setting $\ell = \Theta(\log n)$, we obtain the following corollary.

▶ **Corollary 8.** *If $t = n - \Theta(n/\log n)$, then there exists a t-secure MPC protocol Π computing a symmetric function $h : \{0,1\}^n \to \{0,1\}$ such that $\mathrm{CR}(\Pi) = O((\log n)^2)$ and $\mathrm{BC}(\Pi) = O(\log n)$.*

Protocol Π_{Sym}

Notations.

- Let $h : \{0,1\}^n \to \{0,1\}$ be a symmetric function.
- Let $f : \{0,1,\dots,n\} \to \{0,1\}$ be a function such that $h(x_1,\dots,x_n) = f(\sum_{i\in[n]} x_i)$ for all $(x_1,\dots,x_n) \in \{0,1\}^n$.
- Let $\ell \le n+1$, $k := \lceil (n+1)/\ell \rceil$ and $m := \ell k$.
- Define $\mathbf{F} = (F_i)_{i\in\mathbb{Z}_m} \in \mathbb{K}^m$ by $F_i = f(i)$ if $0 \le i \le n$ and $F_i = 0$ otherwise.

Input. Each player P_i has $x_i \in \{0,1\}$.

Output. Every player obtains $z = h(x_1,\dots,x_n)$.

Setup.

1. Let $r \leftarrow_{\$} \mathbb{Z}_m$ and $(r_i)_{i\in[n]} \leftarrow \mathsf{Additive}_{\mathbb{Z}_m}(r)$.
2. Define $\mathbf{S} \in \mathbb{K}^m$ by $\mathbf{S} = \mathsf{Shift}_r(\mathbf{F})$ and decompose $\mathbf{S}$ into ℓ vectors $\mathbf{U}^{(0)},\dots,\mathbf{U}^{(\ell-1)}$ of dimension k, i.e., $\mathbf{S} = (\mathbf{U}^{(0)},\dots,\mathbf{U}^{(\ell-1)})$.
3. For each $j = 0,1,\dots,\ell-1$, let $(v_i^{(j)})_{i\in[n]} \leftarrow \mathsf{RSS}_k(\mathbf{U}^{(j)})$.
4. Each player P_i receives $(r_i, v_i^{(0)},\dots,v_i^{(\ell-1)})$.

Protocol.

1. Each player P_i computes $y_i = x_i + r_i \bmod m$.
2. Players obtain $y = \mathcal{F}_{\mathsf{Sum}}((y_i)_{i\in[n]})$.
3. Each player computes $(\sigma,\tau) \in \mathbb{Z}_\ell \times \mathbb{Z}_k$ such that $y = \sigma k + \tau$.
4. Each player P_i computes $z_i = \mathsf{Reconst}_k(\tau, i; v_i^{(\sigma)})$.
5. Players obtain $z = \mathcal{F}_{\mathsf{Sum}}((z_i)_{i\in[n]})$.
6. Each player P_i outputs z.

Figure 2 Our first protocol Π_{Sym} for computing a symmetric function.

Setting $\ell \approx 1/\epsilon$ for a constant $0 < \epsilon < 1/2$, we also obtain a $(1-\epsilon)n$-secure protocol such that both bottleneck complexity and the amount of correlated randomness are $O(\log n)$.

▶ **Corollary 9.** *For any constant ϵ such that $0 < \epsilon < 1/2$, there exists a $(1-\epsilon)n$-secure MPC protocol Π computing a symmetric function $h : \{0,1\}^n \to \{0,1\}$ such that* $\mathrm{CR}(\Pi) = O(\epsilon^{-1}\log n) = O(\log n)$ *and* $\mathrm{BC}(\Pi) = O(\log n)$.

▶ Remark 10 (Round complexity). We have $\mathrm{Round}(\Pi_{\mathsf{Sym}}) = O(n)$ if $\mathcal{F}_{\mathsf{Sum}}$ is instantiated with Π_{Sum}. The round complexity of Π_{Sum} can be reduced to $O(\log n)$ without changing asymptotic bottleneck complexity and amount of correlated randomness. Indeed, there is a more round-efficient protocol Π'_{Sum} realizing $\mathcal{F}_{\mathsf{Sum}}$ such that $\mathrm{CR}(\Pi'_{\mathsf{Sum}}) = O(\log|\mathbb{G}|)$, $\mathrm{BC}(\Pi'_{\mathsf{Sum}}) = O(\log|\mathbb{G}|)$, and $\mathrm{Round}(\Pi'_{\mathsf{Sum}}) = O(\log n)$, where $\mathbb{G}$ is an abelian group from which inputs take values [21]. If we implement $\mathcal{F}_{\mathsf{Sum}}$with Π'_{Sum}, then Π_{Sym} achieves $\mathrm{Round}(\Pi_{\mathsf{Sym}}) = O(\log n)$. This modification increases the amount of correlated randomness by $O(\log m) + O(\log|\mathbb{K}|) = O(\log n)$ but does not change overall complexities in an asymptotic sense. In summary, we have a t-secure MPC protocol Π_{Sym} for h such that $\mathrm{CR}(\Pi_{\mathsf{Sym}}) = O(\ell\log n)$, $\mathrm{BC}(\Pi_{\mathsf{Sym}}) = O(\log n)$ and $\mathrm{Round}(\Pi_{\mathsf{Sym}}) = O(\log n)$.

▶ Remark 11 (Computational complexity). Each player receives $O(\ell)$ elements in $\mathbb{K}$ and performs a constant number of operations in $\mathbb{K}$. The computational complexity of Π_{Sym} is thus $O(\ell)$ field operations.

5 Our Second Protocol for Symmetric Functions

In this section, we show a protocol for any symmetric function whose bottleneck complexity is $O((\log n)^2)$ and amount of correlated randomness is $O(\log n)$. First, we construct two building-block protocols with low bottleneck complexity, and then we show our main protocol.

5.1 Additional Building Blocks

For parameters k, ℓ, we consider the following sub-functionalities:

Linear transformation $\mathcal{F}_{\mathsf{LT}}$. Given ramp shares of a k-dimensional secret vector $\mathbf{s}$, players obtain ramp shares of an ℓ-dimensional vector $\mathbf{u} := \mathbf{M} \cdot \mathbf{s}$, where $\mathbf{M}$ is a public ℓ-by-k matrix. The formal description is shown in Fig. 4.

Inner product $\mathcal{F}_{\mathsf{IP}}$. Given ramp shares of two ℓ-dimensional vectors $\mathbf{v}$ and $\mathbf{w}$, players obtain the inner product $\langle \mathbf{v}, \mathbf{w} \rangle$. The formal description is shown in Fig. 5.

We show protocols for $\mathcal{F}_{\mathsf{LT}}$ and $\mathcal{F}_{\mathsf{IP}}$. The formal descriptions and proofs are given in Appendices A and B.

▶ **Proposition 12.** *Let k, ℓ be positive integers with $\ell \leq k \leq n$ and $\mathbf{M}$ be an ℓ-by-k matrix over $\mathbb{K}$. Suppose that $t \leq n - \ell$. Then, the protocol Π_{LT} described in Fig. 4 is a t-secure MPC protocol for $\mathcal{F}_{\mathsf{LT}}$ in the $\mathcal{F}_{\mathsf{Sum}}$-hybrid model. Implementing $\mathcal{F}_{\mathsf{Sum}}$, the protocol Π_{LT} achieves $\mathrm{CR}(\Pi_{\mathsf{LT}}) = O(\log n)$ and $\mathrm{BC}(\Pi_{\mathsf{LT}}) = O(\ell \log n)$.*

▶ **Proposition 13.** *Let ℓ be a positive integer with $\ell \leq n$. Suppose that $t \leq n - \ell$. Then, the protocol Π_{IP} described in Fig. 5 is a t-secure MPC protocol for $\mathcal{F}_{\mathsf{IP}}$ in the $(\mathcal{F}_{\mathsf{Sum}}, \mathcal{F}_{\mathsf{LT}})$-hybrid model. Implementing $\mathcal{F}_{\mathsf{Sum}}$ and $\mathcal{F}_{\mathsf{LT}}$, the protocol Π_{IP} achieves $\mathrm{CR}(\Pi_{\mathsf{IP}}) = O(\log n)$ and $\mathrm{BC}(\Pi_{\mathsf{IP}}) = O(\ell \log n)$.*

5.2 Main Protocol

Now, for two primes k, ℓ with $\ell k > n$, we show an $(n-k)$-secure protocol for any symmetric function h such that the bottleneck complexity is $O(\ell \log n)$ and the amount of correlated randomness is $O(\log n)$.

▶ **Theorem 14.** *Let $h : \{0,1\}^n \to \{0,1\}$ be a symmetric function. Let ℓ, k be primes such that $\ell < k$ and $n + 1 \leq \ell k \leq O(n)$, and suppose that $t \leq n - k$. The protocol Π'_{Sym} described in Fig. 3 is a t-secure MPC protocol for $\mathcal{F}_h$ in the $(\mathcal{F}_{\mathsf{Sum}}, \mathcal{F}_{\mathsf{LT}}, \mathcal{F}_{\mathsf{IP}})$-hybrid model. Implementing $\mathcal{F}_{\mathsf{Sum}}$, $\mathcal{F}_{\mathsf{LT}}$ and $\mathcal{F}_{\mathsf{IP}}$, the protocol Π'_{Sym} achieves $\mathrm{CR}(\Pi'_{\mathsf{Sym}}) = O(\log n)$ and $\mathrm{BC}(\Pi'_{\mathsf{Sym}}) = O(\ell \log n)$.*

Proof. First, we prove the correctness of Π'_{Sym}. Let $\mathbf{x} \in \{0,1\}^n$ be any input. Since $r = \sum_{i \in [n]} r_i$, it holds that $y = r - \sum_{i \in [n]} x_i$. Let $(\sigma', \tau') := (\sigma + u, \tau + v)$. Note that we have $\phi(\sum_{i \in [n]} x_i) = \phi(y) + \phi(r) = (\sigma', \tau')$. Since $(d_i)_{i \in [n]}$ are shares of RSS_ℓ for a secret vector $\mathbf{e}_v$, the functionality of $\mathcal{F}_{\mathsf{LT}}$ implies that $(d'_i)_{i \in [n]}$ are shares of a secret vector $\mathbf{N}_y \cdot \mathbf{e}_v = \mathbf{P}_\sigma^\top \cdot \mathbf{M} \cdot \mathbf{P}_\tau \cdot \mathbf{e}_v = \mathbf{P}_\sigma^\top \cdot \mathbf{M} \cdot \mathbf{e}_{\tau'}$. Furthermore, since $(c_i)_{i \in [n]}$ are shares of RSS_k for a secret vector $\mathbf{e}_u$, the functionality of $\mathcal{F}_{\mathsf{IP}}$ implies that $z = \langle \mathbf{e}_u, \mathbf{P}_\sigma^\top \cdot \mathbf{M} \cdot \mathbf{e}_{\tau'} \rangle = \langle \mathbf{P}_\sigma \cdot \mathbf{e}_u, \mathbf{M} \cdot \mathbf{e}_{\tau'} \rangle = \langle \mathbf{e}_{\sigma'}, \mathbf{M} \cdot \mathbf{e}_{\tau'} \rangle = \mathbf{M}[\sigma', \tau']$ where $\mathbf{M}[\sigma', \tau']$ is the (σ', τ')-th entry of $\mathbf{M}$. Therefore, we have that $z = f(\phi^{-1}(\sigma', \tau')) = f(\sum_{i \in [n]} x_i) = h(x_1, \ldots, x_n)$.

Next, we prove the privacy of Π'_{Sym}. Let $T \subseteq [n]$ be the set of corrupted players. Recall that α_i (resp. β_j) is the point associated with the i-th share (resp. the j-th component of a secret vector) of RSS_ℓ and RSS_k. To simplify notations, we denote $(\varphi(\alpha_i))_{i \in T}$ by $\varphi(\alpha_T)$ for a

polynomial φ. In the $\mathcal{F}_{\mathsf{Sum}}$-hybrid model, corrupted players' view at Step 2 only contains their inputs $(y_i)_{i\in T}$ to $\mathcal{F}_{\mathsf{Sum}}$ and the output y. Also, in the $\mathcal{F}_{\mathsf{LT}}$-hybrid model, corrupted players' view at Step 5 (including their correlated randomness for $\mathcal{F}_{\mathsf{LT}}$) only contains their inputs $(d_i)_{i\in T}$ to $\mathcal{F}_{\mathsf{LT}}$ and the outputs $(d'_i)_{i\in T}$. It is sufficient to show that the joint distribution of the following elements is simulated from $(x_i)_{i\in T}$ and $h(x_1,\dots,x_n)$ since the other elements are locally computed from them:
Correlated randomness. (r_i, c_i, d_i) for all $i \in T$;
Online messages. $y = \sum_{i\in[n]} x_i + r$, $(d'_i)_{i\in T}$, and z.
To analyze the distribution of the above element, we define $\mathsf{View} = ((r_i, c_i, d_i, d'_i)_{i\in T}, y, z)$. Observe that the distribution of View is given by

$$\mathsf{View} = \left((r_i)_{i\in T}, \phi_c(\alpha_T), \phi_d(\alpha_T), \phi_{d'}(\alpha_T), y = \sum_{i\in[n]} x_i + \sum_{i\in[n]} r_i, z\right),$$

where $(r_1,\dots,r_n) \leftarrow_\$ \mathbb{Z}_m^n$, $(u,v) = \phi(\sum_{i\in[n]} r_i)$, $\phi_c \leftarrow_\$ \mathcal{R}_{\mathbf{e}_u}$, $\phi_d \leftarrow_\$ \mathcal{R}_{\mathbf{e}_v}$, and $\phi_{d'} \leftarrow_\$ \mathcal{R}_{\mathbf{N}_y\cdot\mathbf{e}_v}$. The correctness of Π'_{Sym} implies that

$$\mathsf{View} = \left((r_i)_{i\in T}, \phi_c(\alpha_T), \phi_d(\alpha_T), \phi_{d'}(\alpha_T), y = \sum_{i\in[n]} x_i + \sum_{i\in[n]} r_i, h(x_1,\dots,x_n)\right).$$

Lemma 3 ensures that for any $\mathbf{v} \in \mathbb{K}^\ell$, there is a polynomial $\Delta_{\mathbf{v}} \in \mathcal{R}_{\mathbf{v}}$ such that $\Delta_{\mathbf{v}}(\alpha_i) = 0$ for all $i \in T$. If $\widetilde{\phi}_c$ are uniformly distributed over $\mathcal{R}_{\mathbf{0}_\ell}$, then $\widetilde{\phi}_c + \Delta_{\mathbf{e}_u}$ is uniformly distributed over $\mathcal{R}_{\mathbf{e}_u}$ from Lemma 4 and $(\widetilde{\phi}_c + \Delta_{\mathbf{e}_u})(\alpha_i) = 0$ for all $i \in T$. Similarly, if $\widetilde{\phi}_d, \widetilde{\phi}_{d'} \leftarrow_\$ \mathcal{R}_{\mathbf{0}_k}$, then it holds that $\widetilde{\phi}_d + \Delta_{\mathbf{e}_v} \leftarrow_\$ \mathcal{R}_{\mathbf{e}_v}$ and $\widetilde{\phi}_{d'} + \Delta_{\mathbf{N}_y\cdot\mathbf{e}_v} \leftarrow_\$ \mathcal{R}_{\mathbf{N}_y\cdot\mathbf{e}_v}$. It also holds that $(\widetilde{\phi}_d + \Delta_{\mathbf{e}_v})(\alpha_i) = \widetilde{\phi}_d(\alpha_i)$ and $(\widetilde{\phi}_{d'} + \Delta_{\mathbf{N}_y\cdot\mathbf{e}_v})(\alpha_i) = \widetilde{\phi}_{d'}(\alpha_i)$ for all $i \in T$. We thus have that

$$\mathsf{View} = \left((r_i)_{i\in T}, \widetilde{\phi}_c(\alpha_T), \widetilde{\phi}_d(\alpha_T), \widetilde{\phi}_{d'}(\alpha_T), y = \sum_{i\in[n]} x_i + \sum_{i\in[n]} r_i, h(x_1,\dots,x_n)\right),$$

where $(r_1,\dots,r_n) \leftarrow_\$ \mathbb{Z}_m^n$, $\widetilde{\phi}_c \leftarrow_\$ \mathcal{R}_{\mathbf{0}_\ell}$, $\widetilde{\phi}_d, \widetilde{\phi}_{d'} \leftarrow_\$ \mathcal{R}_{\mathbf{0}_k}$. Since $T \neq [n]$ and $(r_i)_{i\in[n]}$ are independent and uniformly random elements, the joint distribution of $(r_i)_{i\in T}$ and $y = \sum_{i\in[n]} x_i + \sum_{i\in[n]} r_i$ is the uniform distribution over $\mathbb{Z}_m^{|T|+1}$. We thus have that

$$\mathsf{View} = \left((\widetilde{r}_i)_{i\in T}, \widetilde{\phi}_c(\alpha_T), \widetilde{\phi}_d(\alpha_T), \widetilde{\phi}_{d'}(\alpha_T), \widetilde{y}, h(x_1,\dots,x_n)\right),$$

where $((\widetilde{r}_i)_{i\in T}, \widetilde{y}) \leftarrow_\$ \mathbb{Z}_m^{|T|+1}$, $\widetilde{\phi}_c \leftarrow_\$ \mathcal{R}_{\mathbf{0}_\ell}$, $\widetilde{\phi}_d, \widetilde{\phi}_{d'} \leftarrow_\$ \mathcal{R}_{\mathbf{0}_k}$. Therefore, we conclude that View is simulated from $h(x_1,\dots,x_n)$ only.

Finally, since players also need to receive correlated randomness for executions of protocols implementing $\mathcal{F}_{\mathsf{Sum}}$, $\mathcal{F}_{\mathsf{LT}}$ and $\mathcal{F}_{\mathsf{IP}}$, we have $\mathrm{CR}(\Pi'_{\mathsf{Sym}}) = O(\log m) + O(\log n) + O(\log n) = O(\log n)$ and $\mathrm{BC}(\Pi'_{\mathsf{Sym}}) = O(\log m) + O(\ell \log n) + O(\ell \log n) = O(\ell \log n)$. ◀

Thanks to Bertrand's postulate [45, Theorem 5.8], we can choose primes k, ℓ such that $k = \Theta(n/\log n)$ and $\ell = \Theta(\log n)$. Then we obtain an $(n - o(n))$-secure protocol such that the bottleneck complexity is $O((\log n)^2)$ and the amount of correlated randomness is $O(\log n)$. More formally, the following corollary holds.

▶ **Corollary 15.** *If $t = n - \Theta(n/\log n)$, then there exists a t-secure MPC protocol Π computing a symmetric function $h : \{0,1\}^n \to \{0,1\}$ such that $\mathrm{CR}(\Pi) = O(\log n)$ and $\mathrm{BC}(\Pi) = O((\log n)^2)$.*

Protocol Π'_{Sym}

Notations.

