View HTML Version

Thesis Proposal

Linsheng Liu

December 1, 2025

Contents

Abstract

This dissertation investigates how to reduce communication overhead in privacy-preserving applications. It develops two complementary directions toward this goal. The first studies how sublinear communication protocols can be achieved for tasks such as secure search over encrypted datasets, secure collaborative sampling protocols, and sublinear approximation algorithms such as Min-Hash for set similarity. These techniques serve as common building blocks across a wide range of privacy-preserving applications. Together, these results reveal a unified perspective on how limited communication and provable privacy reinforce each other across modern cryptographic and data-analytic settings. The second direction extends these to practical systems such as the FACTS framework for privacy-preserving accountability on end-to-end encrypted messaging systems (EEMS). This work enables threshold tracebacks to identify the originator of reported messages while the cost depends only on the number of reports, and includes preliminary designs and analysis for extending these mechanisms to identify the super-spreader of reported messages.

1 Introduction

The modern digital landscape is defined by a fundamental tension between the exponential growth of data and the imperative to protect the privacy of the individuals generating it. As datasets scale into the terabytes and petabytes, and as digital communication becomes the primary medium for societal discourse, the computational paradigms that served the early internet are reaching their limits. Traditional cryptographic protocols for privacy-preserving applications, such as Secure Multi-Party Computation (MPC) and Fully Homomorphic Encryption (FHE), offer mathematically robust guarantees of confidentiality. However, these protocols typically exhibit linear or super-linear complexity with respect to the input size ($O(n)$). In a world of massive datasets, linear complexity is often synonymous with infeasibility. A query that requires touching every record in a billion-row database, even if cryptographically secure, remains practically useless if it takes days to execute.

Consequently, one important frontier of cryptographic research has shifted toward the development of **sublinear algorithms**—protocols whose complexity is logarithmic, constant with respect to the input size, or dependent only on the output size. This thesis proposal report synthesizes four seminal contributions that pioneer this shift across two distinct but interrelated domains.

The first domain, **Sublinear Secure Protocols**, comprises three studies that attack the fundamental algorithmic bottlenecks of secure computation. These works introduce a new data structure for compressing large sparse data vectors for homomorphically encrypted search to allow for parallel, sublinear retrieval, develop protocols for secure sampling from distributed distributions without linear communication, and investigate the inherent privacy properties of sketching algorithms (Min-Hash).

The second domain, designated as **FACTS**, addresses the societal crisis of misinformation within end-to-end encrypted messaging systems (EEMS). Here, the challenge is not merely computational efficiency but the architectural reconciliation of user privacy with platform accountability. The proposed solution, the Fuzzy Anonymous Complaint Tally System (FACTS), leverages a novel probabilistic data structure to achieve sublinear scalability in complaint auditing, proving that privacy and moderation need not be mutually exclusive.

A unifying theme across both groups is the strategic utilization of **approximation** and **probabilistic correctness**. Whether it is the fuzzy counting of complaints in FACTS, the rejection sampling in secure estimation, or the false-positive rates of compressed encodings in search, these works demonstrate that controlled relaxation of exactness is the key to unlocking sublinear performance. This introduction delineates the theoretical underpinnings and practical implications of these advancements, setting the stage for a detailed technical analysis.

1.1 The Linear Bottleneck in Secure Computation

To understand the necessity of the contributions discussed herein, one must first appreciate the "Linearity Wall." In standard secure multi-party computation, if Party A holds a vector x and Party B holds a vector y, computing a function $f(x, y)$ usually requires processing every bit of the inputs to avoid leaking information about which bits were "useful". For example, in Private Information Retrieval (PIR), the server must process the entire database to answer a single query; otherwise, the server learns which items were *not* touched, narrowing down the user's interest.

While techniques like Oblivious RAM (ORAM) can hide access patterns, they impose logarithmic overheads that, while asymptotically better than linear scanning for multiple accesses, still require significant bandwidth and state management. The research in Sublinear Secure Protocols specifically targets scenarios where even these overheads are unacceptable, aiming for protocols where communication depends on the *output size* or the *security parameter*, rather than the input database size.

1.2 Sublinear Secure Protocols: Primitives for Sublinear Privacy

This line of research explores the building blocks of sublinear privacy.

- **Encrypted Search:** The first paper tackles the sequential bottleneck in FHE search. Prior systems required a round of interaction for every matching record found. The authors propose **Compressed Oblivious Encoding (COE)**, allowing a server to compress all search results into a single package. This reduces the fetching time by orders of magnitude (1800x in experiments), effectively making FHE search viable for real-time applications.