- Let $h : \{0,1\}^n \to \{0,1\}$ be a symmetric function.
- Let $f : \{0,1,\dots,n\} \to \{0,1\}$ be a function such that $h(x_1,\dots,x_n) = f(\sum_{i\in[n]} x_i)$ for all $(x_1,\dots,x_n) \in \{0,1\}^n$.
- Let ℓ, k be primes such that $\ell < k$ and $n+1 \le \ell k$, and set $m = \ell k$.
- Let $\phi : \mathbb{Z}_m \to \mathbb{Z}_\ell \times \mathbb{Z}_k$ be the ring isomorphism induced by the Chinese remainder theorem.
- Define a matrix $\mathbf{M} \in \mathbb{K}^{\ell\times k}$ as follows: For $(\sigma,\tau) \in \mathbb{Z}_\ell \times \mathbb{Z}_k$, the (σ,τ)-th entry of $\mathbf{M}$ is $f(\phi^{-1}(\sigma,\tau))$ if $\phi^{-1}(\sigma,\tau) \in \{0,1,\dots,n\}$, and 0 otherwise, where we identify the sets indexing the rows and columns of $\mathbf{M}$ as $\mathbb{Z}_\ell$ and $\mathbb{Z}_k$, respectively.

Input. Each player P_i has $x_i \in \{0,1\}$.

Output. Every player obtains $z = h(x_1,\dots,x_n)$.

Setup.

1. Let $r \leftarrow_{\$} \mathbb{Z}_m$, $(r_i)_{i\in[n]} \leftarrow \mathsf{Additive}_{\mathbb{Z}_m}(r)$, and $(u,v) = \phi(r)$.
2. Let $(c_i)_{i\in[n]} \leftarrow \mathsf{RSS}_\ell(\mathbf{e}_u)$ and $(d_i)_{i\in[n]} \leftarrow \mathsf{RSS}_k(\mathbf{e}_v)$, where $\mathbf{e}_u \in \mathbb{K}^\ell$ (resp. $\mathbf{e}_v \in \mathbb{K}^k$) is the unit vector whose entry is 1 at position $u \in \mathbb{Z}_\ell$ (resp. $v \in \mathbb{Z}_k$), and 0 otherwise.
3. Each player P_i receives (r_i, c_i, d_i).

Protocol.

1. Each player P_i computes $y_i = x_i - r_i \bmod m$.
2. Players obtain $y = \mathcal{F}_{\mathsf{Sum}}((y_i)_{i\in[n]})$.
3. Each player computes $(\sigma,\tau) = \phi(y) \in \mathbb{Z}_\ell \times \mathbb{Z}_k$ and $\mathbf{N}_y = \mathbf{P}_\sigma^\top \cdot \mathbf{M} \cdot \mathbf{P}_\tau$.
4. Players obtain $(d'_i)_{i\in[n]} \leftarrow \mathcal{F}_{\mathsf{LT}}(\mathbf{N}_y; (d_i)_{i\in[n]})$.
5. Players obtain $z \leftarrow \mathcal{F}_{\mathsf{IP}}((c_i, d'_i)_{i\in[n]})$.
6. Each player P_i outputs z.

Figure 3 Our second protocol Π'_{Sym} for computing a symmetric function.

Note that setting k and ℓ as primes close to ϵn and $1/\epsilon$ (resp.) leads to a protocol with asymptotically the same complexity as Corollary 9.

▶ Remark 16 (Round and computational complexity). The round complexity of Π'_{Sym} is $\mathrm{Round}(\Pi'_{\mathsf{Sym}}) = O(n)$. Since the computation of $\mathbf{N}_y = \mathbf{P}_\sigma^\top \cdot \mathbf{M} \cdot \mathbf{P}_\tau$ is just permuting rows and columns of $\mathbf{M}$, it can be done by $O(\ell k)$ field operations. The computational complexities of Π_{LT} implementing $\mathcal{F}_{\mathsf{LT}}$ and Π_{IP} implementing $\mathcal{F}_{\mathsf{IP}}$ are $O(\ell k)$ and $O(\ell^2)$ field operations, respectively. Since $\ell < k$, the computational complexity of Π'_{Sym} is $O(\ell k) = O(n)$ field operations.

References

1 Donald Beaver. Efficient multiparty protocols using circuit randomization. In *Advances in Cryptology – CRYPTO '91*, pages 420–432, 1992.

2 Zuzana Beerliová-Trubíniová and Martin Hirt. Perfectly-secure mpc with linear communication complexity. In *Theory of Cryptography*, pages 213–230, 2008.

3 Amos Beimel, Ariel Gabizon, Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, and Anat Paskin-Cherniavsky. Non-interactive secure multiparty computation. In *Advances in Cryptology – CRYPTO 2014, Part II*, pages 387–404, 2014.

4 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 1–10, 1988.

5 Eli Ben-Sasson, Serge Fehr, and Rafail Ostrovsky. Near-linear unconditionally-secure multiparty computation with a dishonest minority. In *Advances in Cryptology – CRYPTO 2012*, pages 663–680, 2012.

6 Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *Advances in Cryptology – EUROCRYPT 2011*, pages 169–188, 2011.

7 G. R. Blakley and C. Meadows. Security of ramp schemes. In *Advances in Cryptology – CRYPTO '84*, pages 242–268, 1985.

8 Elette Boyle, Niv Gilboa, and Yuval Ishai. Secure computation with preprocessing via function secret sharing. In *Theory of Cryptography*, pages 341–371, 2019.

9 Elette Boyle, Niv Gilboa, Yuval Ishai, and Ariel Nof. Sublinear gmw-style compiler for mpc with preprocessing. In *Advances in Cryptology – CRYPTO 2021*, pages 457–485, 2021.

10 Elette Boyle, Abhishek Jain, Manoj Prabhakaran, and Ching-Hua Yu. The Bottleneck Complexity of Secure Multiparty Computation. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 24:1–24:16, 2018.

11 David Chaum, Claude Crépeau, and Ivan Damgard. Multiparty unconditionally secure protocols. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 11–19, 1988.

12 Koji Chida, Daniel Genkin, Koki Hamada, Dai Ikarashi, Ryo Kikuchi, Yehuda Lindell, and Ariel Nof. Fast large-scale honest-majority MPC for malicious adversaries. In *Advances in Cryptology – CRYPTO 2018, Part III*, pages 34–64, 2018.

13 Geoffroy Couteau. A note on the communication complexity of multiparty computation in the correlated randomness model. In *Advances in Cryptology – EUROCRYPT 2019*, pages 473–503, 2019.

14 Ronald Cramer, Ivan Damgård, and Robbert de Haan. Atomic secure multi-party multiplication with low communication. In *Advances in Cryptology – EUROCRYPT 2007*, pages 329–346, 2007.

15 Ronald Cramer, Ivan Damgård, and Ueli Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *Advances in Cryptology – EUROCRYPT 2000*, pages 316–334, 2000.

16 Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In *Advances in Cryptology – EUROCRYPT 2010*, pages 445–465, 2010.

17 Ivan Damgård and Jesper Buus Nielsen. Scalable and unconditionally secure multiparty computation. In *Advances in Cryptology – CRYPTO 2007*, pages 572–590, 2007.

18 Ivan Damgård, Jesper Buus Nielsen, Michael Nielsen, and Samuel Ranellucci. The tinytable protocol for 2-party secure computation, or: Gate-scrambling revisited. In *Advances in Cryptology – CRYPTO 2017*, pages 167–187, 2017.

19 Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology – CRYPTO 2012*, pages 643–662, 2012.

20 Varsha Dani, Valerie King, Mahnush Movahedi, Jared Saia, and Mahdi Zamani. Secure multi-party computation in large networks. *Distributed Computing*, 30:193–229, 2017.

21 Reo Eriguchi. Unconditionally secure multiparty computation for symmetric functions with low bottleneck complexity. In *Advances in Cryptology – ASIACRYPT 2023*, pages 335–368, 2023.

22 Daniel Escudero, Vipul Goyal, Antigoni Polychroniadou, and Yifan Song. TurboPack: Honest majority mpc with constant online communication. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, pages 951–964, 2022.

23 Matthew Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '92, pages 699–710, 1992.

24 Yuval Gelles and Ilan Komargodski. Optimal load-balanced scalable distributed agreement. Cryptology ePrint Archive, Paper 2023/1139, 2023. URL: `https://eprint.iacr.org/2023/1139`.

25 Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–0178, 2009.

26 Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology – CRYPTO 2013*, pages 75–92, 2013.

27 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, 1987.

28 Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge University Press, 2009.

29 Vipul Goyal, Hanjun Li, Rafail Ostrovsky, Antigoni Polychroniadou, and Yifan Song. ATLAS: Efficient and scalable MPC in the honest majority setting. In *Advances in Cryptology – CRYPTO 2021, Part II*, pages 244–274, 2021.

30 Vipul Goyal, Yanyi Liu, and Yifan Song. Communication-efficient unconditional MPC with guaranteed output delivery. In *Advances in Cryptology – CRYPTO 2019, Part II*, pages 85–114, 2019.

31 Vipul Goyal, Yifan Song, and Chenzhi Zhu. Guaranteed output delivery comes free in honest majority MPC. In *Advances in Cryptology – CRYPTO 2020, Part II*, pages 618–646, 2020.

32 Shai Halevi, Yuval Ishai, Abhishek Jain, Eyal Kushilevitz, and Tal Rabin. Secure multiparty computation with general interaction patterns. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, ITCS '16, pages 157–168, 2016.

33 Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In *Advances in Cryptology – CRYPTO 2011*, pages 132–150, 2011.

34 Martin Hirt and Ueli Maurer. Robustness for free in unconditional multi-party computation. In *Advances in Cryptology – CRYPTO 2001*, pages 101–118, 2001.

35 Martin Hirt, Ueli Maurer, and Bartosz Przydatek. Efficient secure multi-party computation. In *Advances in Cryptology – ASIACRYPT 2000*, pages 143–161, 2000.

36 Martin Hirt and Daniel Tschudi. Efficient general-adversary multi-party computation. In *Advances in Cryptology – ASIACRYPT 2013, Part II*, pages 181–200, 2013.

37 Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In *Theory of Cryptography*, pages 600–620, 2013.

38 Stasys Jukna. *Boolean Function Complexity*. Springer, Berlin, Heidelberg, 1 edition, 2012.

39 Hannah Keller, Claudio Orlandi, Anat Paskin-Cherniavsky, and Divya Ravi. MPC with low bottleneck-complexity: Information-theoretic security and more. In *4th Information-Theoretic Cryptography (ITC) Conference*, 2023. URL: `https://eprint.iacr.org/2023/683`.

40 Marcel Keller. MP-SPDZ: A versatile framework for multi-party computation. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, pages 1575–1590, 2020.

41 Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Scalable leader election. In *SODA*, volume 6, pages 990–999, 2006.

42 R. J. McEliece and D. V. Sarwate. On sharing secrets and Reed-Solomon codes. *Communications of the ACM*, 24(9):583–584, 1981.

43 Claudio Orlandi, Divya Ravi, and Peter Scholl. On the bottleneck complexity of mpc with correlated randomness. In *Public-Key Cryptography – PKC 2022, Part I*, pages 194–220, 2022.

44 T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, pages 73–85, 1989.

45 Victor Shoup. *A computational introduction to number theory and algebra*. Cambridge university press, 2009.

46 Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology – EUROCRYPT 2010*, pages 24–43, 2010.

47 H. Yamamoto. Secret sharing system using (k, L, n) threshold scheme. *Electronics and Communications in Japan (Part I: Communications)*, 69(9):46–54, 1986.

48 Andrew C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 160–164, 1982.

A Proof of Proposition 12

Functionality $\mathcal{F}_{\mathsf{LT}}(\mathbf{M}; (v_i)_{i\in[n]})$

1. Players have shares $(v_i)_{i\in[n]}$ of RSS_k for a secret $\mathbf{s} = (s_0, \ldots, s_{k-1})$.
2. $\mathcal{F}_{\mathsf{LT}}$ receives $v_i \in \mathbb{K}$ from each player P_i.
3. $\mathcal{F}_{\mathsf{LT}}$ reconstructs $s_j = \sum_{i\in[n]} \mathsf{Reconst}_k(j, i; v_i)$ for all $j = 0, 1, \ldots, k-1$, and computes $\mathbf{u} = \mathbf{M} \cdot \mathbf{s} \in \mathbb{K}^\ell$.
4. $\mathcal{F}_{\mathsf{LT}}$ computes shares $(w_i)_{i\in[n]} \leftarrow \mathsf{RSS}_\ell(\mathbf{u})$ and gives w_i to each player P_i.

Protocol Π_{LT}

Input. Each player P_i has the i-th share $v_i \in \mathbb{K}$ of RSS_k for a secret $\mathbf{s} = (s_0, \ldots, s_{k-1})$.
Output. Each player P_i obtains w_i, where $(w_i)_{i\in[n]} \leftarrow \mathcal{F}_{\mathsf{LT}}(\mathbf{M}; (v_i)_{i\in[n]})$.
Setup.

1. Let $\mathbf{r} \leftarrow_{\$} \mathbb{K}^\ell$.
2. Let $(a_i)_{i\in[n]} \leftarrow \mathsf{RSS}_\ell(\mathbf{r})$ and $(b_i)_{i\in[n]} \leftarrow \mathsf{RSS}_\ell(\mathbf{0}_\ell)$.
3. Each player P_i receives (a_i, b_i).

Protocol.

1. Each player P_i computes
$$\mathbf{x}_i = \mathbf{M} \cdot \begin{pmatrix} \mathsf{Reconst}_k(0, i; v_i) \\ \vdots \\ \mathsf{Reconst}_k(k-1, i; v_i) \end{pmatrix} - \begin{pmatrix} \mathsf{Reconst}_\ell(0, i; a_i) \\ \vdots \\ \mathsf{Reconst}_\ell(\ell-1, i; a_i) \end{pmatrix}$$
2. Players obtain $\mathbf{y} = \mathcal{F}_{\mathsf{Sum}}((\mathbf{x}_i)_{i\in[n]})$, where $\mathcal{F}_{\mathsf{Sum}}$ is invoked in an element-wise way.
3. Each player P_i computes $w'_i = \mathsf{FixedShare}_\ell(i, \mathbf{y})$.
4. Each player P_i outputs $w_i = w'_i + a_i + b_i$.

Figure 4 The functionality $\mathcal{F}_{\mathsf{LT}}$ and a protocol Π_{LT} implementing it.

Recall that α_i (resp. β_j) is the point associated with the i-th share (resp. the j-th component of a secret vector) of RSS_ℓ and RSS_k. To simplify notations, we denote $(\varphi(\alpha_i))_{i\in T}$ by $\varphi(\alpha_T)$ for a set $T \subseteq [n]$ and a polynomial φ.

Let T be a subset of size at most t and $(v_i)_{i\in[n]}$ be an input to the protocol $\Pi = \Pi_{\mathsf{LT}}$. Let $\mathbf{s}$ be the secret of RSS_k determined by $(v_i)_{i\in[n]}$ and set $\mathbf{u} = \mathbf{M}\cdot\mathbf{s}$.

Consider the real process. Observe that the a_i's and b_i's can be written as $a_i = A(\alpha_i)$ and $b_i = B(\alpha_i)$ for random polynomials $A \leftarrow_\$ \mathcal{R}_{\mathbf{r}}$ and $B \leftarrow_\$ \mathcal{R}_{\mathbf{0}_\ell}$. Also, it holds that

$$\begin{aligned}
\mathbf{y} &= \sum_{i\in[n]} \mathbf{x}_i \\
&= \mathbf{M}\cdot\sum_{i\in[n]}\begin{pmatrix}\mathsf{Reconst}_k(0,i;v_i)\\ \vdots \\ \mathsf{Reconst}_k(k-1,i;v_i)\end{pmatrix} - \sum_{i\in[n]}\begin{pmatrix}\mathsf{Reconst}_\ell(0,i;a_i)\\ \vdots \\ \mathsf{Reconst}_\ell(\ell-1,i;a_i)\end{pmatrix} \\
&= \mathbf{M}\cdot\mathbf{s} - \mathbf{r} \\
&= \mathbf{u} - \mathbf{r}.
\end{aligned}$$

Furthermore, for all $i \in [n]$,

$$w_i = \psi_{\mathbf{y}}(\alpha_i) + A(\alpha_i) + B(\alpha_i),$$

where $\psi_{\mathbf{y}} \in \mathcal{R}_{\mathbf{y}}$ is the polynomial computed by the deterministic algorithm $\mathsf{FixedShare}_\ell$. Thus, the output of the real process in the $\mathcal{F}_{\mathsf{Sum}}$-hybrid model is

$$\begin{aligned}
\mathsf{Real}_\Pi(T,(v_i)_{i\in[n]}) &= ((\mathsf{View}_{\Pi,i}((v_i)_{i\in[n]}))_{i\in T}; (\mathsf{Output}_{\Pi,i}((v_i)_{i\in[n]}))_{i\in[n]}) \\
&= ((v_i)_{i\in T}, A(\alpha_T), B(\alpha_T), \mathbf{y}; (\psi_{\mathbf{y}} + A + B)(\alpha_{[n]})),
\end{aligned}$$

where $\mathbf{r} \leftarrow_\$ \mathbb{K}^\ell$, $A \leftarrow_\$ \mathcal{R}_{\mathbf{r}}$, $\mathbf{y} = \mathbf{u} - \mathbf{r}$, and $B \leftarrow_\$ \mathcal{R}_{\mathbf{0}_\ell}$. Here, we omit $\mathbf{x}_i$ and w'_i from the view of corrupted players since they are locally computed by the other elements.

Since $t \le n-\ell$, Lemma 3 ensures that there exists a polynomial $\Delta_{\mathbf{r}} \in \mathcal{R}_{\mathbf{r}}$ such that $\Delta_{\mathbf{r}}(\alpha_i) = 0$ for all $i \in T$. If we set $A' = A - \Delta_{\mathbf{r}}$, then A' is uniformly distributed over $\mathcal{R}_{\mathbf{0}_\ell}$ and $A'(\alpha_i) = A(\alpha_i)$ for all $i \in T$ from Lemma 4. Thus, we have that

$$\mathsf{Real}_\Pi(T,(v_i)_{i\in[n]}) = ((v_i)_{i\in T}, A'(\alpha_T), B(\alpha_T), \mathbf{y}; (\psi_{\mathbf{y}} + A' + \Delta_{\mathbf{r}} + B)(\alpha_{[n]})),$$

where $\mathbf{r} \leftarrow_\$ \mathbb{K}^\ell$, $\mathbf{y} = \mathbf{u} - \mathbf{r}$, and $A', B \leftarrow_\$ \mathcal{R}_{\mathbf{0}_\ell}$. Since $\mathbf{u} - \mathbf{r}$ is uniformly distributed over $\mathbb{K}^\ell$, we have that

$$\mathsf{Real}_\Pi(T,(v_i)_{i\in[n]}) = ((v_i)_{i\in T}, A'(\alpha_T), B(\alpha_T), \mathbf{y}'; (\psi_{\mathbf{y}'} + A' + \Delta_{\mathbf{u}-\mathbf{y}'} + B)(\alpha_{[n]})),$$

where $\mathbf{y}' \leftarrow_\$ \mathbb{K}^\ell$ and $A', B \leftarrow_\$ \mathcal{R}_{\mathbf{0}_\ell}$. Since $\psi_{\mathbf{y}'} \in \mathcal{R}_{\mathbf{y}'}$, $A' \in \mathcal{R}_{\mathbf{0}_k}$ and $\Delta_{\mathbf{u}-\mathbf{y}'} \in \mathcal{R}_{\mathbf{u}-\mathbf{y}'}$, it holds that $\psi_{\mathbf{y}'} + A' + \Delta_{\mathbf{u}-\mathbf{y}'} \in \mathcal{R}_{\mathbf{u}}$. If we set $\phi' := \psi_{\mathbf{y}'} + A' + \Delta_{\mathbf{u}-\mathbf{y}'} + B$, then ϕ' is uniformly distributed over $\mathcal{R}_{\mathbf{u}}$ from Lemma 4. Since $\Delta_{\mathbf{u}-\mathbf{y}'}(\alpha_i) = 0$ for all $i \in T$, we have that

$$\begin{aligned}
\mathsf{Real}_\Pi(T,(v_i)_{i\in[n]}) &= ((v_i)_{i\in T}, A'(\alpha_T), (\phi' - \psi_{\mathbf{y}'} - A' - \Delta_{\mathbf{u}-\mathbf{y}'})(\alpha_T), \mathbf{y}'; \phi'(\alpha_{[n]})) \\
&= ((v_i)_{i\in T}, A'(\alpha_T), (\phi' - \psi_{\mathbf{y}'} - A')(\alpha_T), \mathbf{y}'; \phi'(\alpha_{[n]})),
\end{aligned}$$

where $\mathbf{y}' \leftarrow_\$ \mathbb{K}^\ell$, $A' \leftarrow_\$ \mathcal{R}_{\mathbf{0}_\ell}$ and $\phi' \leftarrow_\$ \mathcal{R}_{\mathbf{u}}$.

On the other hand, we define a simulator $\mathsf{Sim}(T,(v_i)_{i\in T},(w_i)_{i\in T})$ as follows: First, it samples $\widetilde{\mathbf{y}} \leftarrow_\$ \mathbb{K}^\ell$ and $\widetilde{A} \leftarrow_\$ \mathcal{R}_{\mathbf{0}_\ell}$, and sets $\widetilde{a}_i = \widetilde{A}(\alpha_i)$ and $\widetilde{b}_i = w_i - \psi_{\widetilde{\mathbf{y}}}(\alpha_i) - \widetilde{A}(\alpha_i)$ for $i \in T$. Then, it outputs

$$\mathsf{Sim}(T,(v_i)_{i\in T},(w_i)_{i\in T}) = ((v_i)_{i\in T}, (\widetilde{a}_i)_{i\in T}, (\widetilde{b}_i)_{i\in T}, \widetilde{\mathbf{y}}).$$

Note that the functionality $\mathcal{F} = \mathcal{F}_{\mathsf{LT}}$ gives players fresh shares of RSS_ℓ for a secret $\mathbf{u}$. Formally, the i-th player P_i receives $\phi(\alpha_i)$, where $\phi \leftarrow_\$ \mathcal{R}_{\mathbf{u}}$. Then, the output of the ideal process with respect to the functionality $\mathcal{F}$ and the simulator Sim is

$$\mathsf{Ideal}_{\mathcal{F},\mathsf{Sim}}(T, (v_i)_{i\in[n]}) = (\mathsf{Sim}(T, (v_i)_{i\in T}, \widetilde{\phi}(\alpha_T)); \widetilde{\phi}(\alpha_{[n]})),$$

where $\widetilde{\phi} \leftarrow_\$ \mathcal{R}_{\mathbf{u}}$. From the construction of Sim, we have that

$$\mathsf{Ideal}_{\mathcal{F},\mathsf{Sim}}(T, (v_i)_{i\in[n]}) = ((v_i)_{i\in T}, \widetilde{A}(\alpha_T), (\widetilde{\phi} - \psi_{\widetilde{\mathbf{y}}} - \widetilde{A})(\alpha_T), \widetilde{\mathbf{y}}; \widetilde{\phi}(\alpha_{[n]})),$$

where $\widetilde{\mathbf{y}} \leftarrow_\$ \mathbb{K}^\ell$, $\widetilde{A} \leftarrow_\$ \mathcal{R}_{\mathbf{0}_\ell}$, and $\widetilde{\phi} \leftarrow_\$ \mathcal{R}_{\mathbf{u}}$.

Therefore, we conclude that

$$\mathsf{Ideal}_{\mathcal{F},\mathsf{Sim}}(T, (v_i)_{i\in[n]}) = \mathsf{Real}_\Pi(T, (v_i)_{i\in[n]}).$$

Since players receives two shares of RSS_ℓ, the size of correlated randomness is $\mathrm{CR}(\Pi_{\mathsf{LT}}) = O(\log|\mathbb{K}|) = O(\log n)$. In the online phase, the protocol invokes $\mathcal{F}_{\mathsf{Sum}}$ ℓ times and hence we have $\mathrm{BC}(\Pi_{\mathsf{LT}}) = O(\ell \log|\mathbb{K}|) = O(\ell \log n)$.

B Proof of Proposition 13

Let $\mathbf{x} = (x_0, \dots, x_{\ell-1})$ (resp. $\mathbf{y} = (y_0, \dots, y_{\ell-1})$) be the secret determined by shares $(v_i)_{i\in[n]}$ (resp. $(w_i)_{i\in[n]}$).

First, we see the correctness of Π_{IP}. The linearity of RSS_ℓ implies that at Step 1 of the protocol, $(v'_i)_{i\in[n]}$ (resp. $(w'_i)_{i\in[n]}$) are shares of a secret $\mathbf{x} - \mathbf{a}$ (resp. $\mathbf{y} - \mathbf{b}$). We thus have that $\mathbf{x}' = \mathbf{x} - \mathbf{a}$, $\mathbf{y}' = \mathbf{y} - \mathbf{b}$ and $\mathbf{z}' = (\mathbf{x} - \mathbf{a}) * (\mathbf{y} - \mathbf{b})$ at Steps 3 and 4. On the other hand, the functionality of $\mathcal{F}_{\mathsf{LT}}$ ensures that $(a'_i)_{i\in[n]}$ are shares of a secret

$$\mathbf{a}' := \mathrm{diag}(\mathbf{y}') \cdot \mathbf{a} = (\mathbf{y} - \mathbf{b}) * \mathbf{a}$$

and similarly, $(b'_i)_{i\in[n]}$ are shares of a secret $\mathbf{b}' := (\mathbf{x} - \mathbf{a}) * \mathbf{b}$. The linearity of RSS_ℓ implies that $d_i = z'_i + a'_i + b'_i + c_i + r_i$ is the i-th share of a secret

$$\begin{aligned}\mathbf{z}' + \mathbf{a}' + \mathbf{b}' + \mathbf{c} + \mathbf{s} &= (\mathbf{x} - \mathbf{a}) * (\mathbf{y} - \mathbf{b}) + (\mathbf{y} - \mathbf{b}) * \mathbf{a} + (\mathbf{x} - \mathbf{a}) * \mathbf{b} + \mathbf{a} * \mathbf{b} + \mathbf{s} \\ &= \mathbf{x} * \mathbf{y} + \mathbf{s}.\end{aligned}$$

Thus it holds that $\mathbf{d} = \mathbf{x} * \mathbf{y} + \mathbf{s}$. The correctness follows from

$$z = \langle \mathbf{1}_\ell, \mathbf{d} \rangle = \langle \mathbf{1}_\ell, \mathbf{x} * \mathbf{y} \rangle + \langle \mathbf{1}_\ell, \mathbf{s} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle.$$

We show the privacy of Π_{IP}. Let $T \subsetneq [n]$ be the set of corrupted players. Recall that α_i (resp. β_j) is the point associated with the i-th share (resp. the j-th component of a secret vector) of RSS_ℓ. To simplify notations, we denote $(\varphi(\alpha_i))_{i\in T}$ by $\varphi(\alpha_T)$ for a polynomial φ. In the $\mathcal{F}_{\mathsf{Sum}}$-hybrid model, corrupted players' view at Steps 3 and 7 (including their correlated randomness for $\mathcal{F}_{\mathsf{Sum}}$) only contains their inputs $(\mathbf{x}'_i, \mathbf{y}'_i, \mathbf{d}_i)_{i\in T}$ to $\mathcal{F}_{\mathsf{Sum}}$ and the outputs $\mathbf{x}', \mathbf{y}', \mathbf{d}$. Also, in the $\mathcal{F}_{\mathsf{LT}}$-hybrid model, corrupted players' view at Step 5 (including their correlated randomness for $\mathcal{F}_{\mathsf{LT}}$) only contains their inputs $(a_i, b_i)_{i\in T}$ to $\mathcal{F}_{\mathsf{LT}}$ and the outputs $(a'_i, b'_i)_{i\in T}$. It is therefore sufficient to show that the joint distribution of the following elements is simulated from $(v_i, w_i)_{i\in T}$ and $z = \mathcal{F}_{\mathsf{IP}}((v_i, w_i)_{i\in[n]})$ since the other elements are locally computed from them:

Correlated randomness. $(a_i, b_i, c_i, r_i)_{i\in T}$;

Functionality $\mathcal{F}_{\mathsf{IP}}((v_i, w_i)_{i\in[n]})$

1. Players have shares $(v_i)_{i\in[n]}$ and $(w_i)_{i\in[n]}$ of RSS_ℓ for secrets $\mathbf{x} = (x_0, \ldots, x_{\ell-1})$ and $\mathbf{y} = (y_0, \ldots, y_{\ell-1})$, respectively.
2. $\mathcal{F}_{\mathsf{IP}}$ receives shares $v_i, w_i \in \mathbb{K}$ from each player P_i.
3. $\mathcal{F}_{\mathsf{IP}}$ reconstructs

   $$x_j = \sum_{i\in[n]} \mathsf{Reconst}_\ell(j, i; v_i), \ \ y_j = \sum_{i\in[n]} \mathsf{Reconst}_\ell(j, i; w_i)$$

   for all $j = 0, 1, \ldots, \ell - 1$, and computes $z = \langle \mathbf{x}, \mathbf{y} \rangle$.
4. $\mathcal{F}_{\mathsf{IP}}$ gives z to every player P_i.

Protocol Π_{IP}

Input. Each player P_i has the i-th shares $v_i, w_i \in \mathbb{K}$ of RSS_ℓ for secrets $\mathbf{x} = (x_0, \ldots, x_{\ell-1})$ and $\mathbf{y} = (y_0, \ldots, y_{\ell-1})$, respectively.
Output. Each player P_i obtains $z = \mathcal{F}_{\mathsf{IP}}((v_i, w_i)_{i\in[n]})$.
Setup.
1. Let $\mathbf{a}, \mathbf{b} \leftarrow_{\$} \mathbb{K}^\ell$ and $\mathbf{c} = \mathbf{a} * \mathbf{b}$, where $*$ is the element-wise multiplication.
2. Let $(a_i)_{i\in[n]} \leftarrow \mathsf{RSS}_\ell(\mathbf{a})$, $(b_i)_{i\in[n]} \leftarrow \mathsf{RSS}_\ell(\mathbf{b})$ and $(c_i)_{i\in[n]} \leftarrow \mathsf{RSS}_\ell(\mathbf{c})$.
3. Choose a random vector $\mathbf{s} \in \mathbb{K}^\ell$ such that $\langle \mathbf{1}_\ell, \mathbf{s} \rangle = 0$.
4. Let $(r_i)_{i\in[n]} \leftarrow \mathsf{RSS}_\ell(\mathbf{s})$.
5. Each player P_i receives (a_i, b_i, c_i, r_i).

Protocol.
1. Each player P_i computes $v_i' = v_i - a_i$ and $w_i' = w_i - b_i$.
2. Each player P_i computes

   $$\mathbf{x}_i' = (\mathsf{Reconst}_\ell(0, i; v_i'), \ldots, \mathsf{Reconst}_\ell(\ell - 1, i; v_i')),$$
   $$\mathbf{y}_i' = (\mathsf{Reconst}_\ell(0, i; w_i'), \ldots, \mathsf{Reconst}_\ell(\ell - 1, i; w_i')).$$

3. Players obtain $\mathbf{x}' = \mathcal{F}_{\mathsf{Sum}}((\mathbf{x}_i')_{i\in[n]})$ and $\mathbf{y}' = \mathcal{F}_{\mathsf{Sum}}((\mathbf{y}_i')_{i\in[n]})$, where $\mathcal{F}_{\mathsf{Sum}}$ is invoked in an element-wise way.
4. Each player P_i computes $\mathbf{z}' = \mathbf{x}' * \mathbf{y}'$ and $z_i' = \mathsf{FixedShare}_\ell(i, \mathbf{z}')$.
5. Players obtain

   $$(a_i')_{i\in[n]} \leftarrow \mathcal{F}_{\mathsf{LT}}(\mathbf{N}; (a_i)_{i\in[n]}), \ \ (b_i')_{i\in[n]} \leftarrow \mathcal{F}_{\mathsf{LT}}(\mathbf{M}; (b_i)_{i\in[n]}),$$

   where $\mathbf{M} = \mathrm{diag}(\mathbf{x}')$ and $\mathbf{N} = \mathrm{diag}(\mathbf{y}')$.
6. Each player P_i computes $d_i = z_i' + a_i' + b_i' + c_i + r_i$ and

   $$\mathbf{d}_i = (\mathsf{Reconst}_\ell(0, i; d_i), \ldots, \mathsf{Reconst}_\ell(\ell - 1, i; d_i)).$$

7. Players obtain $\mathbf{d} = \mathcal{F}_{\mathsf{Sum}}((\mathbf{d}_i)_{i\in[n]})$.
8. Every player outputs $z = \langle \mathbf{1}_\ell, \mathbf{d} \rangle$.

Figure 5 The functionality $\mathcal{F}_{\mathsf{IP}}$ and a protocol Π_{IP} implementing it.

Online messages. $\mathbf{x}' = \mathbf{x} - \mathbf{a}$, $\mathbf{y}' = \mathbf{y} - \mathbf{b}$, $(a'_i, b'_i)_{i \in T}$ and $\mathbf{d} = \mathbf{x} * \mathbf{y} + \mathbf{s}$.
To analyze the distribution of the above elements, we define

$$\mathsf{View} = ((a_i, b_i, c_i, r_i, a'_i, b'_i)_{i \in T}, \mathbf{x}', \mathbf{y}', \mathbf{d}).$$

Observe that the distribution of View is given by

$$\mathsf{View} = (\phi_a(\alpha_T), \phi_b(\alpha_T), \phi_c(\alpha_T), \phi_s(\alpha_T), \phi_{a'}(\alpha_T), \phi_{b'}(\alpha_T), \mathbf{x} - \mathbf{a}, \mathbf{y} - \mathbf{b}, \mathbf{x} * \mathbf{y} + \mathbf{s}),$$

where

$$\mathbf{a}, \mathbf{b} \leftarrow_{\$} \mathbb{K}^\ell,\ \mathbf{s} \leftarrow_{\$} V_0 := \{\mathbf{s} \in \mathbb{K}^\ell : \langle \mathbf{1}_\ell, \mathbf{s} \rangle = 0\},\ \phi_a \leftarrow_{\$} \mathcal{R}_{\mathbf{a}},\ \phi_b \leftarrow_{\$} \mathcal{R}_{\mathbf{b}},$$
$$\phi_c \leftarrow_{\$} \mathcal{R}_{\mathbf{a}*\mathbf{b}},\ \phi_s \leftarrow_{\$} \mathcal{R}_{\mathbf{s}},\ \phi_{a'} \leftarrow_{\$} \mathcal{R}_{(\mathbf{y}-\mathbf{b})*\mathbf{a}},\ \phi_{b'} \leftarrow_{\$} \mathcal{R}_{(\mathbf{x}-\mathbf{a})*\mathbf{b}}.$$

Lemma 3 ensures that for any $\mathbf{v} \in \mathbb{K}^\ell$, there is a polynomial $\Delta_{\mathbf{v}} \in \mathcal{R}_{\mathbf{v}}$ such that $\Delta_{\mathbf{v}}(\alpha_i) = 0$ for all $i \in T$. If $\widetilde{\phi}_a$ is uniformly distributed over $\mathcal{R}_{\mathbf{0}_\ell}$, then $\widetilde{\phi}_a + \Delta_{\mathbf{a}}$ is uniformly distributed over $\mathcal{R}_{\mathbf{a}}$ from Lemma 4 and $(\widetilde{\phi}_a + \Delta_{\mathbf{a}})(\alpha_i) = \widetilde{\phi}_a(\alpha_i)$ for all $i \in T$. Similarly, let $\widetilde{\phi}_b, \widetilde{\phi}_c, \widetilde{\phi}_s, \widetilde{\phi}_{a'}, \widetilde{\phi}_{b'} \leftarrow_{\$} \mathcal{R}_{\mathbf{0}_\ell}$, and then it holds that

$$\widetilde{\phi}_b + \Delta_{\mathbf{b}} \leftarrow_{\$} \mathcal{R}_{\mathbf{b}},\ \widetilde{\phi}_c + \Delta_{\mathbf{a}*\mathbf{b}} \leftarrow_{\$} \mathcal{R}_{\mathbf{a}*\mathbf{b}},\ \widetilde{\phi}_s + \Delta_{\mathbf{s}} \leftarrow_{\$} \mathcal{R}_{\mathbf{s}},$$
$$\widetilde{\phi}_{a'} + \Delta_{(\mathbf{y}-\mathbf{b})*\mathbf{a}} \leftarrow_{\$} \mathcal{R}_{(\mathbf{y}-\mathbf{b})*\mathbf{a}},\ \widetilde{\phi}_{b'} + \Delta_{(\mathbf{x}-\mathbf{a})*\mathbf{b}} \leftarrow_{\$} \mathcal{R}_{(\mathbf{x}-\mathbf{a})*\mathbf{b}}.$$