- **Secure Sampling:** The second paper addresses the problem of sampling from joint distributions (e.g., L_1, L_2, Product) partitioned across users. It proves that while general product sampling is impossible with sublinear communication, specific correlated inputs allow for efficient protocols. The introduction of **Corrective Sampling**—a technique that corrects a proxy distribution into a target distribution without computing normalization factors—allows for constant-round sampling protocols.

- **Min-Hash Privacy:** The third paper in this group interrogates the "folklore" that sketching algorithms like Min-Hash are inherently private. It rigorously proves that while sketches compress data (sublinear representation), they do not automatically guarantee Differential Privacy (DP) in the public hash setting. The authors introduce **Distributional Differential Privacy (DDP)** and **Noisy Min-Hash** to bridge this gap, allowing for constant-size comparisons of massive sets with formal privacy guarantees.

Collectively, these works argue that the future of privacy-preserving technologies lies in the domain of sublinear algorithms, where mathematical approximation provides the necessary slack to achieve scalability.

1.3 FACTS: Accountability via Probabilistic Structures

This line of research focuses on the application of "fighting fake news". end-to-end encrypted messaging systems (EEMS) platforms like WhatsApp and Signal utilize encryption that hides message content from the service provider. This prevents any content moderation methods that rely on analyzing the content in the clear, such as tools used by platforms like Facebook or Twitter. The FACTS system introduces a paradigm where moderation is triggered only by a consensus of user complaints.

The core innovation here is the **Collaborative Counting Bloom Filter (CCBF)**. This data structure allows the system to tally complaints against millions of messages without the server knowing which complaints correspond to which message until a threshold is crossed. This is achieved by "mixing" counters in a shared bit array. The efficiency is sublinear in the number of messages in the system per epoch: identifying a message for audit requires no linear scan of the database, and registering a complaint requires flipping a single bit. This work exemplifies how probabilistic data structures—traditionally used for efficiency in networking–can be repurposed as privacy primitives.

2 Literature Review

This section surveys existing literature relevant to the two primary lines of work. First, we discuss the foundational cryptographic primitives and protocols for **Secure Search, Secure Sampling, and Privacy-Preserving Set Similarity**. Second, we review the literature relevant to our specific application in accountability systems of end-to-end encrypted messaging systems, **FACTS**.

2.1 Line 1: Secure Search, Sampling, and Similarity

This line of work focuses on optimizing privacy-preserving protocols for retrieving and analyzing data. We divide the literature into three interconnected domains: secure search, secure sampling, and set similarity.

Differential privacy (DP). Differential privacy limits what an adversary can learn about any individual input from the output of a computation [60, 63]. For an overview of DP and standard mechanisms in both the curator setting and distributed settings, we refer the reader to the book by Dwork and Roth [66]. DP will appear repeatedly in our discussion, both as an explicit privacy goal and as a tool for trading accuracy for improved efficiency.

2.1.1 Secure Search

Secure search is a widely-studied problem with solutions spanning various cryptographic settings. In the discussion below, let n denote the number of data items.

Secure pattern matching (SPM) on FHE-encrypted data. In SPM, given an encrypted query $[\![q]\!]$ and n FHE-encrypted data items $([\![x_1]\!], \ldots, [\![x_n]\!])$, the protocol returns a vector of n ciphertexts $[\![b_1]\!], \ldots, [\![b_n]\!]$, where b_i indicates whether the i-th data element matches the query [43, 44, 123, 187]. These works primarily focus on optimizing the search circuits used to determine matches. Consequently, the communication complexity and the client's running time remain proportional to the number of data items. In contrast, our work focuses on the orthogonal problem of optimizing the *retrieval* of matched data items, aiming for sublinear communication and client computation.

Searchable encryption (SE). Searchable encryption [29, 167] enables highly efficient search (typically $o(n)$ time) over encrypted data. Efficient SE schemes have been proposed for a wide variety of queries, including equality queries [41, 54], range queries [107, 157], and conjunctive queries [37, 147]. However, to achieve sublinear query performance, SE schemes generally require significant preprocessing and must relax security guarantees, allowing partial information (such as access patterns) to leak to the server [82]. Unlike standard SE, our work focuses on achieving preprocessing-free constructions that leak nothing about the queries or results beyond their sizes.