It also holds that

$$\left(\widetilde{\phi}_b + \Delta_{\mathbf{b}}\right)(\alpha_i) = \widetilde{\phi}_b(\alpha_i),\ \left(\widetilde{\phi}_c + \Delta_{\mathbf{a}*\mathbf{b}}\right)(\alpha_i) = \widetilde{\phi}_c(\alpha_i),\ \left(\widetilde{\phi}_s + \Delta_{\mathbf{s}}\right)(\alpha_i) = \widetilde{\phi}_s(\alpha_i),$$
$$\left(\widetilde{\phi}_{a'} + \Delta_{(\mathbf{y}-\mathbf{b})*\mathbf{a}}\right)(\alpha_i) = \widetilde{\phi}_{a'}(\alpha_i),\ \left(\widetilde{\phi}_{b'} + \Delta_{(\mathbf{x}-\mathbf{a})*\mathbf{b}}\right)(\alpha_i) = \widetilde{\phi}_{b'}(\alpha_i)$$

for all $i \in T$. We thus have that

$$\mathsf{View} = (\widetilde{\phi}_a(\alpha_T), \widetilde{\phi}_b(\alpha_T), \widetilde{\phi}_c(\alpha_T), \widetilde{\phi}_s(\alpha_T), \widetilde{\phi}_{a'}(\alpha_T), \widetilde{\phi}_{b'}(\alpha_T), \mathbf{x} - \mathbf{a}, \mathbf{y} - \mathbf{b}, \mathbf{x} * \mathbf{y} + \mathbf{s}),$$

where $\mathbf{a}, \mathbf{b} \leftarrow_{\$} \mathbb{K}^\ell$, $\mathbf{s} \leftarrow_{\$} V_0$, and $\widetilde{\phi}_b, \widetilde{\phi}_c, \widetilde{\phi}_s, \widetilde{\phi}_{a'}, \widetilde{\phi}_{b'} \leftarrow_{\$} \mathcal{R}_{\mathbf{0}_\ell}$. Since $\widetilde{\mathbf{a}} := \mathbf{x} - \mathbf{a}$ and $\widetilde{\mathbf{b}} := \mathbf{y} - \mathbf{b}$ are uniformly distributed over $\mathbb{K}^\ell$, we have that

$$\mathsf{View} = (\widetilde{\phi}_a(\alpha_T), \widetilde{\phi}_b(\alpha_T), \widetilde{\phi}_c(\alpha_T), \widetilde{\phi}_s(\alpha_T), \widetilde{\phi}_{a'}(\alpha_T), \widetilde{\phi}_{b'}(\alpha_T), \widetilde{\mathbf{a}}, \widetilde{\mathbf{b}}, \mathbf{x} * \mathbf{y} + \mathbf{s}),$$

where $\widetilde{\mathbf{a}}, \widetilde{\mathbf{b}} \leftarrow_{\$} \mathbb{K}^\ell$, $\mathbf{s} \leftarrow_{\$} V_0$, and $\widetilde{\phi}_b, \widetilde{\phi}_c, \widetilde{\phi}_s, \widetilde{\phi}_{a'}, \widetilde{\phi}_{b'} \leftarrow_{\$} \mathcal{R}_{\mathbf{0}_\ell}$. Since $z = \mathcal{F}_{\mathsf{IP}}((v_i, w_i)_{i \in [n]}) = \langle \mathbf{x}, \mathbf{y} \rangle$, it holds that $\langle \mathbf{1}_\ell, \mathbf{x} * \mathbf{y} - z \cdot \mathbf{e}_0 \rangle = \langle \mathbf{x}, \mathbf{y} \rangle - z = 0$. and hence $\mathbf{s}_0 := \mathbf{x} * \mathbf{y} - z \cdot \mathbf{e}_0 \in V_0$, where $\mathbf{e}_0 = (1, 0, \ldots, 0) \in \mathbb{K}^\ell$. Furthermore, since V_0 is a linear space, if $\mathbf{s}$ is uniformly distributed over V_0, then so is $\mathbf{s} + \mathbf{s}_0$. In particular, if $\mathbf{s}, \widetilde{\mathbf{s}} \leftarrow_{\$} V_0$, then $\mathbf{x} * \mathbf{y} + \mathbf{s}$ and $z \cdot \mathbf{e}_0 + \widetilde{\mathbf{s}}$ follow the same distribution. We then have that

$$\mathsf{View} = (\widetilde{\phi}_a(\alpha_T), \widetilde{\phi}_b(\alpha_T), \widetilde{\phi}_c(\alpha_T), \widetilde{\phi}_s(\alpha_T), \widetilde{\phi}_{a'}(\alpha_T), \widetilde{\phi}_{b'}(\alpha_T), \widetilde{\mathbf{a}}, \widetilde{\mathbf{b}}, z \cdot \mathbf{e}_0 + \widetilde{\mathbf{s}}),$$

where $\widetilde{\mathbf{a}}, \widetilde{\mathbf{b}} \leftarrow_{\$} \mathbb{K}^\ell$, $\widetilde{\mathbf{s}} \leftarrow_{\$} V_0$, and $\widetilde{\phi}_a, \widetilde{\phi}_b, \widetilde{\phi}_c, \widetilde{\phi}_{a'}, \widetilde{\phi}_{b'}, \widetilde{\phi}_s \leftarrow_{\$} \mathcal{R}_{\mathbf{0}_\ell}$. Therefore, we conclude that View is simulated from z only.

Since players receive four shares of RSS_ℓ and correlated randomness for two invocations of $\mathcal{F}_{\mathsf{LT}}$, we have $\mathrm{CR}(\Pi_{\mathsf{IP}}) = O(\log |\mathbb{K}|) = O(\log n)$. In the online phase, the protocol invokes $\mathcal{F}_{\mathsf{Sum}}$ three times and $\mathcal{F}_{\mathsf{LT}}$ twice, and hence we have $\mathrm{BC}(\Pi_{\mathsf{IP}}) = O(\ell \log |\mathbb{K}|) = O(\ell \log n)$.

Fast Secure Computations on Shared Polynomials and Applications to Private Set Operations

Pascal Giorgi
LIRMM, Univ. Montpellier, CNRS, France

Fabien Laguillaumie
LIRMM, Univ. Montpellier, CNRS,, France

Lucas Ottow
LIRMM, Univ. Montpellier, CNRS, France

Damien Vergnaud
LIP6, Sorbonne University, CNRS, France

Abstract

Secure multi-party computation aims to allow a set of players to compute a given function on their secret inputs without revealing any other information than the result of the computation. In this work, we focus on the design of secure multi-party protocols for shared polynomial operations. We consider the classical model where the adversary is honest-but-curious, and where the coefficients (or any secret values) are either encrypted using an additively homomorphic encryption scheme or shared using a threshold linear secret-sharing scheme. Our protocols terminate after a constant number of rounds and minimize the number of secure multiplications.

In their seminal article at PKC 2006, Mohassel and Franklin proposed constant-rounds protocols for the main operations on (shared) polynomials. In this work, we improve the *fan-in multiplication of nonzero polynomials*, the *multi-point polynomial evaluation* and the *polynomial interpolation* (on secret points) to reach a quasi-linear complexity (instead of quadratic in Mohassel and Franklin's work) in the degree of shared input/output polynomials.

Computing with shared polynomials is a core component of several multi-party protocols for privacy-preserving operations on private sets, like the *private disjointness test* or the *private set intersection*. Using our new protocols, we are able to improve the complexity of such protocols and to design the first variants which always return a correct result.

2012 ACM Subject Classification Theory of computation → Cryptographic protocols; Security and privacy → Information-theoretic techniques

Keywords and phrases Multi-party computation, polynomial operations, privacy-preserving set operations

Digital Object Identifier 10.4230/LIPIcs.ITC.2024.11

Related Version *Full Version*: `https://eprint.iacr.org/2024/470`

Funding This work was supported by the France 2030 ANR project SecureCompute ANR-22-PECY-0003, the French ANR SANGRIA project ANR-21-CE39-0006, and the French project CRYPTANALYSE ANR-22-PECY-0010.

Acknowledgements We would like to thank the anonymous reviewers for their helpful comments.

1 Introduction

Secure multi-party computation (MPC), which dates back to fundamental works by Yao [36] and Goldreich, Micali, and Widgerson [19], is a family of cryptographic techniques that enables parties to jointly compute a function over their private inputs while keeping those inputs confidential. This approach ensures that none of the participating parties need to reveal any information on their data to one another, yet they can still obtain the desired computation result. Unconditionally secure MPC protocols were first proposed by Ben-Or, Goldwasser, and Widgerson [2] and Chaum, Crépeau and Damgård [5]. On the other hand, several computationally secure protocols have been proposed, relying on many different techniques, like verifiable secret sharing [8, 15] or linearly homomorphic encryption [12, 9].

In the realm of computer algebra, polynomial evaluation and interpolation algorithms stand out as versatile tools with applications spanning numerous domains [14]. In 2006, Mohassel and Franklin [27] proposed several secure MPC protocols for various polynomial operations where the polynomial coefficients are private inputs of the parties. It is the main goal of this paper to improve some of their protocols and to illustrate their usability *via* several applications to private set operations protocols. All of the presented protocols are considered in the *honest-but-curious* model.

1.1 Secure computation on shared data

The computation on shared elements dates back to the works of Ben-Or, Goldwasser, and Widgerson [2] and Bar-Ilan and Beaver [1]. In the general setting, it involves m players that want to compute a function (seen as an arithmetic circuit) over secret inputs (that we will consider as elements of $\mathbb{F}_q$). These secret inputs are assumed to be shared between the parties before any computation. This can done either via a secret-sharing scheme (such as an additive secret-sharing scheme or Shamir's secret-sharing scheme [35], in the information theoretical model) or via a threshold linearly homomorphic scheme (such as a threshold variant of Paillier's encryption [32] or CL encryption [3], in the computational model). Parties can compute any arithmetic circuit on the shared inputs by combining several basic operations on the shared data (additions and multiplications). While the work in [2, 1] addresses the general problem of computing any circuit, many other results have been provided to improve the efficiency on specific problems [7, 27, 11, 10, 28]. Our work follows the latter line of research without relying on any specific underlying secret-sharing scheme or any threshold homomorphic encryption scheme. One of our main objectives is to derive protocols in a general framework, encompassing both theoretical and computational models, that achieve a constant number of rounds of communication between each party while minimizing the total amount of exchanged data.

Let us denote by $[x]$ an element $x \in \mathbb{F}_q$ that is shared among the parties. In order to simplify the presentation, we will often assume that our protocols are implemented using a secret-sharing scheme. Therefore, we will note by $[x]_j$ the part of the secret belonging to the j-th player. Our framework assumes that the players are able to do elementary operations, such as *addition of two shares*, *scalar multiplication with public value*, and *multiplication of two shares*, in a constant number of rounds.

In the information-theoretic model, the first two operations do not require any communication as each player j can just compute individually: $[a+b]_j = [a]_j + [b]_j$ and $\lambda[a]_j$ to get a share of the results $a+b$ and λa. Computing the product $[ab]$ from the shares of $[a], [b]$ is trickier, and cannot be done locally. Several interactive solutions exist: the BGW protocol [2], or Beaver triples [1] offer alternatives which can be both done in a constant number of rounds.

In the case of threshold linearly homomorphic schemes, the first two operations can also be done locally on encrypted data, and solutions exist to compute the product of two encrypted field elements. Finally, our framework further requires that the sharing method allows for secure constant-round methods to share a constant element (for example to share the values 1 or 0) and to share a uniformly random (unknown) element.

The complexity measure of our multi-party protocols will be given as the number of "secure multiplication" in the base field $\mathbb{F}_q$, *i.e.* multiplication of two shared elements. Since it is the only operation requiring communication between players, this will express the communication complexity of our protocols. To guarantee that our protocols still run in constant-round, we will extensively use parallel executions of constant-round protocols.

We are considering the *honest-but-curious* model. Therefore, privacy is only a matter of checking that when a shared value is revealed, the revealed value is uniformly random and independent of the value of the secret inputs. We achieve security for any of the protocols presented in this paper as long as the elementary operations presented above (*i.e.* addition and multiplications on shared elements of $\mathbb{F}_q$) can be composed in parallel and remain secure. This follows a long line of work which are based on the same assumptions in secure linear algebra [1, 7, 31, 24, 10, 28] or in secure polynomial computation [27].

1.2 Toolbox for Secure Polynomial Computation

In their seminal paper, Mohassel and Franklin [27] proposed the very first protocols for secure polynomial multiplication, division with remainder, and polynomial interpolation. They considered the scenario in which the coefficients of the involved polynomials, evaluation points, or values are shared. For a polynomial $f = \sum_{k=0}^{d-1} f_k X^k$, we denote by $[f]$ a sharing of the polynomial, that is a collection of sharings of its coefficients $[f_0], \dots, [f_{d-1}]$.

The goal of Mohassel and Franklin was to propose efficient protocols with good communication and round complexities. For example, in the case of polynomial interpolation, assuming that the m parties hold shares (or ciphertexts) of $n \geq 1$ pairs $([x_i], [y_i]) \in \mathbb{F}_q^2$ for $i \in \{1, \dots, n\}$ (with $x_i \neq x_j$ for $i \neq j$), they proposed a protocol with a constant number of rounds and communication complexity of $\mathcal{O}(n^2)$ multiplications. Note that this protocol has remained the most efficient since 2006.

As a first contribution, we present new protocols for efficient and secure operations on shared polynomials. Our model is identical to the one used by Mohassel and Franklin. In particular, our proposals can be implemented using a threshold linearly homomorphic encryption scheme (and in this case achieve semi-honest computational security) or using threshold linear secret sharing (and achieve then semi-honest information-theoretic security).

Our protocols are parameterized by an integer constant τ. Our protocols have a number of rounds proportional to τ and they achieve a quasi-optimal communication complexity, i.e. exponential in $1 + 1/\tau$. More precisely:

- We present a first protocol (**FastPolyFanIn**) where the parties are given shares of n non-zero polynomials $[f_1], \dots, [f_n]$ in $\mathbb{F}_q[X]$ of degree less than d and compute shares of the polynomial $[f_1 \times \cdots \times f_n]$ of degree at most nd. Mohassel and Franklin [27] proposed a constant-round protocol with communication complexity of $\mathcal{O}(n^2 d)$ multiplications. Our improved protocol has communication complexity of only $\mathcal{O}(\tau n^{1+1/\tau} d)$ multiplications and $\mathcal{O}(\tau)$ rounds.
- Our second protocol (**FastEval**) allows the parties sharing a polynomial $[f] \in \mathbb{F}_q[X]$ of degree at most n and shared points $[\alpha_1], \dots, [\alpha_n]$ in $\mathbb{F}_q$ to compute shares of the n evaluations $[f(\alpha_1)], \dots, [f(\alpha_n)]$. This protocol achieves communication complexity of $\mathcal{O}(\tau n^{1+1/\tau})$ multiplications and $\mathcal{O}(\tau)$ rounds. The previously best-known protocol has communication complexity $\mathcal{O}(n^2)$, see [10].
- Our third protocol (**FastInterpol**) performs polynomial interpolation on shared values (as in Mohassel-Franklin protocol), with communication complexity of $\mathcal{O}(\tau n^{1+1/\tau})$ multiplications and $\mathcal{O}(\tau)$ rounds.

Table 1 Summary of our improvements on operations on shared polynomials (n is the number of polynomials for unbounded fan-in multiplication, d bounds the degree of the polynomials and τ is a predetermined constant).

	Our work	**Mohassel-Franklin ([27])**
Unbounded fan-in mult.	$\mathcal{O}(\tau n^{1+1/\tau} d)$	$\mathcal{O}(n^2 d)$
Multi-point evaluation	$\mathcal{O}(\tau n^{1+1/\tau})$	$\mathcal{O}(n^2)$
Interpolation	$\mathcal{O}(\tau n^{1+1/\tau})$	$\mathcal{O}(n^2)$

All of our protocols are perfectly correct, i.e. the result is always computed correctly. Furthermore, they are valid in the *semi-honest* (i.e. *honest-but-curious*) model. Table 1 summarizes the communication complexity of our protocol compared to existing ones in [27]. Our protocol for unbounded fan-in multiplication of shared polynomials can be used straightforwardly to compute the unbounded fan-in multiplication of n shared elements $[x_1], \ldots, [x_n]$ of $\mathbb{F}_q$ that are not necessarily invertible (i.e. some elements might be zero). This is achieved by setting $[f_1] = [X - x_1], \ldots, [f_n] = [X - x_n]$ and extracting the constant coefficient of the product $[f_1 \ldots f_n]$. This protocol is already mentioned in [7], but it did not improve upon the more general approach of Bar-Ilan and Beaver [1] yielding a constant round protocol with a communication complexity of $\mathcal{O}(n^2)$ secure multiplications. Thanks to our new result, the latter operations can now be achieved in constant rounds but with $\mathcal{O}(\tau n^{1+1/\tau})$ secure multiplications.

1.3 Applications to Private Set Operations

As a second contribution, we show that our secure polynomials computation framework can serve to improve protocols on the so-called *privacy-preserving set operations* (*PSO*s), in particular for the general multi-party case involving more than two players.

The setting of PSOs is the following: each participant owns its private input set. The goal is to privately compute a predetermined function on these input sets while revealing no information about each set. We will focus here on some of the most classical functions on the intersection: *emptiness*, *cardinality*, *weighted sum*, or simply revealing the intersection set itself, eventually according to a certain threshold size. *Private Intersection Set* (*PSI*) is a crucial tool in privacy-preserving data analysis and collaborative applications where multiple parties want to discover shared interests, overlaps, or common elements in their datasets without revealing the specific items in their sets.

Many algebraic approaches, mostly based on cryptographic assumptions, allow to provide efficient solutions for this problem. In particular, one can either rely on homomorphic encryption [13, 25] or on oblivious linear evaluation [17, 16] to achieve many of PSO functionalities. Some of the results, notably on *PSI*, have been also proposed without any cryptographic assumptions, achieving security under the information-theoretic model. Note that all these methods rely on the natural representation of a set $\{\alpha_1, \ldots, \alpha_n\} \subset \mathbb{F}_q$ by the degree-n polynomial $f(X) = (X - \alpha_1) \times \cdots \times (X - \alpha_n)$ which allow for instance to recover the intersection as the *greatest common divisor* of many polynomials. It is then very natural to rely on our protocol for distributed secure polynomial computation for dealing efficiently with *PSOs*.

We shall mention that *PSI* is an intensively studied topic and many other methods, not only algebraic, have been designed. We refer the reader to [29] for a nice survey on the numerous approaches to the *PSI* problem. While we are only interested in specific function on the intersection set, the general circuit-*PSI* problem [33, 34] allows to evaluate

Table 2 Summary of the communication complexities, in terms of secure multiplication, for *Private Disjointness Test* protocols. It is assumed that each player hold a set of n entries, and that τ is a pre-determined constant.

	Comm.	**Rounds**	**# players**	**Uncond.**
Ye et al. [37]	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$	2	Yes
Couteau et al. [6]	$\mathcal{O}(n)$	$\mathcal{O}(1)$	2	No
Chandran et al.[4]	$\mathcal{O}(mn\log^2(m))$	$\mathcal{O}(\log(m\log n))$	m	No
Sathya Narayanan et al. [30]	$\mathcal{O}(mn^2)$	$\mathcal{O}(\log(mn))$	m	Yes
Ours (section 4.1)	$\mathcal{O}(mn+\tau n^{1+1/\tau})$	$\mathcal{O}(\tau)$	m	Yes

any function given as a circuit on the intersection set. This generic problem received also a lot of attention, and some protocols might yield the best solution for a particular computation, e.g. testing the emptiness of the intersection set in the multi-party setting [4].

1.3.1 Constant-round multi-party protocol for *Private Disjointness Test*

Testing the emptiness of the intersection set is called *Private Disjointness Test* (*PDT*) in the literature and many secure protocols have been proposed to deal efficiently with *PDT*. The seminal work of Freedman, Nissim and Pinkas [13] proposed the first efficient solutions to both the two-party and the multi-party setting. Their protocols are secure against semi-honest adversaries or even malicious in the random oracle model. Their work has been further improved in [23, 21] notably to remove the random oracle hypothesis in the two-party setting. While the proposed protocols are efficient in terms of communication between the parties, the high depth of the protocol makes the need for a linear number of rounds of communication in the input sets' sizes.

The first constant round protocol for *PDT* is due to Ye et al. [37] for both honest-but-curious and malicious adversary cases. Unlike the previous two-party protocols, their protocols are unconditionally secure and only require communication complexity that is quadratic in the input set size. This result is improved under computational assumption by Couteau, Peters, and Pointcheval [6] who provide a two-party protocol for *PDT* achieving constant round, linear communication complexity and being secure against malicious adversaries.

While many works have studied *PDT* in the two-party setting, only very few works have been done in the multi-party setting, notably after the seminal work of Ye et al. [37]. Only the work of Sathya Narayanan *et al* [30] mentioned a dedicated protocol for *PDT* for more than two players. The given protocol is unconditionally secure against a semi-honest adversary, and it requires a logarithmic number of rounds and a quadratic number of communications in the input sets' size. One can achieve a similar result under a computational assumption (from oblivious transfer and cuckoo hashing) using the *circuit-PSI* protocol from [4], but with a number of rounds that is only double-logarithmic in the input sets' size.

While the challenge of designing constant round protocol having only a linear number of communication in the input sets' size is almost done in the two-party setting, the question remains open for the multi-party setting. By using our new multi-point evaluation protocol, we are able to propose an unconditionally secure constant-round protocol, against a semi-honest adversary, that achieves quasi-linear communication complexity in the input sets' size. More precisely, assuming the m players all hold a set of n elements, our protocol achieves a communication complexity of $\mathcal{O}(mn+\tau n^{1+1/\tau})$ multiplications and $\mathcal{O}(\tau)$ communication rounds, where τ is any non-zero integer constant given to the protocol. Table 2 summarizes the different complexity, security, and number of players for the best-known protocol for *PDT* in this setting.

One can see that our protocol is the first to achieve a constant number of rounds in the multi-party setting. Furthermore, it achieves the lowest number of communications under unconditional security. While our communication cost might be more important than the *circuit-PSI* approach [4] in some cases, the size of the exchanged data will remain in a small $\mathbb{F}_q$, say of size n, which should make a difference in practice compared to the larger fields of [4] required to reach the desired computational security.

We shall mention that our solution as well as all the previous works on PDT may produce an incorrect result, depending on the chosen implementation. In our case, this probability is negligible if the size of each set n is negligible compared to the domain size q. Moreover, this probability is one-sided, and having a wrong result does not reveal information about the inputs. Therefore, the protocol can be repeated to exponentially decrease the error probability.

1.3.2 A general framework to solve *PSO*s without any error probability

Among the *Private Set Operations*, *Private Set Intersection* has been the most studied since its introduction by Freedman, Nissim, and Pinkas in [13]. As for PDT the authors propose efficient protocols for both the two-party and the multi-party *PSI*, achieving security under the semi-honest adversary model or the malicious one. However, their protocol does not achieve a constant number of rounds of communication. Li and Wu provide in [26] the first constant-round protocol for *PSI*, and their protocol is unconditionally secure both with semi-honest and malicious adversaries, but the communication complexity is not yet optimal. Following a similar idea as [13], Hazay and Venkitasubramaniam propose the first protocol achieving constant-round and a linear communication complexity, under the computational model only.

Kissner and Song [25] extend multi-party protocols on sets beyond *PSI* notably by computing some other functions on the input sets.: e.g. union, element reduction, and intersection cardinality. In the *Cardinality Set Intersection* problem, the goal is to compute the number of elements in the intersection while not revealing other information on the intersection set. The protocol in [25] requires some cryptographic assumption and it involves a linear number of rounds of communication in the input sets size. The article [30] mentioned a first solution for *Cardinality Set Intersection* with unconditional security under passive and active adversaries, achieving a logarithmic number of communication rounds.

T-PSI (*Threshold Private Set Intersection*) is a variant of *PSI* in which the intersection is revealed only if its size surpasses a given threshold t. Gosh and Nilges present in [16] a first solution for *Threshold-PSI* in the multi-party setting. Their protocol is based on *Oblivious Linear Evaluations* and achieves unconditional security against malicious adversaries. The communication complexity of their protocol remains however quadratic in the size of each input set. In the two-party case, by using cryptographic assumption (namely the existence of fully homomorphic encryption), Ghosh and Simkin [17] managed to achieve a sub-linear communication complexity, i.e. the number of communications is quasi-linear in the number of elements that differ between the two sets. This result was extended to the multi-party setting in [18] using other cryptographic assumptions (the existence of linear homomorphic encryption and obvious transfer).

One of the most recent problems with private sets is the *Private Intersection Sum* problem. The setting, introduced in [22], is that one party has a set of elements together with some weight for each element, and he wants to compute the sum of the weight of all elements in common with a set of another party, of course without learning which elements are in common. The protocol proposed in [22] achieves security only in the computational model and for the two-party case. To our knowledge, no specific construction for this problem exists neither in the information-theoretic setting nor for more than two parties.

Unlike *PDT*, all of these contributions achieve a constant number of rounds and an optimal number of communications. However, the proposed protocols are not always secure under the information-theoretic setting, and all of them may fail to produce a correct result. Our work aims at bridging the gap to always achieve security without any cryptographic assumptions and to provide protocols designed for more than two parties, and that are perfectly correct (i.e. no incorrect result can be computed).

Here again, we show that by re-using our multi-point evaluation protocol together with some techniques that enable us to securely deal with boolean formula [11] we can achieve a general framework for *PSO*s that yields constant-round protocol without any incorrect result. As our solution embraces our generic MPC framework, the results are valid for the computational as well as the information-theoretic model. More precisely, we achieve a communication complexity of $\mathcal{O}(\tau m n^{1+1/\tau} + mn \log n \log\log n)$ secure multiplications for *PSI*, *PDT*, *Cardinality Set Intersection*, *Threshold-PSI* and *Private Intersection Sum*.

2 Technical overview

In this section, we present a brief overview of the techniques used in our protocols. This covers protocols on shared polynomials and for private set operations. We only focus on the main ideas of how the protocols work, and we give the full technical details in later sections. Without further assumption, the number of parties in the protocols will always be m.

2.1 Fast operations on shared polynomials

In section 3, we present new constant-round protocols for the unbounded fan-in multiplication of polynomials, multi-point evaluation, and interpolation involving only shared data. Our approach has some similarities with the work of Mohassel and Weinreb from [28] to lower the number of communications for secure linear algebra operations. More precisely, our approach allows us to choose any constant parameter $\tau \in \mathbb{N}^*$ and provides a quasi-optimal communication complexity, *i.e.* exponential in $1 + 1/\tau$, while achieving a constant number of rounds of $\mathcal{O}(\tau)$. Our improvement is based on the observation that the studied operations are strongly regular and thus can be decomposed into several instances of the same problem with smaller entries. Therefore, applying a generalized divide and conquer approach, *i.e.* splitting the problem of size accordingly to τ, and using existing protocols on sub-instances suffices to improve the communication complexity. For the sake of clarity, we will only present the idea behind our protocol for the case $\tau = 2$ in this technical overview. All the technical details and the more general approach for any τ is postponed to section 3.

2.1.1 Unbounded fan-in multiplication of polynomials

Let $[f_1], \dots, [f_n]$ be n shared non-zero polynomials in $\mathbb{F}_q[X]$ of degree $< d$. Parties want to compute shares of the polynomial $[f_1 \times \cdots \times f_n]$ of degree $< nd$. Mohassel and Franklin proposed in [27] a constant round protocol to compute such a product with $\mathcal{O}(n^2 d)$ secure multiplications in $\mathbb{F}_q$. We must mention that the complexity is quadratic in n because the protocol uses $\mathcal{O}(n)$ products in an extension field of $\mathbb{F}_q$ of degree nd. Our goal here is to perform most of the computation in smaller extension fields to reduce the complexity.

Assuming that n is a perfect square, one may remark that dividing the computation in $\sqrt{n}$ sub-products of $\sqrt{n}$ polynomials allow us to reach a better complexity. Indeed, parties can compute in parallel each sub-product of $\sqrt{n}$ polynomials with $\sqrt{n}$ calls to the protocol of Mohassel and Franklin [27] for a total cost of $\mathcal{O}(n^{1.5} d)$ secure multiplication in $\mathbb{F}_q$. To finish the computation, parties have to multiply $\sqrt{n}$ shared polynomials of degree less than $\sqrt{n}d$. Again this can be achieved by one call to the protocol from [27] for a cost of $\mathcal{O}(n^{1.5}d)$ secure multiplication in $\mathbb{F}_q$. We thus reduce the number of secured multiplication by $\sqrt{n}$ while we only double the number of rounds. We can generalize this idea to any fixed parameter $\tau \in \mathbb{N}^*$ in order to replace $\sqrt{n}$ with $n^{1/\tau}$ and then achieve a number of secure multiplication of $\mathcal{O}(n^{1+\frac{1}{\tau}})$ and $\mathcal{O}(\tau)$ rounds. Indeed, the explanation corresponds to the particular case of $\tau = 2$, but grouping the products by chunks of size $n^{\frac{1}{\tau}}$ at each step would only require τ steps to get the result.

2.1.2 Multi-point evaluation

Let $[f]$ be a shared polynomial in $\mathbb{F}_q[X]$ of degree $< n$ and $[\alpha_1], \ldots, [\alpha_n]$ be shared of points in $\mathbb{F}_q$. Parties want to compute the shares of the n polynomial evaluations $[f(\alpha_1)], \ldots, [f(\alpha_n)]$.

In the case that parties want to evaluate $[f]$ at a single shared point $[\alpha]$, they can rely on [10] to have a secure protocol that is constant-round and that has a linear communication complexity in the degree of f. Note that the protocol heavily relies on unbounded fan-in multiplication of 2×2 matrices in order to guarantee that no leakage occurs when $[\alpha] = [0]$. No previous works have considered the simultaneous evaluation on a shared set of points, and to the best of our knowledge, the single points approach from [10] remains the most efficient for that case. In particular, this yields a constant-round protocol with a communication complexity of $\mathcal{O}(n^2)$ for evaluating f on a set of n shared points.

Instead of relying on computing powers of the $[\alpha_i]$ as in [10], our approach relies on the polynomial division of f by well-chosen polynomials. It is fairly classical that $[f(\alpha_i)] = [f \bmod (X - \alpha_i)]$ (see [14]). Here, we exploit the constant-round protocol for the polynomial division of Franklin and Mohassel [27] that only requires a linear number of secure multiplications according to the dividend size. Unfortunately, applying straightforwardly this protocol for the division of f by each $(X - \alpha_i)$ would also require $\mathcal{O}(n^2)$ secure multiplications. Instead, we will re-use our unbounded fan-in multiplication protocol to compute the product of the n polynomials $(X - \alpha_i)$. Together with this product, we can obtain for free the $\sqrt{n}$ intermediate sub-products computed by the protocol (corresponding to the splitting of the result into $\sqrt{n}$ products of degree $\sqrt{n}$). There, we can reduce the polynomial $[f]$ modulo all these intermediate polynomials at a cost of $\sqrt{n}$ call to the division protocol of [27], yielding a communication complexity of $\mathcal{O}(n^{1.5})$ secure multiplications. Finally, we can further reduce these $\sqrt{n}$ distinct polynomials of degree less than $\sqrt{n}$ modulo the corresponding $(X - \alpha_i)$. Here again this can be done with only $\mathcal{O}(n^{1.5})$ secure multiplications since each reduction modulo one $(X - \alpha_i)$ costs $\mathcal{O}(n^{0.5})$.

To generalize this approach for any value of τ, we will exploit the general divide-and-conquer strategy of our unbounded fan-in multiplication protocol. In particular, we will reduce the polynomial $[f]$ modulo all the intermediate polynomials computed in our unbounded fan-in multiplication protocol, using a breadth-first browsing of the τ-ary tree.

2.1.3 Interpolation

Given $2n$ shared elements $[\alpha_1], \ldots, [\alpha_n]$ and $[y_1], \ldots, [y_n]$ in $\mathbb{F}_q$ such that $\alpha_1, \ldots, \alpha_n$ are distinct, parties want to compute the shares of $[f]$ such that f is the unique polynomial in $\mathbb{F}_q[X]_{<n}$ such that $y_i = f(\alpha_i)$ for $1 \leq i \leq n$. Franklin and Mohassel [27] proposed a constant-round protocol for this operation that requires $\mathcal{O}(n^2)$ secure multiplications. Their idea is to use Lagrange interpolation to compute $f = \sum_{i=1}^{n} y_i L_i(X)/L_i(\alpha_i)$, where $L = \prod_i^n (X - \alpha_i)$ and $L_i = L/(X - \alpha_i)$. Therefore, using constant-round protocols for

unbounded fan-in polynomial multiplication, euclidean division, and field element inversion suffices to reconstruct f. Unfortunately, this approach requires $\mathcal{O}(n^2)$ secure multiplication since most of the tasks boil down to n calls to a protocol that requires a linear number of multiplications, *i.e.* n divisions involving the polynomial L or the n polynomial evaluations $L_i(\alpha_i)$.

To remove the need to compute the terms depending on the L_is, we use the classical remark that $f/L = \sum_{i=1}^{n} c_i/(X-\alpha_i)$ where $c_i = y_i/L'(\alpha_i)$ and L' is the derivative of L. First, we compute L and then the c_i's using our previous protocols for unbounded fan-in multiplication of polynomials and multi-point evaluation. This is done in constant round with $\mathcal{O}(n^{1.5})$ secure multiplications (using $\tau = 2$). Then, we reconstruct the numerator of the fraction f/L in two steps, using a similar splitting strategy as in our previous protocols. First, we compute $\sqrt{n}$ different sums of $\sqrt{n}$ fractions of the form $c_i/(X-\alpha_i)$.

Let us define the polynomial $P_{1,1} = \prod_{l=1}^{\sqrt{n}}(X-\alpha_l)$. We can remark that the following equality holds $\sum_{i=1}^{\sqrt{n}} c_i/(X-\alpha_i) = (1/P_{1,1})\sum_{i=1}^{\sqrt{n}} c_i P_{1,1}/(X-\alpha_i)$ and that $G_{1,1} = \sum_{i=1}^{\sqrt{n}} c_i P_{1,1}/(X-\alpha_i)$ is a polynomial of degree $< \sqrt{n}$.

This equality extends naturally to all the $\sqrt{n}$ sums, and we thus can define the resulting fractions as $G_{1,1}/P_{1,1}, \ldots, G_{1,\sqrt{n}}/P_{1,\sqrt{n}}$. One may remark that computing numerators of these fractions amounts to taking linear combinations of the quotient of $P_{1,i}/(X-\alpha_j)$, which are exactly the same quotients as in the multi-point evaluation of L at the α_i's (second step). This step costs exactly $\mathcal{O}(n^{1.5})$ secure multiplication remarking that there is a total of n linear combinations of polynomial of degree less than $\sqrt{n}$.

As a second step, we write $f = \sum_{j=1}^{\sqrt{n}} G_{1,j}\frac{L}{P_{1,j}}$ where $L/P_{1,j}$ are polynomials of degree exactly $n - \sqrt{n}$. It thus remains to perform $\sqrt{n}$ products of polynomials of degree at most n and sum the results. Using the division and multiplication protocols of Mohassel and Franklin [27], both steps can be done in constant rounds with $\mathcal{O}(n^{1.5})$ secure multiplications. Altogether, we obtain a constant round protocol with $\mathcal{O}(n^{1.5})$ secure multiplications.

As before, splitting every computations in chunks of size $n^{\frac{1}{\tau}}$ implies $\mathcal{O}(\tau)$ rounds, yielding a protocol with communication complexity $\mathcal{O}(\tau n^{1+\frac{1}{\tau}})$.

2.2 Private set operations

We present in section 4 our solutions to many variants of the *PSI* problem using our fast protocols on shared polynomials. In these problems, m parties have the respective sets $\mathcal{A}_1, \ldots, \mathcal{A}_m \subseteq \mathbb{F}_q$ of size n each and wish to compute some function of the intersection, i.e. $f(\bigcap_{i=1}^{m} \mathcal{A}_i)$ for a predetermined function f. Following the seminal work of [13], the main algebraic approaches in the literature rely on encoding the parties' sets as polynomials. Let a set $\mathcal{A} \subseteq \mathbb{F}_q$, one can define $P_{\mathcal{A}}(X) = \prod_{\alpha\in\mathcal{A}}(X-\alpha)$ to be an encoding of A. Each party can then compute locally its own polynomial $P_j = P_{\mathcal{A}_j}$ for all $1 \leq j \leq m$, and engage in a constant round protocol to distribute shares of all these polynomials. From there, parties would have to compute the gcd of these shared polynomials to get a representation of the intersection, and then apply some computation on this gcd to get the desired result.

2.2.1 Constant-round protocol for Private Disjointness Test

Our first contribution concerns the *PDT* problem where the m parties want to know whether the intersection set is empty or not. We revamp a technique from [25] and [26] that uses a property on the gcd of many polynomials [14, Section 6.9]. More precisely, let $G = \gcd(P_{\mathcal{A}_1}, \ldots, P_{\mathcal{A}_m})$, and $R = \sum_j R_j P_{\mathcal{A}_j}$ where the R_j's are random polynomials of degree at most n in $\mathbb{F}_q[X]$. Therefore, we have that $G = \gcd(R, P_{\mathcal{A}_j})$ for $1 \leq j \leq m$ with high probability. The protocols of [25] and [26] aim at computing the polynomial R and making it public so that any party can compute locally the intersection set. Since the R_j are random polynomials, the security relies on the fact that R will not be distinguishable from any degree-n polynomial in the polynomial ideal $\langle G\rangle$.

For PDT we cannot afford to make the polynomial R public. Indeed, every party would then learn the intersection and this leaks more information than the emptiness of the intersection. However, we can keep this polynomial R private and use our fast protocol on shared polynomials to perform the computation. More precisely, let us define $[R] = \sum_{j=2}^{m}[r_j][P_{\mathcal{A}_j}]$ where r_j are nonzero random elements from $\mathbb{F}_q$. This polynomial R is an encoding of the intersection set between the parties $(2, \dots, m)$ with high probability. It is sufficient to ask the first party to share its elements $[\alpha_j]$ such that $\alpha_j \in \mathcal{A}_1$ among all the participants. Our protocol for PDT consists of evaluating the polynomial $[R]$ on all the $[\alpha_j]$, multiplying these evaluations, and checking whether the product is zero or not. All the steps are constant round and only need $\mathcal{O}(mn + \tau n^{1+1/\tau})$ secure multiplications using our multi-point evaluation protocol from Section 3.

As for many of the algebraic solutions to PDT, our approach may fail to produce a correct answer. Indeed, the polynomial R may contain roots that are not in the intersection while being an element of the set $\mathcal{A}_1$. This could happen with probability at most n/q. By taking a domain size q that is way larger than the size of the sets n, this probability is negligible. Moreover, as the error is one-sided, parties can decide to repeat the protocol several times to further lower the probability of an incorrect result.

2.2.2 Perfectly correct protocol for Private Set Operations

As seen in Section 1.3.2 most of the known protocols for PSO are constant round with an optimal communication complexity of $\mathcal{O}(mn)$ secure multiplications. However, similarly to our previous approach for PDT, the result may be incorrect due to the use of randomization: e.g. either because of sampling polynomial in the polynomial ideal defined by the intersection polynomial [26, 17] or because hashing technique may have collisions [13, 20]. One may ask whether it would be possible to have a protocol with similar complexities, i.e. constant round and linear communication, that always returns a correct output.