Property Preserving Encryption (PPE). PPE [146] produces ciphertexts that maintain certain relationships (e.g., equality or order) of the underlying plaintexts. Examples include deterministic encryption [19] and order-preserving encryption [27, 28]. However, it has been demonstrated [96, 97, 109] that such ciphertexts leak significant information, rendering them undesirable for many security-critical applications.

General Techniques (PIR, MPC, ORAM, ODS). Private Information Retrieval (PIR) [47] allows retrieval while hiding the index, but requires the client to know the index beforehand. General MPC [186] and ORAM [89] can theoretically solve secure search, but generic constructions typically incur high communication ($\Omega(n)$ for MPC) or significant overhead to hide access patterns. Oblivious data structures (ODS) can support richer data structures (e.g., search trees), but typical ODS constructions incur $\Omega(\log^2 n)$ rounds per operation, which motivates constant-round designs for practical secure search.

2.1.2 Secure Sampling

Sampling from streaming data. Non-private sampling from data streams has been studied extensively [49, 83, 115, 140, 182], typically achieving ℓ_p sampling with sublinear computation. These works generally operate in a single-party setting and do not consider privacy.

Secure multiparty sampling. In the privacy domain, works like [153, 154] investigate sampling in the information-theoretic setting, while Champion et al. [39] consider the computational setting for publicly-known distributions. Our work is distinct in focusing on sampling from a *private* distribution in the computational setting, with a specific emphasis on reducing communication.

Secure MPC of differentially private functionalities. Since Dwork et al. [62], significant work has focused on using MPC to realize DP functionalities [1, 48, 70]. While these works address machine learning and aggregation, we focus specifically on optimizing the communication complexity for sampling functionalities.

2.1.3 Private Sketching and Secure Sketching

Secure sketching. A long line of work studies secure sketches for estimating statistics (e.g., Tor usage, web traffic, unique count, median) with sublinear communication and computation [45, 69, 112, 136, 178]. These works are closely related in spirit, but generally focus on building secure protocols for specific streaming-style statistics, whereas our focus is on sublinear primitives (search, sampling, and similarity) that can serve as broadly reusable building blocks.

Private sketching. Sketching algorithms are sublinear-space methods that produce compact summaries enabling efficient storage, merging, and processing. A growing body of work observes that sketches can also aid privacy, since information loss in the sketch can make the sketch itself differentially private or reduce the additional noise required [15, 16, 24, 46, 56, 101, 104, 127, 137, 138, 145, 165, 179, 189]. Related work also constructs private sketches for set cardinality and set operations, including mergeable sketches for estimating intersection and union [101, 124, 142, 144, 168, 169]. This perspective directly motivates our study of privacy properties of Min-hash and related similarity sketches.

2.1.4 Set Similarity via Min-Hash

Jaccard Index and Min-Hash. Many works have constructed 2-party Jaccard index estimation protocols using Min-hash with sublinear communication to avoid the high cost of exact computation [52, 72, 156].

Differentially Private (DP) Min-Hash. Recent research aims to output Jaccard similarity estimates while preserving differential privacy. Aumüller et al. [10] achieve local DP min-hash by perturbing vectors with noise. Other attempts have faced challenges; for example, [185] was shown to have flaws in its privacy claim, while [184] incurs high noise overhead.

Optimizing Secure Computation using DP. Our work sits at the intersection of DP and MPC. A specific line of research relevant to our goals is *optimizing secure computation using DP*, initiated by Beimel et al. [17]. Subsequent works have applied this to set intersection [94,100], graph-parallel computations [133], and shuffling [92]. We similarly leverage DP-style relaxations to improve the efficiency of similarity protocols.

Secure approximation. Secure approximation studies what functions can be securely approximated without revealing anything beyond the true output [76,99]. This notion is distinct from DP-style approximation, but it is conceptually adjacent: both investigate how controlled relaxation can enable more efficient privacy-preserving computation.

Robust sketching and property-preserving hashing. Property-preserving hash (PPH) compresses large inputs into short digests enabling computation of a property from the digests alone, and adversarially robust PPH further requires correctness even when inputs are adaptively chosen after the hash is fixed [30, 80, 81, 103]. Related work on robust sketching similarly studies sketches that remain accurate under adaptive inputs [9, 20]. These works focus on robustness to adversarial inputs, whereas our focus is on privacy when the adversary additionally sees the hash functions or sketch randomness.

2.2 Line 2: FACTS (Forwarding Accountability in End-to-End Encrypted Messaging Platforms)

Our second line of work shifts focus to a specific application: enabling accountability in end-to-end encrypted messaging systems.