Our protocol in section 4 is a first step toward achieving such a result. In particular, we achieve most of the PSO functionalities without any errors, using a constant number of rounds and a sub-linear communication complexity of $\mathcal{O}(mn^{1+\frac{1}{\tau}} + mn \log n \log\log n)$ for any integer $\tau > 0$. Our method somehow generalizes the idea in [13, 20] to the multi-party case without having to call any two-party protocol. Here again, we ask the first party to share its set of elements with the other parties, and each of the other parties shares their elements encoded as a polynomial. From there, we can use our multi-point evaluation protocol to evaluate all parties' polynomials (except party one) over all the elements of the first party (or any designated party by the protocol).

Then, we convert all these evaluations into shared booleans using protocols from [11]. These booleans indicate if P_j evaluates to 0 on the i-th element of the first party (for $1 \leq i \leq n$ and $1 \leq j \leq m$). Since manipulations of these shared booleans are not too difficult (as explained in [11]), this allows us to solve many problems related to PSI in the claimed complexities. From this, it is straightforward to compute the logical AND of these booleans (see [11, Section 5.1]). The result is the booleans $[b_j]$, which indicate if the jth element of the first player is in the intersection. From this, it is easy to compute the solution to many problems, in a complexity that is smaller or equal to the complexity of the previous steps. We give here a few examples. The solution to PSI can be computed as $\prod[Q_j]$ where

$[Q_j] = [b_j(X - \alpha_j) + (1 - b_j)]$ ($Q_j = X - \alpha_j$ if the jth element α_j of the first player is in the intersection and $Q_j = 1$ otherwise). The solution for *PDT* can be computed as $\bigwedge b_j$. The solution to *Cardinality Set Intersection* can be computed as $\sum [b_j]$. The solution to *Private Intersection Sum* can be computed as $\sum [b_j][y_j]$. See table 3 for a summary of how to solve these problems using the shared booleans $[b_j]$.

3 Fast operations on shared polynomials

This section is devoted to new protocols for classical operations on shared polynomials that require a constant number of rounds and an almost optimal communication complexity. This follows the work of Mohassel and Franklin in PKC'06 [27] that first proposed such optimal protocols for the multiplication or the Euclidean division of shared polynomials. While their work also improved on the generic constant-round approach of Bar-Ilan and Beaver [1] for the interpolation, unbounded fan-in multiplication, and gcd on shared polynomials, the obtained communication complexity is not yet optimal. In the next sections, we will provide new constant-round protocols with almost optimal communication complexity, *i.e.* quasi-linear in the degree of shared input/output polynomials. This concerns the unbounded fan-in multiplication of shared polynomials and the multi-evaluation or the interpolation for polynomials on sets of points that are all shared.

3.1 Technical background

We begin by presenting some existing techniques that we need in our results. All of the following ideas are presented in either [1], [7] [27] or [10]. In particular, the protocols specifically designed for polynomials are results from [27]. All of these techniques are secure and in a constant number of rounds.

Generating a random invertible field element. This protocol generates a shared *non-zero* element $[y]$ of $\mathbb{F}_q$ (whose value is hidden from the players). This is done by generating two (unknown) shared elements of $\mathbb{F}_q$, multiplying them together securely, and revealing the result. If it is invertible, then one of the elements is taken as the result, otherwise, parties rerun the protocol. This protocol fails with probability at most $2/q$, so it only takes a constant number of secure multiplications (except with negligible probability). In our protocol, we denote this operation as "$[y] \overset{\$}{\leftarrow} \mathbb{F}_q^*$". Note that this can also be used for generating invertible matrices of constant size.

Inversion of an invertible field element. To invert an invertible shared element $[x] \in \mathbb{F}_q^*$, parties can use the previous technique to generate $[y] \overset{\$}{\leftarrow} \mathbb{F}_q^*$, compute $[z] = [x][y]$ and reveal its value. Lastly, each participant can locally compute their share of $[x^{-1}] = z^{-1}[y]$. It only takes a constant number of secure multiplication. Note that this can also be used for inverting invertible matrices of constant size.

Unbounded fan-in multiplication of invertible field elements. Multiplying n shared *invertible* elements $[x_1], \ldots, [x_n]$ cannot be done in a naive way in a constant number of rounds. To compute the product in MPC, parties generate n random invertible elements $[r_1], \ldots, [r_n]$ in parallel, as well as their inverse. Then, they compute in parallel $[p_1] = [x_1][r_1]$ and $[p_j] = [r_{j-1}^{-1}][x_j][r_j]$ for $2 \leq j \leq n$. The values of the p_j's are then revealed and everyone can compute their share of $\left[\prod x_j\right] = \left(\prod p_j\right)[r_n]$. This requires $2n - 1$ secure multiplications in total. Note that this protocol can also be used for computing the shared product of invertible matrices and that every prefix of the total product can be computed as a sharing, by locally computing $\left[\prod^k x_j\right] = \left(\prod^k p_j\right)[r_k]$. We denote this protocol as **FanInMul**.

Powers of any field element. This cannot be done directly using **FanInMul** with all multiplicand being the same shared element $[x]$ if $[x]$ might be zero. Indeed, when computing powers of $[0]$, the protocol would leak that information when revealing the products of the p_j's. To overcome this problem, Cramer, Kiltz and Padró in [10, Section 3] replaced x by an invertible 2×2 polynomial matrix $M(x)$ whose powers are related to Chebyshev polynomials of the first kind. Their method to compute shared powers consists then essentially in applying **FanInMul** to $M(x)$ and then carrying out public linear operations corresponding to the change of basis between the Chebyshev basis and the monomial one. Note that this latter step does not require any communication. The resulting protocol requires $\mathcal{O}(n)$ secure multiplications. From there, they can evaluate a publicly known or shared polynomial of degree n in a shared point $[x]$ in the same complexity.

Product of two polynomials. Multiplying two shared polynomials of degree n in $\mathbb{F}_q[X]$ naively would require $\mathcal{O}(n^2)$ secure multiplications in $\mathbb{F}_q$. Mohassel and Franklin [27, Section 3] proposed a protocol in $\mathcal{O}(n)$ instead, by noting that evaluating and interpolating a shared polynomial on public and pre-agreed points does not require any communication. It therefore suffices to evaluate both polynomials on $2n+1$ public points, securely multiply the evaluations on each point in parallel, and interpolate the resulting polynomial. Overall, only $2n+1 = \mathcal{O}(n)$ secure multiplications are performed. Using this protocol makes it possible to compute secure multiplication on shared elements of an extension field of $\mathbb{F}_q$ of degree n in $\mathcal{O}(n)$ secure multiplications in $\mathbb{F}_q$. This protocol needs that parties first agree on a public quotient polynomial, ensuring that modular reduction can be done with linear communication. We denote the protocol for polynomial multiplication over $\mathbb{F}_q[X]$ by **Poly2Mult**.

Euclidian division of polynomials. Mohassel and Franklin described in [27, Section 4] a protocol to compute shares of the quotient and remainder of two shared polynomials $[f]$ and $[g]$ with $d = \deg f \geq \deg g$. The main idea is to adapt the classical fast division approach [14, Section 9] to the shared setting. Notably, this is achieved by re-using the protocol for inverting group element of [1] to the case of the multiplicative subgroup $\mathbb{F}_q[X]/X^t$, *i.e.* when computing shared reverse polynomial of the quotient, and to use their subsequent protocol **Poly2Mult**. The division protocol requires a total $\mathcal{O}(d)$ secure multiplications in $\mathbb{F}_q$, and we denote it by **PolyDiv**.

Unbounded fan-in multiplication of non-zero polynomials. Given n non-zero polynomials $[f_1], \ldots, [f_n]$ of degree less than d, Mohassel and Franklin proposed in [27, Section 5] a protocol to compute $\left[\prod f_j\right]$ in constant-round, achieving the best-known complexity to date. This method boils down to using protocol **FanInMul** in an extension field of degree at least $n \times d$ and considering the polynomials as elements of this larger field. Since the protocol **FanInMul** requires $\mathcal{O}(n)$ secure multiplications in this extension field, each of them requires $\mathcal{O}(nd)$ secure multiplications in $\mathbb{F}_q$, thus the whole protocol requires a total of $\mathcal{O}(n^2 d)$ secure multiplications in $\mathbb{F}_q$. We note that this protocol imposes all parties to agree on a predetermined irreducible polynomial of degree nd. We suppose that such a polynomial can be predetermined outside any call of this protocol and thus no communication complexity will be counted for its computation. We denote this protocol by **PolyMult**.

3.2 Unbounded fan-in multiplication of polynomials

Let $[f_1], \ldots, [f_n]$ be n shared non-zero polynomials in $\mathbb{F}_q[X]$ of degree $< d$. Parties want to compute shares of the polynomial $[f_1 \times \cdots \times f_n]$ of degree $< nd$. Without loss of generality, we assume that $n = \lambda^\tau$ for an integer λ (the extra padding required to achieve this only adds a size negligible in n). Our approach consists in computing in parallel sub-products of exactly λ polynomials of increasing degree $< \lambda^i d$ for $i \in [0, \ldots, \tau - 1]$. For this, we define in definition 1 the polynomial products $P_{i,j}$ that we need to compute and that follow the following recursive definition. Lemma 2 bounds the degree of those polynomials. See the appendix for a proof of lemma 2.

▶ **Definition 1.** *Let $P_{i,j}$ be a polynomial of $\mathbb{F}_q[X]$ defined such that :*
$P_{0,j} = f_j$ for $1 \leq j \leq n$, and $P_{i,j} = \prod_{l=1}^{\lambda} P_{i-1,(j-1)\lambda+l}$ for $1 \leq i \leq \tau, 1 \leq j \leq \lambda^{\tau-i}$.

▶ **Lemma 2.** *For $0 \leq i \leq \tau$ and $1 \leq j \leq \lambda^{\tau-i}$, the polynomial $P_{i,j}$ is of degree less than $\lambda^i d$. Moreover, we have $\prod_{j=1}^{\lambda^{\tau-i}} P_{i,j} = \prod_{j=1}^{n} f_j$ for $0 \leq i \leq \tau$.*

From this lemma we notice that $P_{\tau,1} = \prod_{j=1}^{n} f_j$. We are then able to define a protocol that computes the shared polynomial $[P_{\tau,1}]$ in $\mathcal{O}(\tau)$ rounds. At each step, starting from step $i = 1$, parties compute in parallel all the shares of $[P_{i,j}]$ from the shares of $[P_{i-1,j}]$ obtained in the previous steps. The protocol FastPolyMult is described below as well as theorem 3 ensuring the correctness, security, and complexity of the protocol. The proof is available in the appendix.

Algorithm 1 FastPolyMult.

```
   Input: $n$ shared non-zero polynomial $[f_1], \ldots, [f_n]$ in $\mathbb{F}_q[X]_{<d}$, and $\tau \in \mathbb{N}^*$
   Output: shares of polynomial $\left[\prod_{i=1}^{n} f_i\right]$
1  for $i$ from 1 to $\tau$ do
       In parallel, players compute for $1 \leq j \leq \lambda^{\tau-i}$:
       $[P_{i,j}] = \prod_{l=1}^{\lambda}[P_{i-1,(j-1)\lambda+l}]$                    ▷ PolyMult
   end
2  return $[P_{\tau,1}]$
```

▶ **Theorem 3.** *FastPolyMult is correct, secure and requires $\mathcal{O}(\tau)$ rounds of communications and $\mathcal{O}(\tau n^{1+\frac{1}{\tau}} d)$ secure multiplications in $\mathbb{F}_q$.*

Note that in this protocol, the shared polynomials $[P_{i,j}]$ can be also returned as output. Notably, we will use these polynomials in the next section to improve the multi-evaluation of a shared polynomial on a shared set of points.

We note that the protocol FastPolyMult can be used to achieve faster unbounded fan-in multiplication of n shared (possibly zero) elements $[x_1], \ldots, [x_n]$ of $\mathbb{F}_q$. One can use the technique from [10] to construct a constant-round unbounded fan-in multiplication protocol having a linear communication complexity. However, such protocol only works with elements from the subgroup $\mathbb{F}_q^*$. In order to allow elements from $\mathbb{F}_q$ one must rely on the generic framework of Bar-Ilan and Beaver [1], but this comes with an expense of $\mathcal{O}(n^2)$ communications. In [7, Section 6.4], the authors suggest an alternative that is to compute shares of the polynomial $\prod(X - x_i)$ and get its constant term which is $(-1)^n \prod x_i$. Parties then simply need to multiply this shared coefficient by a publicly known constant to obtain the desired product. Thanks to our new protocol for unbounded fan-in multiplication of polynomials, we are now able to tackle this task with an almost linear communication complexity, improving on any previously known methods.

We shall mention that our strategy which consists to use a generic divide-and-conquer strategy with a depth of τ can also be applied to the approach of Bar-Ilan and Beaver. Their idea consists of replacing the multiplication of two field elements by a constant number of 3×3 non-zero matrix products. This yields a new unbounded fan-in multiplication problem with a similar number of terms but with constant-size matrices instead of field elements. Those products can of course be gathered similarly to our polynomials $P_{i,j}$, hence obtaining also a communication complexity of $\mathcal{O}(\tau n^{1+\frac{1}{\tau}})$.

Computing the same operations with potentially zero polynomials in the same multiplication complexity remains an open question. To our knowledge, the best currently known method is to consider the input polynomials as elements of a field extension of degree nd, and then to either use FastPolyMult on polynomials whose coefficients are in the field extension or to use the approach of Bar-Ilan and Beaver coupled with our generic divide and conquer strategy in the field extension. Both methods require $\mathcal{O}(\tau n^{2+\frac{1}{\tau}} d)$ secure multiplications to compute the product of n potentially zero polynomials of degree less than d.

3.3 Polynomial evaluation on shared set of points

Let $[f]$ be a shared polynomial in $\mathbb{F}_q[X]$ of degree $< n$ and $[\alpha_1], \dots, [\alpha_n]$ be shared of points in $\mathbb{F}_q$. Parties want to compute the shares of the n polynomial evaluations $[f(\alpha_1)], \dots, [f(\alpha_n)]$.

Assuming that n is a perfect square, we can replace the n evaluations of f with n evaluations of polynomials of degrees less than $\sqrt{n}$. Indeed, let $P_{1,1} = \prod_{l=1}^{\sqrt{n}}(X - \alpha_l)$ and $R_{1,1} = f \bmod P_{1,1}$ we have that $f(\alpha_l) = R_{1,1}(\alpha_l)$ for $1 \le l \le \sqrt{n}$. The same kind of relation holds for all polynomials $R_{1,j} = f \bmod P_{1,j}$ where $P_{1,j}$ follows Definition 1 with $\tau = \sqrt{n}$ and $f_j = (X - \alpha_j)$, *i.e.* $P_{1,j} = \prod_{l=(j-1)\sqrt{n}+1}^{j\sqrt{n}}(X - \alpha_l)$ for $1 \le j \le \sqrt{n}$. Using our protocol FastPolyMult we can compute the shared polynomials $[P_{1,1}], \dots, [P_{1,\sqrt{n}}]$ in constant-round with only $\mathcal{O}(n^{1.5})$ secure multiplications in $\mathbb{F}_q$. Computing the shared polynomials $[f \bmod P_{1,1}], \dots, [f \bmod P_{1,\sqrt{n}}]$ amounts to the same complexity and rounds by using $\sqrt{n}$ calls to the protocol **PolyDiv**, *i.e.* each division involves f and a polynomial of degree $\sqrt{n}$. To conclude the computation, parties have now to take every shared polynomial $[f \bmod P_{1,j}]$ of degree $< \sqrt{n}$ and to reduce each of them modulo the corresponding linear polynomials $(X - \alpha_k)$. This amounts to exactly n calls to protocol **PolyDiv** with a dividend of degree $< \sqrt{n}$ and a divisor of degree 1. This final step also costs $\mathcal{O}(n^{1.5})$ secure multiplications in $\mathbb{F}_q$, and it is also constant-round. This idea generalizes by assuming $n = \lambda^\tau$ for $\tau \in \mathbb{N}^*$. We can define the polynomial $P_{i.j}$ as in Definition 1 where $f_j = (X - \alpha_j)$. Let us also define the polynomials $R_{i,j}$ as follows, which satisfy a recurrence relationas stated in lemma 4. A proof for lemma 4 is avaible in the appendix.

► **Lemma 4.** *Let $R_{i,j} = f \bmod P_{i,j}$ be a polynomial of $\mathbb{F}_q[X]$. These polynomials satisfy the following recurrence relation:*
$R_{\tau,1} = f$, and $R_{i,j} = R_{i+1,\lceil j/\lambda \rceil} \bmod P_{i,j}$ for $0 \le i \le \tau - 1$ and $1 \le j \le \lambda^{\tau-i}$.

One may remark from the definitions of the polynomials $R_{i,j}$ that $R_{0,j} = f \bmod (X - \alpha_j) = f(\alpha_j)$ for $1 \le j \le n$. We can thus obtain a protocol in $\mathcal{O}(\tau)$ rounds by first computing the shares of the polynomials $[P_{i,j}]$ and then apply the recursive property of the polynomials $R_{i,j}$ to compute the shares of $[R_{0,j}]$ from $[f]$ and the $[P_{i,j}]$'s. Protocol FastEval is described below as well as theorem 5 ensuring the expected properties for this protocol. The proof for theorem 5 is avaible in the appendix. In the protocol, we let $[P_{0,1}], \dots, [P_{0,n}] = [(X - \alpha_0)], \dots, [(X - \alpha_n)]$.

Algorithm 2 FastEval.

```
Input: A shared polynomial $[f]$ of $\mathbb{F}_q[X]_{<n}$, a set $[\alpha_1], \ldots, [\alpha_n]$ of shared points in
       $\mathbb{F}_q$ and $\tau \in \mathbb{N}^*$
Output: $[f(\alpha_1)], \ldots, [f(\alpha_n)]$
1 Players compute $[P_{i,j}]$ for $1 \leq i \leq \tau$, $1 \leq j \leq \lambda^{\tau-i}$            ▷ FastPolyMult
2 for $i$ from $\tau - 1$ down to 0 do
      In parallel, players compute for $1 \leq j \leq \lambda^i$:
      $[R_{i,j}] = [R_{i+1,\lceil j/\lambda \rceil}] \mod [P_{i,j}]$                                ▷ PolyDiv
  end
  return $[R_{0,1}], \ldots, [R_{0,n}]$ ;
```

▶ **Theorem 5.** *FastEval is correct, secure and requires $\mathcal{O}(\tau)$ rounds of communications and $\mathcal{O}(\tau n^{1+\frac{1}{\tau}})$ secure multiplications in $\mathbb{F}_q$.*

3.4 Polynomial interpolation

Given $2n$ shared elements $[\alpha_1], \ldots, [\alpha_n]$ and $[y_1], \ldots, [y_n]$ in $\mathbb{F}_q$ such that the α_is are distinct, parties want to compute the shares of $[f]$ such that f is the unique polynomial in $\mathbb{F}_q[X]_{<n}$ such that $y_i = f(\alpha_i)$ for $1 \leq i \leq n$. The Lagrange interpolation states that $f = \sum_{i=1}^n y_i L_i(X)/L_i(\alpha_i)$ where $L = \prod_i^n (X - \alpha_i)$ and $L_i = L/(X - \alpha_i)$. It is well known, see [14], that the computation of $L_i(\alpha_i)$ can be replaced by $L'(\alpha_i)$ where L' is the derivative of L. To further remove the need to compute the polynomial L_i, one can use the also classical remark that $f/L = \sum_{i=1}^n c_i/(X - \alpha_i)$ where $c_i = y_i/L'(\alpha_i)$. Since our constant-round protocols FastPolyMult and FastEval allow us to compute efficiently the shares of $[L]$ and $[L'(\alpha_i)]$ for $1 \leq i \leq n$, the only remaining difficulty is the shares of $\sum_{i=1}^n c_i/(X - \alpha_i)$. Note that the derivative of L' can be done without any communication and the last multiplication by $L(X)$ is not needed as the α_i's are all distinct hence the denominator of $\sum_{i=1}^n c_i/(X - \alpha_i)$ will be indeed $L(X)$. Assuming that n is a perfect square, as we already did before, we can define the polynomial $P_{1,1} = \prod_{l=1}^{\sqrt{n}} (X - \alpha_l)$. We can remark that the following equality holds for the first $\sqrt{n}$ sumands: $\sum_{i=1}^{\sqrt{n}} \frac{c_i}{(X-\alpha_i)} = \frac{1}{P_{1,1}} \sum_{i=1}^{\sqrt{n}} \frac{c_i P_{1,1}}{(X-\alpha_i)}$, where $G_{1,1} = \sum_{i=1}^{\sqrt{n}} c_i P_{1,1}/(X - \alpha_i)$ is a polynomial of degree $< \sqrt{n}$ by definition of $P_{1,1}$. Doing similarly for the $\sqrt{n}$ chunks of the equation, each involving $\sqrt{n}$ summands, we will get $f/L = \sum_{j=1}^{\sqrt{n}} \frac{G_{1,j}}{P_{1,j}}$ where all denominators are of degree exactly $\sqrt{n}$. Therefore, we can write $f = \sum_{j=1}^{\sqrt{n}} G_{1,j} \frac{L}{P_{1,j}}$ where $L/P_{1,j}$ are polynomials of degree $n - \sqrt{n}$.

Assuming that shares of the $\sqrt{n}$ polynomial $[P_{1,j}]$ are known, this is the case since they are needed to compute shares of $[L]$ to get $[L']$ using protocol FastPolyMult. We also assume that the shares of $[c_i]$ has been computed efficiently using our protocol FastEval. Parties will have to perform $\sqrt{n}$ divisions for each $P_{1,j}$ by the adequate linear forms $(X - \alpha_i)$. Since $P_{1,j}$ is of degree $\sqrt{n}$ this amounts to $\sqrt{n} \times \mathcal{O}(\sqrt{n})$ secure multiplications in $\mathbb{F}_q$ using **PolyDiv** for each $P_{i,j}$. Computing all the shares $[G_{1,i}]$ thus requires $\mathcal{O}(n^{1.5})$ secure multiplications in $\mathbb{F}_q$. Parties then need to compute shares of $[L/P_{1,1}], \ldots, [L/P_{1,\sqrt{n}}]$. This is achieved with $\sqrt{n}$ call to **PolyDiv** with polynomials of degree at most n and it thus requires $\mathcal{O}(n^{1.5})$ secure multiplications in $\mathbb{F}_q$. Lastly, parties compute the shares of $[f] = \sum_{j=1}^{\sqrt{n}} [G_{1,j}] \times [L/P_{1,j}]$ which is done with $\mathcal{O}(n^{1.5})$ secure multiplications in $\mathbb{F}_q$ using **Poly2Mult**, *i.e.* all polynomials have degree at most $\sqrt{n}$. Altogether, the whole interpolation protocol is constant-round and has a communication complexity of $\mathcal{O}(n^{1.5})$.

To further generalize this approach, let us assume that $n = \lambda^\tau$ for $\tau \in \mathbb{N}^*$ and the polynomial $P_{i.j}$ are defined by Definition 1 where $f_j = (X - \alpha_j)$. We now define the general form for the polynomials $G_{i,j}$ in definition 6, and state lemma 7 which ensures that the polynomials $G_{i,j}$ have the expected property. The proof of lemma 7 is availble in the appendix.

▶ **Definition 6.** *Let $G_{i,j}$ be polynomials defined by the following relations, for $0 \leq i \leq \tau$ and $1 \leq j \leq \lambda^{\tau-i}$: $G_{0,j} = c_j$, and $G_{i,j} = \sum_{l=1}^{\lambda} G_{i-1,(j-1)\lambda+l} P_{i,j}/P_{i-1,(j-1)\lambda+l}$.*

▶ **Lemma 7.** *For $0 \leq i \leq \tau$ and $1 \leq j \leq \lambda^{\tau-i}$, $G_{i,j}$ has degree $< \lambda^i$. Moreover, $\sum_{j=1}^{\lambda^{\tau-i}} G_{i,j}/P_{i,j} = f/L$ for $1 \leq i \leq \tau$.*

We shall mention that thanks to the definition of the $G_{i,j}$ we have $G_{\tau,1} = f$. Protocol FastInterpol as well as the related theorem 8 follow. Its proof is avaible in the appendix.

Algorithm 3 FastInterpol.

```
Input: 2n shared elements $[\alpha_1], \ldots, [\alpha_n]$ and $[y_1], \ldots, [y_n]$ in $\mathbb{F}_q$ such that $\alpha_1, \ldots, \alpha_n$
       are distinct, $\tau \in \mathbb{N}^*$
Output: $[f]$ such that $f$ is the unique polynomial of degree $< n$ such that
        $y_i = f(\alpha_i)$.
1 Players compute $[P_{i,j}]$ for $1 \leq i \leq \tau$, $1 \leq j \leq \lambda^{\tau-i}$          ▷ FastPolyMult
2 Players compute locally $[L'] = [P'_{\tau,1}]$                                  ▷ no communication
3 Players compute $[L'(\alpha_1)], \ldots, [L'(\alpha_n)]$                              ▷ FastEval
4 Players compute $[G_{0,1}, \ldots, G_{0,n}] = [y_1][L'(\alpha_1)^{-1}], \ldots, [y_n][L'(\alpha_n)^{-1}]$
                         ▷ only secure multiplications and inversions of field elements
5 for $i$ from 1 to $\tau$ do
    a. Players compute (in parallel) for $1 \leq j \leq \lambda^{\tau-i}$, $1 \leq l \leq \lambda$:
       $[\gamma_{i,j,l}] = [P_{i,j}/P_{i-1,(j-1)\lambda+l}]$                          ▷ PolyDiv
    b. Players compute (in parallel) for $1 \leq j \leq \lambda^{\tau-i}$, $1 \leq l \leq \lambda$:
       $[\beta_{i,j,l}] = [G_{i-1,(j-1)\lambda+l}][\gamma_{i,j,l}]$                    ▷ Poly2Mult
    c. Players compute for $1 \leq j \leq \lambda^{\tau-i}$:
       $[G_{i,j}] = \sum_{l=1}^{\lambda}[\beta_{i,j,l}]$                              ▷ no communication
  end
6 return $[G_{\tau,1}]$
```

▶ **Theorem 8.** *FastInterpol is correct, secure and requires $\mathcal{O}(\tau)$ rounds of communications and $\mathcal{O}(\tau n^{1+\frac{1}{\tau}})$ secure multiplications in $\mathbb{F}_q$.*

4 Application to private set operations

In this section, we show how our protocols can be used to design several multi-party protocols for operations on private sets. We suppose that m players participate in the protocol and each of them has a set of n elements of $\mathbb{F}_q$. They want to compute a predetermined function of the intersection of their input sets. We denote by $\mathcal{A}_1, \ldots, \mathcal{A}_m$ their respective sets. In the *Private Set Intersection* problem, first introduced in [13], the participants want to compute the intersection $\bigcap_{i=1}^{m} \mathcal{A}_i$. In the *Private Disjointness Test* problem, also introduced in [13], the player wants to know whether the intersection $\bigcap_{i=1}^{m} \mathcal{A}_i$ is empty or not. In the *Cardinality Set Intersection* (*CSI*) problem, participants want to know the number of elements in the intersection. In the *Threshold Private Set Intersection* (*T-PSI*) problem, parties want to obtain the intersection if and only if the number of elements inside the intersection exceeds a public threshold t. In the *Private Intersection Sum* (*PIS*) problem, introduced in [22], the first participant has also a set $\mathcal{Y}$ of n integers such that each element is associated with an element of $\mathcal{A}_1$. The goal is to compute the sum of elements of $\mathcal{Y}$ such that the associated elements of $\mathcal{A}_1$ are in the intersection $\bigcap_{i=1}^{m} \mathcal{A}_i$.

All the multi-party protocols that we present to solve these problems manipulate shared polynomials in order to compute the solution without revealing information about a secret input set. For a set $\mathcal{A} \subseteq \mathbb{F}_q$, we define the encoding polynomial $P_{\mathcal{A}}(X) = \prod_{\alpha \in \mathcal{A}}(X - \alpha)$, and we let $P_j = P_{\mathcal{A}_j}$ for all $1 \leq j \leq m$. The obvious thing to note is that for $x \in \mathbb{F}_q$, x is in $\mathcal{A}$ if and only if $P_{\mathcal{A}}(x) = 0$. Moreover, in our methods, we use polynomial evaluations on a designated player's inputs. For the sake of clarity, we suppose that this player is the first and we let $\mathcal{A}_1 = \{\alpha_1, \dots, \alpha_m\}$. In section 4.2, we also manipulate shared booleans, which are equal to 1 when the predicate that the boolean represents is true and 0 otherwise.

4.1 A probabilistic solution for *Private Disjointness Test*

In this section, we present a protocol for solving the *PDT* problem using our evaluation protocol from section 3.3. Our approach is inspired by techniques used for instance in [25] and [26]. Essentially, it consists in privately generating random polynomials $R_1, \dots, R_m$ and privately computing $F = \sum R_i F_i$ before revealing F to all the participants. They then evaluate F locally on their input elements. This method doesn't leak any information other than the intersection because F is in fact a uniformly random multiple of $\gcd(P_1, \dots, P_m)$ with a given degree bound. Moreover, with high probability F evaluates to 0 only on the elements in the intersection among all the input elements. This solution for *PSI* requires $\mathcal{O}(mn)$ secure multiplications. Our solution for *PDT* uses the same idea of privately computing a polynomial $G = \sum r_i P_i$. But in this case, G is *not* revealed so that the r_i's only need to be random elements in $\mathbb{F}_q$. Then, G is privately evaluated using our protocol FastEval on the elements of the designated participant, which are denoted by $\alpha_1, \dots, \alpha_n$. With probability n/q, the only α_js on which G evaluates to zero are the points in the intersection. In order for this probability to be negligible, q needs to be overwhelmingly large compared to n when using this protocol. Then, since the remark at the end section 3.2 allows participants to use FastPolyMult to multiply n potentially zero shared field elements, parties can compute the product of the evaluations. Knowing if the product is zero is enough to know whether the intersection is empty or not. A formal description of this protocol FastPDT and theorem 9 follow. The proof of theorem 9 is avaible in the appendix.

Algorithm 4 FastPDT.

```
Input: Player $i \geq 2$ knows $\mathcal{A}_i \subseteq \mathbb{F}_q$ of size $n$, Player 1 knows
       $\mathcal{A}_1 = \{\alpha_1, \dots, \alpha_n\} \subseteq \mathbb{F}_q$, everyone knows $\tau \in \mathbb{N}$.
Output: All players know whether $\mathcal{A}_1 \bigcap \cdots \bigcap \mathcal{A}_m = \emptyset$ or not.
1 For $2 \leq i \leq m$, each player $i$ computes $P_i = \prod_{\alpha \in \mathcal{A}_i}(X - \alpha)$ locally. All these
    polynomials as well as $\alpha_1, \dots, \alpha_n$ are shared between all the participants.
2 In parallel, players generate $[r_i] \xleftarrow{\$} \mathbb{F}_q$ for $2 \leq i \leq m$.
3 In parallel, players compute for $2 \leq i \leq m$: $[r_i P_i] = [r_i][P_i]$.        ▷ multiplications
4 Without communication, players compute $[G] = \sum_{i=2}^{m} [r_i P_i]$.
5 Players compute $[G(\alpha_1)], \dots, [G(\alpha_n)]$.                                ▷ FastEval
6 Players compute $[b] = [G(\alpha_1) \dots G(\alpha_n)]$.                             ▷ FastPolyMult
7 Players generate the sharing $[r] \xleftarrow{\$} \mathbb{F}_q^*$.
8 Players compute $[br] = [b][r]$ and reveal the value $br$. If $br = 0$, players return "not
    empty". Else, players return "empty".
```

▶ **Theorem 9.** *FastPDT is secure and requires $\mathcal{O}(\tau)$ rounds of communications and $\mathcal{O}(mn + \tau n^{1+1/\tau})$ secure multiplications. Moreover, parties always deduce the correct result if the intersection is non-empty. If it is empty, then parties deduce the correct result with a probability larger than $1 - n/q$.*

4.2 A New Generic Technique for Perfectly Correct Private Set Operations

We present a general approach to designing secure protocols for *PSOs* on the intersection of sets. This solution is slower than our method to solve *PDT* but can be used as a general framework to solve multiple problems related to set intersection, with no probability of returning an incorrect result. We will need two MPC techniques presented in [11]. The first protocol (*cf.* [11, Section 7.1]) computes, from a share $[x]$ of an element of $\mathbb{F}_q$, a shared boolean denoted $[x = 0]$ which is equal to one if and only if $x = 0$. It is constant-round and requires $\mathcal{O}(\log q \log\log q)$ secure multiplications. The authors explain that with negligible probability, this protocol can leak information. However, it is possible for the participant to detect when it is the case, and to abort the protocol and retry it before information is leaked. The second one is explained in [11, Section 5.1]. It aims to compute a symmetrical logical operation (in our case we only need to compute the logical "and" and the logical "or") from n sharing in constant-round and $\mathcal{O}(n)$ secure multiplications, and is secure. For these protocols to work, it is required that the field F_q is a prime field. Moreover, for this protocol we can take a prime q such that $q = \mathcal{O}(n)$, so that the secure multiplication complexity of computing $[x = 0]$ from $[x]$ is $\mathcal{O}(\log q \log\log q) = \mathcal{O}(\log n \log\log n)$.

Our generic method is then as follows: players privately evaluate every polynomial P_i on the input elements of the first player (using our protocol from 3.3). Then, they convert the evaluations into booleans using the method from [11], *i.e.*, to get shares of $[P_i(\alpha_j) \neq 0]$ (these shared booleans are 0 if the evaluation is also 0 and 1 otherwise). These booleans can be used to privately compute shares of the booleans $b_j = (\alpha_j \notin \mathcal{A}_1 \bigcap \cdots \bigcap \mathcal{A}_m)$, which indicate if each element is in the intersection. This can be done using the second method from [11], since $b_j = \bigvee_{i=2}^{m} (P_i(\alpha_j) \neq 0)$. Once the parties have shares of the b_j, it is almost straightforward to get the output of the desired problem. We first present the generic protocol BoolIntersection and then give a few examples of how it can be used to solve *PSI*-related problems. The proof of theorem 10 can be found in the appendix.

Algorithm 5 BoolIntersection.

```
Input: Player $i \geq 2$ knows $\mathcal{A}_i \subseteq \mathbb{F}_q$ of size $n$, Player 1 knows
       $\mathcal{A}_1 = \{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}_q$, everyone knows $\tau \in \mathbb{N}$.
Output: $[b_1] = [\alpha_1 \notin \mathcal{A}_1 \bigcap \cdots \bigcap \mathcal{A}_m], \ldots, [b_n] = [\alpha_n \notin \mathcal{A}_1 \bigcap \cdots \bigcap \mathcal{A}_m]$.
1 For $2 \leq i \leq m$, each player $i$ computes $P_i = \prod_{\alpha \in \mathcal{A}_i} (X - \alpha)$ locally. All these
    polynomials as well as $\alpha_1, \ldots, \alpha_n$ are shared between all the players.
2 In parallel, players compute for $2 \leq i \leq m$ and $1 \leq j \leq n$:
  $[P_i(\alpha_j)]$.                                              ▷ FastEval
3 In parallel, players compute for $2 \leq i \leq m$ and $1 \leq j \leq n$:
  $[P_i(\alpha_j) \neq 0]$.                                       ▷ [11, Section 7.1]
4 In parallel, players compute for $1 \leq j \leq n$:
  $[b_j] = \bigvee_{i=2}^{m} [P_i(\alpha_j) \neq 0]$.             ▷ [11, Section 5.1]
5 return $[b_1], \ldots, [b_n]$.
```

▶ **Theorem 10.** *BoolIntersection is correct, secure and requires $\mathcal{O}(\tau)$ rounds of communications and $\mathcal{O}(mn \log n \log\log n + \tau mn^{1+1/\tau})$ secure multiplications.*

Once parties execute protocol BoolIntersection, it is easy to solve a multitude of problems without any error and in a secure manner. We give below a few examples for which the extra steps require less communication than the protocol to generate the booleans. Table 3 will summarize the computations that are described below.

Table 3 Algebraization of variants of the *PSI* primitive for n parties with private sets $\mathcal{A}_1, \dots, \mathcal{A}_n$ and the intersection set $\mathcal{I} = \mathcal{A}_1 \cap \cdots \cap \mathcal{A}_n$ where b_j denotes the Boolean $b_j = (\alpha_j \notin \mathcal{A}_1 \bigcap \cdots \bigcap \mathcal{A}_m)$ and $Q_j = b_j(X - \alpha_j) + (1 - b_j)$ for $j \in \{1, \dots, n\}$.

Name	Aim	Algebraization
PSI	$\mathcal{I}$	$\prod_{j=1}^{n}[Q_j]$
PDT	$\mathcal{I} \overset{?}{=} \emptyset$	$[b'] = \bigwedge_{j=1}^{m}[b_j]$
CSI	$\#\mathcal{I}$	$\sum_{j=1}^{n}[b_j]$
T-PSI	$\mathcal{I}$ only if $\#\mathcal{I} \geq t$	$\left[t \geq \sum_{j=1}^{n} b_j\right] \prod_{j=1}^{n}[Q_j]$
PIS	$\sum_{j \mid \alpha_j \in \mathcal{I}} y_j$	$\sum_{j=1}^{n}[b_j][y_j]$

Private Disjointness Test (PDT). Parties can simply compute $[b'] = \bigwedge_{j=1}^{m}[b_j]$ and reveal its value.

Private Set Intersection (PSI). By letting $Q_j = b_j(X - \alpha_j) + (1 - b_j)$, we note that $Q_j = 1$ if α_j is not in the intersection and $Q_j = X - \alpha_j$ otherwise. Therefore $P_{\mathcal{A}_1 \bigcap \cdots \bigcap \mathcal{A}_m} = \prod Q_j$. To solve *PSI*, parties can compute in parallel $[Q_j]$ for $1 \leq j \leq n$, and then use protocol FastPolyMult to compute $[P_{\mathcal{A}_1 \bigcap \cdots \bigcap \mathcal{A}_m}]$ and then reveal it. Parties can then evaluate locally this polynomial on their input points in order to know the intersection.

Cardinality Set Intersection (CSI). Parties can simply compute and reveal $\sum[b_j]$ without secure multiplications to know the number of elements in the intersection.