Message Franking. The prevailing approach for reporting malicious messages is *message franking* [57,95,176]. This technique allows a recipient to cryptographically prove the identity of the sender. However, message franking is limited to identifying the *immediate* sender and cannot trace the original source in a forwarding chain. Furthermore, it does not provide threshold guarantees to prevent unmasking users based on a single complaint.

Scalable Oblivious Data Structures. FACTS requires scalable oblivious storage to track complaints. While multi-client ORAM protocols exist [38, 102, 131], they do not yet scale to the millions of users required for messaging applications. Similarly, oblivious counters [86, 120] generally focus on exact counting or complex operations, lacking the specific compression traits of the Compact Count-Min Bloom Filter (CCBF) we utilize. More generally, oblivious data structures [122, 161, 180] enable higher-level oblivious operations over encrypted data, but do not directly provide the compression needed to store and update complaint tallies at large scale.

Private Sketching for User-Server Settings. CCBF can be viewed as a compact *sketch* for storing complaint counts over a large set of messages. There has been significant interest in privacy-preserving sketching algorithms for cardinality estimation, frequency measurement, and related approximations [45, 178]. However, much of this literature targets multi-party settings (often with multiple servers) where parties run secure computation to evaluate the statistic. In contrast, FACTS restricts communication to the user–server model, which limits the direct applicability of these approaches.

3 Secure Search

3.1 Introduction

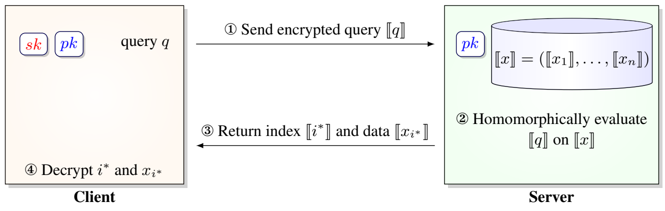
As computing paradigms are shifting to cloud-centric technologies, users of these technologies are increasingly concerned with the privacy and confidentiality of the data they upload to the cloud. Specifically, a *client* uploads data to the *server* and expects the following guarantees:

1. The uploaded data should remain private, even from the server itself;
2. The server should be able to perform computations on the uploaded data in response to client queries;
3. The client should be able to efficiently recover the results of the server's computation with minimal post-processing.

In this work, we will focus on the computational task of secure search. In this application, the client uploads a set of records to the server, and later posts queries to the server. Computation proceeds in two steps called *matching* and *fetching*. In the matching step, the server compares the encrypted search query from the client with all encrypted records in the database, and computes an encrypted $0/1$ vector, with 1 indicating that the corresponding record satisfies the query. The fetching step returns all the 1-valued indexes and the corresponding records, to the client for decryption.

While seemingly conflicting goals, the guarantees of (1), (2), (3) can be simultaneously achieved for the secure search setting via techniques such as secure multiparty computation and searchable encryption. Recently, a line of works has focused on Fully Homomorphic Encryption (FHE)-based secure search, which we describe next.

FHE-based secure search. The simplicity of the framework of *secure search on FHE encrypted data* is attractive. Compared to other secure search systems, no costly setup procedure is necessary; it is sufficient for the client merely to upload the encrypted database to the server. Confidentiality is provided because the server works only on the encrypted query and records. The server can still perform the search correctly due to the powerful property of the full homomorphism of the underlying encryption scheme.



sk
pk
query q
① Send encrypted query $[\![q]\!]$
pk
$[\![x]\!] = ([\![x_1]\!], \ldots, [\![x_n]\!])$
② Homomorphically evaluate $[\![q]\!]$ on $[\![x]\!]$
③ Return index $[\![i^*]\!]$ and data $[\![x_{i^*}]\!]$
④ Decrypt i^* and x_{i^*}
Client
Server

In the above, $[\![\cdot]\!]$ denotes an FHE-encrypted ciphertext.

Figure 1: The secure search framework in [2]

For this reason, researchers have been paying increasing attention to this problem. In particular, Akavia et al. [2] introduce a framework of performing secure search on FHE-encrypted data (see Figure 1).

Informally, a secure, homomorphic encrypted search scheme has the following Setup:

1. (Setup) The client encrypts and uploads n items $x = (x_1, \ldots, x_n)$ to the server. Let $[\![x]\!] = ([\![x_1]\!], \ldots, [\![x_n]\!])$. denote the encrypted data stored in the server.