Threshold Private Set Intersection (T-PSI). Given the threshold t, parties can compute $[l] = \sum[b_j]$ without communication and then compute $[t \geq l]$ using techniques presented in [11]. Then, using the same method as for solving *PSI*, parties compute $[P_{\mathcal{A}_1 \bigcap \cdots \bigcap \mathcal{A}_m}]$ and they reveal $[t \geq l][P_{\mathcal{A}_1 \bigcap \cdots \bigcap \mathcal{A}_m}]$. If it is the zero polynomial, then it means that the threshold is not reached and nothing is revealed. If it is a non-zero polynomial, then it means the threshold is reached and the intersection can be computed locally by every participant.

Private Intersection Sum (PIS). Given a payload $\mathcal{Y} = \{y_1, \dots, y_n\}$ known by the first player, its elements are shared between everyone. Then, players can simply compute $[b_j][y_j]$ for $1 \leq j \leq n$ in parallel. Lastly, players compute $\sum[b_j y_j]$ and reveal it to obtain the result.

All these constant-round protocols are secure, with no occurrence of an incorrect result, and require $\mathcal{O}(\tau m n^{1+1/\tau} + mn \log q \log\log q)$ secure multiplications.

References

1 Judit Bar-Ilan and Donald Beaver. Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In Piotr Rudnicki, editor, *8th ACM PODC*, pages 201–209. ACM, August 1989. doi:10.1145/72981.72995.

2 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988. doi:10.1145/62212.62213.

3 Guilhem Castagnos and Fabien Laguillaumie. Linearly homomorphic encryption from DDH. In Kaisa Nyberg, editor, *CT-RSA 2015*, volume 9048 of *LNCS*, pages 487–505. Springer, Heidelberg, April 2015. doi:10.1007/978-3-319-16715-2_26.

4 Nishanth Chandran, Nishka Dasgupta, Divya Gupta, Sai Lakshmi Bhavana Obbattu, Sruthi Sekar, and Akash Shah. Efficient linear multiparty PSI and extensions to circuit/quorum PSI. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 1182–1204. ACM Press, November 2021. doi:10.1145/3460120.3484591.

5 David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th ACM STOC*, pages 11–19. ACM Press, May 1988. doi:10.1145/62212.62214.

6 Geoffroy Couteau, Thomas Peters, and David Pointcheval. Encryption switching protocols. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 308–338. Springer, Heidelberg, August 2016. doi:10.1007/978-3-662-53018-4_12.

7 Ronald Cramer and Ivan Damgård. Secure distributed linear algebra in a constant number of rounds. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 119–136. Springer, Heidelberg, August 2001. doi:10.1007/3-540-44647-8_7.

8 Ronald Cramer, Ivan Damgård, and Ueli M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 316–334. Springer, Heidelberg, May 2000. doi:10.1007/3-540-45539-6_22.

9 Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 280–299. Springer, Heidelberg, May 2001. doi:10.1007/3-540-44987-6_18.

10 Ronald Cramer, Eike Kiltz, and Carles Padró. A note on secure computation of the Moore-Penrose pseudoinverse and its application to secure linear algebra. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 613–630. Springer, Heidelberg, August 2007. doi:10.1007/978-3-540-74143-5_34.

11 Ivan Damgård, Matthias Fitzi, Eike Kiltz, Jesper Buus Nielsen, and Tomas Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 285–304. Springer, Heidelberg, March 2006. doi:10.1007/11681878_15.

12 Matthew K. Franklin and Stuart Haber. Joint encryption and message-efficient secure computation. *Journal of Cryptology*, 9(4):217–232, September 1996. doi:10.1007/BF00189261.

13 Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer, Heidelberg, May 2004. doi:10.1007/978-3-540-24676-3_1.

14 Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra (third edition)*. Cambridge University Press, 2013. doi:10.1017/CBO9781139856065.

15 Rosario Gennaro, Michael O. Rabin, and Tal Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In Brian A. Coan and Yehuda Afek, editors, *17th ACM PODC*, pages 101–111. ACM, June / July 1998. doi:10.1145/277697.277716.

16 Satrajit Ghosh and Tobias Nilges. An algebraic approach to maliciously secure private set intersection. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 154–185. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17659-4_6.

17 Satrajit Ghosh and Mark Simkin. The communication complexity of threshold private set intersection. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019,*

Part II, volume 11693 of *LNCS*, pages 3–29. Springer, Heidelberg, August 2019. `doi:10.1007/978-3-030-26951-7_1`.

18 Satrajit Ghosh and Mark Simkin. Threshold private set intersection with better communication complexity. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part II*, volume 13941 of *LNCS*, pages 251–272. Springer, Heidelberg, May 2023. `doi:10.1007/978-3-031-31371-4_9`.

19 Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th ACM STOC*, pages 365–377. ACM Press, May 1982. `doi:10.1145/800070.802212`.

20 Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. Scalable multi-party private set-intersection. In Serge Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 175–203. Springer, Heidelberg, March 2017. `doi:10.1007/978-3-662-54365-8_8`.

21 Susan Hohenberger and Stephen A. Weis. Honest-verifier private disjointness testing without random oracles. In George Danezis and Philippe Golle, editors, *PET 2006*, volume 4258 of *LNCS*, pages 277–294. Springer, Heidelberg, June 2006. `doi:10.1007/11957454_16`.

22 Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, Mariana Raykova, David Shanahan, and Moti Yung. On deploying secure computing: Private intersection-sum-with-cardinality. In *IEEE European Symposium on Security and Privacy, EuroS&P 2020, Genoa, Italy, September 7-11, 2020*, pages 370–389. IEEE, 2020. `doi:10.1109/EuroSP48549.2020.00031`.

23 Aggelos Kiayias and Antonina Mitrofanova. Testing disjointness of private datasets. In Andrew Patrick and Moti Yung, editors, *FC 2005*, volume 3570 of *LNCS*, pages 109–124. Springer, Heidelberg, February / March 2005.

24 Eike Kiltz, Payman Mohassel, Enav Weinreb, and Matthew K. Franklin. Secure linear algebra using linearly recurrent sequences. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 291–310. Springer, Heidelberg, February 2007. `doi:10.1007/978-3-540-70936-7_16`.

25 Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 241–257. Springer, Heidelberg, August 2005. `doi:10.1007/11535218_15`.

26 Ronghua Li and Chuankun Wu. An unconditionally secure protocol for multi-party set intersection. In Jonathan Katz and Moti Yung, editors, *ACNS 07*, volume 4521 of *LNCS*, pages 226–236. Springer, Heidelberg, June 2007. `doi:10.1007/978-3-540-72738-5_15`.

27 Payman Mohassel and Matthew Franklin. Efficient polynomial operations in the shared-coefficients setting. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 44–57. Springer, Heidelberg, April 2006. `doi:10.1007/11745853_4`.

28 Payman Mohassel and Enav Weinreb. Efficient secure linear algebra in the presence of covert or computationally unbounded adversaries. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 481–496. Springer, Heidelberg, August 2008. `doi:10.1007/978-3-540-85174-5_27`.

29 Daniel Morales, Isaac Agudo, and Javier Lopez. Private set intersection: A systematic literature review. *Computer Science Review*, 49:100567, 2023. `doi:10.1016/j.cosrev.2023.100567`.

30 G. Sathya Narayanan, T. Aishwarya, Anugrah Agrawal, Arpita Patra, Ashish Choudhary, and C. Pandu Rangan. Multi party distributed private matching, set disjointness and cardinality of set intersection with information theoretic security. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS 09*, volume 5888 of *LNCS*, pages 21–40. Springer, Heidelberg, December 2009.

31 Kobbi Nissim and Enav Weinreb. Communication efficient secure linear algebra. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 522–541. Springer, Heidelberg, March 2006. `doi:10.1007/11681878_27`.

32 Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Heidelberg, May 1999. doi:10.1007/3-540-48910-X_16.

33 Benny Pinkas, Thomas Schneider, Oleksandr Tkachenko, and Avishay Yanai. Efficient circuit-based PSI with linear communication. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 122–153. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17659-4_5.

34 Peter Rindal and Phillipp Schoppmann. VOLE-PSI: Fast OPRF and circuit-PSI from vector-OLE. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 901–930. Springer, Heidelberg, October 2021. doi:10.1007/978-3-030-77886-6_31.

35 Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

36 Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982. doi:10.1109/SFCS.1982.38.

37 Qingsong Ye, Huaxiong Wang, Josef Pieprzyk, and Xian-Mo Zhang. Efficient disjointness tests for private datasets. In Yi Mu, Willy Susilo, and Jennifer Seberry, editors, *ACISP 08*, volume 5107 of *LNCS*, pages 155–169. Springer, Heidelberg, July 2008.

A Appendix

Proof of Lemma 2. Let us prove the lemma by induction on $0 \leq i \leq \tau$. When $i = 0$, for $1 \leq j \leq n$, $P_{0,j} = f_j$ is of degree $< d$ by hypothesis, and $\prod_{i=1}^{n} P_{0,i} = \prod_{i=1}^{n} f_i$. When $1 \leq i \leq \tau$, by definition $P_{i,j} = \prod_{l=1}^{\lambda} P_{i-1,(j-1)\lambda+l}$ for $1 \leq j \leq \lambda^{\tau-i}$. Therefore, $P_{i,j}$ is the product of λ polynomials of degree $< \lambda^{i-1}d$ by induction hypothesis, thus it is a polynomial of degree $< \lambda^i d$. Moreover, we notice that $[\![1, \lambda^{\tau-i+1}]\!] = \{(j-1)\lambda+l \mid 1 \leq j \leq \lambda^{\tau-i},\ 1 \leq l \leq \lambda\}$. Therefore, by induction:

$$\prod_{j=1}^{\lambda^{\tau-i}} P_{i,j} = \prod_{j=1}^{\lambda^{\tau-i}} \prod_{l=1}^{\lambda} P_{i-1,(j-1)\lambda+l} = \prod_{j=1}^{\lambda^{\tau-i+1}} P_{i-1,j} = \prod_{j=1}^{n} f_j,$$

which concludes the proof. ◀

Proof of Lemma 3. Correctness of the protocol is ensured by Lemma 2 while its security is guaranteed by the use of protocol **PolyMult** and that the f_i's are non-zero. Since the latter protocol is constant-round and there is τ sequential steps, the protocol requires $\mathcal{O}(\tau)$ rounds of communications. Moreover, **PolyMult** is used $\lambda^{\tau-i}$ times at step i and each call requires $\mathcal{O}(\lambda^2\lambda^{i-1}d)$ secure multiplication,*i.e.* each call computes the product of λ shared polynomials of degree $< \lambda^{i-1}d$. Hence, step i requires $\mathcal{O}(\lambda^{\tau+1}d)$ secure multiplications. Summing over all the steps leads to $\mathcal{O}(\tau\lambda^{\tau+1}d) = \mathcal{O}(\tau n^{1+\frac{1}{\tau}}d)$ secure multiplications in $\mathbb{F}_q$. ◀

Proof of Lemma 4. $R_{i+1,\lceil j/\lambda \rceil}$ is well-defined since $1 \leq j \leq \lambda^{\tau-i}$, therefore $1 \leq \lceil j/\lambda \rceil \leq \lambda^{\tau-i-1}$. It now suffices to prove that $P_{i,j} | P_{i+1,\lceil j/\lambda \rceil}$, because then:

$$R_{i+1,\lceil j/\lambda \rceil} \bmod P_{i,j} = (f \mod P_{i+1,\lceil j/\lambda \rceil}) \mod P_{i,j} = f \mod P_{i,j} = R_{i,j}.$$

By letting $l = j - (\lceil j/\lambda \rceil - 1)\lambda$, we have that $1 \leq l \leq \lambda$ and $j = (\lceil j/\lambda \rceil - 1)\lambda + l$. By definition of $P_{i+1,\lceil j/\lambda \rceil}$ (see Def. 1), we have that $P_{i,j}$ divides $P_{i+1,\lceil j/\lambda \rceil}$. This concludes the proof. ◀

Proof of Theorem 5. Correctness is ensured by the definition of the $R_{i,j}$ and that the protocol **PolyDiv** correctly computes Euclidean division. Moreover, it is secure since the only used protocols, *i.e.* FastPolyMult and **PolyDiv**, are secure protocols. Protocol FastPolyMult requires $\mathcal{O}(\tau)$ rounds and **PolyDiv** is constant-round and is used in τ sequential step, so the total number of round is $\mathcal{O}(\tau)$. Lastly, using Lemma 2 one may remark that at step i of the loop we perform $\lambda^{\tau-i}$ divisions with a dividend of degree λ^{i+1} and a divisor of degree λ^i. Therefore, step i requires at most $\mathcal{O}(\lambda^{\tau+1})$ secure multiplications. Summed over all steps, this leads to protocol FastPolyMult requiring at most $\mathcal{O}(\tau n^{1+\frac{1}{\tau}})$ secure multiplications. ◀

Proof of Lemma 7. The statements are proven by induction for $0 \le i \le \tau$. When $i = 0$, $\deg G_{0,j} = \deg c_j = 0 < 1$. Moreover, Lagrange formula implies that $f/L = \sum_{i=1}^{n} c_i/(X - \alpha_i) = \sum_{j=1}^{\lambda^\tau} G_{0,j}/P_{0,j}$. When $0 \le i \le \tau$, we notice that $P_{i,j}$ has degree λ^i and $P_{i-1,(j-1)\lambda+l}$ has degree λ^{i-1} for $1 \le j \le \lambda^{\tau-i}$ and $1 \le l \le \lambda$. Therefore using Definition 6:

$$\begin{aligned}\deg(G_{i,j}) &\le \max_{1\le j\le\lambda^{\tau-i}} \left(\deg(G_{i-1,(j-1)\lambda+l}) + \deg(P_{i,j}) - \deg(P_{i-1,(j-1)\lambda+l})\right)\\ &< \lambda^{i-1} + \lambda^i - \lambda^{i-1}\\ &< \lambda^i.\end{aligned}$$

Moreover, $[\![0, \lambda^{\tau-i+1}]\!] = \{(j-1)\lambda + l \mid 1 \le j \le \lambda^{\tau-i},\ 1 \le l \le \lambda\}$ thus by using both Definitions 1 and 6 :

$$\begin{aligned}\sum_{j=1}^{\lambda^{\tau-i+1}} \frac{G_{i-1,j}}{P_{i-1,j}} &= \sum_{j=1}^{\lambda^{\tau-i}} \sum_{l=1}^{\lambda} \frac{G_{i-1,(j-1)\lambda+l}}{P_{i-1,(j-1)\lambda+l}}\\ &= \sum_{j=1}^{\lambda^{\tau-i}} \frac{\sum_{l=1}^{\lambda} G_{i-1,(j-1)\lambda+l}P_{i,j}/P_{i-1,(j-1)\lambda+l}}{P_{i,j}}\\ &= \sum_{j=1}^{\lambda^{\tau-i}} \frac{G_{i,j}}{P_{i,j}}.\end{aligned}$$

Therefore, $f/L = \sum_{j=1}^{\lambda^{\tau-i+1}} G_{i-1,j}/P_{i-1,j} = \sum_{j=1}^{\lambda^{\tau-i}} G_{i,j}/P_{i,j}$. ◀

Proof of Theorem 8. Correctness is ensured by the definition of the polynomials $P_{i.j}$ and $G_{i,j}$ and Lemma 7 and that all the underlying protocols, *i.e.* FastPolyMult, FastEval, **PolyDiv** and **Poly2Mult**, are correct.

Moreover, the protocol is secure since protocols FastPolyMult, FastEval, **PolyDiv** and **Poly2Mult** are all secure. Protocols FastPolyMult and FastEval require both $\mathcal{O}(\tau)$ rounds. Since we call in sequence τ times protocols **PolyDiv** and **Poly2Mult** that are constant-round, our protocol require a total of $\mathcal{O}(\tau)$ rounds of communications.

For the complexity analysis, steps 1 and 3 require $\mathcal{O}(\tau n^{1+\frac{1}{\tau}})$ secure multiplications in $\mathbb{F}_q$ according to theorems 3 and 5. The other steps except step 5 are negligible for the communication complexity. At step i of the loop, **PolyDiv** is called $\lambda^{\tau-i+1}$ times on shared dividends of the form $[P_{i,j}]$, and by Lemma 2 these polynomials are all of degree $< \lambda^i$, thus requiring $\mathcal{O}(\lambda^{\tau+1})$ secure multiplications in $\mathbb{F}_q$. At the same step, **Poly2Mult** is called $\lambda^{\tau-i+1}$ times on shared polynomials of degrees $< \lambda^i$, which requires also $\mathcal{O}(\lambda^{\tau+1})$ secure multiplications in $\mathbb{F}_q$. Overall, summing over the τ loops, step 5 requires at most $\mathcal{O}(\tau\lambda^{\tau+1}) = \mathcal{O}(\tau n^{1+\frac{1}{\tau}})$ secure multiplications in K, which concludes the proof. ◀

Proof of Theorem 9. FastPolyMult, FastEval are secure according to theorem 3 and 5. Moreover, the only information revealed is br, and since r is a uniformly random element of $\mathbb{F}_q^*$ it only indicates if G was evaluated to zero on at least one α_j, which is the intended output. Thus FastPDT is secure. Moreover, each step requires at most $\mathcal{O}(1)$ rounds of communication except for steps 5 and 6 which require $\mathcal{O}(\tau)$ rounds. Therefore, the protocol requires $\mathcal{O}(\tau)$ rounds. For the multiplication complexity, step 3 requires $\mathcal{O}(mn)$ secure multiplications since the P_i's are of degree n and there is $m-1$ polynomials. Steps 5 and 6 require $\mathcal{O}(\tau n^{1+1/\tau})$ secure multiplications. Other steps require at most a constant number of secure multiplication. In total, the protocol requires $\mathcal{O}(mn+\tau n^{1+1/\tau})$ secure multiplications. We now compute the probability of an incorrect result. If the intersection is not empty, at least one α_j will be such that $G(\alpha_j)=0$, so $b=0$, $br=0$, and parties will deduce the correct result. If the intersection is empty, then for each α_j, at least one P_i will be such that $P_i(\alpha_j)\neq 0$. Therefore $r_iP_i(\alpha_j)$ and $G(\alpha_j)$ are uniformly random over $\mathbb{F}_q$. The overall probability of an incorrect result is therefore:

$$\Pr(b=0) = \Pr\big(\bigcup_{j=1}^{n} G(\alpha_j)=0\big) \leq \sum_{j=1}^{n} \Pr(G(\alpha_j)=0) \leq n/q,$$

which is the desired result. ◀

Proof of Theorem 10. FastEval and protocols from [11] are secure and correct, thus BoolIntersection is also secure and correct. Every step is constant-round except for the calls to FastEval which requires $\mathcal{O}(\tau)$ rounds, thus the protocol requires $\mathcal{O}(\tau)$ rounds. Lastly, Step 2 requires $\mathcal{O}(\tau mn^{1+1/\tau})$ secure multiplication since it consists of calling FastEval $m-1$ times. Step 3 requires $\mathcal{O}(nm\cdot\log n\log\log n)$ secure multiplications since it consists of calling the conversion protocol from [11] $n(m-1)$ times. Step 4 requires $\mathcal{O}(nm)$ secure multiplications since it consists of calling the protocol from [11, Section 5.1] n times. Overall, the protocol requires $\mathcal{O}(mn\log n\log\log n+\tau mn^{1+1/\tau})$ secure multiplications. ◀