Throughout the paper, we let $[\![\cdot]\!]$ denote an FHE-encrypted ciphertext. After the encrypted records have been uploaded, the client can perform a secure search using three algorithms, (Query, Match, Fetch).

2. (Query) The client sends an encrypted query $[\![q]\!]$ to the server.

3. (Match) The server homomorphically evaluates the query $[\![q]\!]$ on each record $[\![x_i]\!]$ to obtain the encrypted matching results $[\![b]\!] = ([\![b_1]\!], \ldots, [\![b_n]\!])$. That is, b_i is 1 if item x_i satisfies the given query q; otherwise, b_i is 0.

4. (Fetch) Given $[\![b]\!]$, the server homomorphically computes $[\![i^*]\!]$, where $i^* = \min\{i \in [n] : b_i = 1\}$ which corresponds to the first matching record index. It fetches $[\![x_{i^*}]\!]$ (obliviously) and sends $([\![i^*]\!], [\![x_{i^*}]\!])$ to the client for decryption.

Multiplications in the fetching step. Akavia et al. also provide a construction that performs the fetching step in $O(n \log^2 n)$ homomorphic multiplications. Subsequently, more efficient algorithms have been presented with $O(n \log n)$ multiplications [3] and $O(n)$ multiplications [181].

3.1.1 Motivation

Bottleneck: fetching records sequentially. Suppose a client wants to fetch all matching items. Under the above framework, the client would first obtain the first matching index i^* and its corresponding item x_{i^*}. To fetch the second matching item, the framework suggests that the client should slightly change the original query q to a new query q'_{i^*} as follows:

- $q'_{i^*}(i, x_i)$ return true if $q(i, x_i)$ is true and $i > i^*$.

Then, by executing a new instance of the protocol with the encrypted query $[\![q'_{i^*}]\!]$, the client will obtain the second matching item. By repeating this procedure, the client will ultimately obtain all the matching records.

Note that the query q'_{i^*} embeds i^* in itself as a constant, which implies that there is no way for the client to construct this query q'_{i^*} without obtaining i^* first. In other words, the client can construct the query for the second matching item, *only after fetching the first matching item.* In this sense, the framework inherently limits the client to fetch only a single matching record at a time in a sequential manner.

If there are ℓ matching records, the client and server have to execute ℓ instances of the Query, Match, and Fetch algorithms. Since each Match and Search step requires costly homomorphic multiplications, the limitation of sequential protocol execution creates a serious bottleneck with respect to the running time. This leads us to ask the following natural question:

> *Is there a different secure search framework that allows the client to fetch all the matching records by executing a smaller number of protocol executions, possibly avoiding sequential record fetching?*

Reducing homomorphic multiplications. All previous schemes have to perform $\Omega(n)$ homomorphic multiplications in the fetching step. Since homomorphic multiplications are costly operations, it is desirable to reduce such computations, which begs the natural following question:

> *Can you reduce the number of homomorphic multiplications in the fetching step?*

In this paper, we answer both of the above questions affirmatively.

3.1.2 Our Work

Parallelizing the Fetch procedure. To address the issues, we introduce a new secure search framework where the matching items are retrieved *in parallel* in a constant number of rounds. Our Setup, Query and Match algorithms are the same as in prior work. However, we modify the Fetch procedure, dividing into two steps: Encode and Decode. In the Encode step, the server homomorphically inserts the matching items into a data structure - the particular structure depends on the construction, as we provide 3 different constructions, each using a different encoding. After receiving the encrypted encoding, the client decrypts the encoding and runs the Decode step to recover the items.

Compressed oblivious encoding. The encoding is computed homomorphically, and, most importantly, allows to encode *the full result set*, rather than just a single item. In particular, we introduce a notion of Compressed Oblivious Encoding (COE). A compressed oblivious encoding takes as input a large,

	rounds	#Match	hmult	hadd	smult	communication	p
LEAF [181]	s	s	$O(ns)$	$O(ns \log n)$	0	$O(s \cdot \log n \cdot \|C\|)$	
Protocol w/ BF-COIE	3	1	0	$O(n \log \frac{n}{s})$	0	$O(s^{1+\epsilon} \log \frac{n}{s} \cdot \|C\| + pir(s))$	
Protocol w/ PS-COIE	3	1	0	$n \cdot s$	$n \cdot s$	$O(s \cdot \|C\| + pir(s))$	
Protocol w/ BFS-CODE	2	1	n	$O(\kappa n)$	0	$O(s\kappa \cdot \|C\|)$	

- κ: statistical security parameter.
- n: number of uploaded encrypted records.
- s: number of matching records.
- ϵ: protocol parameter such that $0 < \epsilon < 1$.
- #Match: number of times the matching algorithm is executed.
- hmult: number of homomorphic multiplication operations used in the overall fetching step.
- hadd: number of homomorphic addition operations used in the overall fetching step.
- smult: number of scalar (plain) multiplication operations used in the overall fetching step.
- $|C|$: length of an FHE ciphertext.
- $pir(s)$: communication complexity required to retrieve s records via a PIR protocol.

Figure 2: Performance Comparisons when s records are fetched

but sparse, vector and compresses it to a much smaller encoding from which the non-zero entries of the original vector can be recovered. What makes this encoding *oblivious* is that the encoding procedure is performed on encrypted data. In certain constructions, the encoding includes the data values (CODE, compressed oblivious data encoding), and in others it only includes the indices (COIE, compressed oblivious index encoding). In the latter case, the Decode procedure is interactive, and allows the client to recover the values from the decoded set of indices.

For simplicity, when describing the generic syntax of secure search scheme, we denote the Encode procedure as taking both the indices and the values as input, and we suppress the fact that when the values are not used during Encoding, the Decoding step must be interactive. Recall, we use $[\![b]\!] = ([\![b_1]\!], \ldots, [\![b_n]\!])$ to denote the encrypted bit vector that results from the Match step.

4. (Encode) Let $S = \{i \in [n] : b_i = 1\}$. Let $V = \{v_i : i \in S\}$. The server homomorphically evaluates an $[\![\mathsf{encoding}(S, V)]\!]$ and send it to the client.

5. (Decode) The client decrypts $[\![\mathsf{encoding}(S)]\!]$ and runs the decoding procedure to recover (S, V).

We assume that the results set $|S|$ is small (i.e., sublinear in n). We would like the size of the compressed encoding to be sublinear in n to maintain meaningful communication cost.

No multiplications in the Encode step. To ensure minimal computational cost for encoding the results, we also wish to minimize the number of homomorphic multiplications. Recall, the best prior work requires $O(n)$ multiplications by the server. Somewhat surprisingly, we demonstrate three encoding algorithms that can be evaluated *without* any homomorphic multiplications!

Using PIR (Private Information Retrieval). The asymptotic complexities and trade-offs of the search protocols are presented in Figure 2.

In some of our protocols (i.e., the search protocols with BF-COIE and PS-COIE; see Sections 4 and 6.3 for more detail), the indices and actual records are fetched in separate steps. This allows us to focus on optimizing the retrieval of the indices after which the values can be fetched using an efficient (setup-free) PIR protocol resulting in overall savings.

However, if reliance on PIR is undesirable, we also offer a variant that fetches the values directly (i.e., the protocol w/ BFS-CODE in Figure 2; see Sections 5 and 6.4 for more detail), as in prior work.

Implementation. We implement all of our proposed schemes and compare their performance with that of prior work. Our experiments show that our schemes outperform the fetching procedure of prior work by a factor of 1800X when fetching 16 records, which results in a 26X speedup for the full search functionality.

3.2 Preliminaries

Let κ be the security parameter. For a vector a, let $\mathsf{idx}(a)$ denote the set of all the positions i such that a_i is non-zero, i.e.,

$$\mathsf{idx}(a) := \{i : a_i \neq 0\}.$$

Chernoff bound. We will use the following version of Chernoff bound.

Theorem 3.1. *Let $X_1, \ldots, X_n$ be independent random variables taking values in $\{0,1\}$ such that $\Pr[X_i = 1] = p$. Let $\mu := \mathbf{Exp}[\sum X_i] = np$. Then for any $\delta > 0$, it holds*

$$\Pr\left[\sum_{i=1}^{n} X_i \geq (1+\delta)\mu\right] \leq \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu}.$$

FHE. We use a standard CPA-secure (leveled) fully homomorphic encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. We refer readers to [3, 181] for a formal definition. We use $[\![x]\!]$ to denote an encryption of x.

We also use $+$ (resp. $\cdot$) to denote homomorphic addition (resp., multiplication). For example, $[\![c]\!] := [\![a]\!] + [\![b]\!]$ means that homomorphic addition of two FHE-ciphertexts $[\![a]\!]$ and $[\![b]\!]$ has been applied, which results in $[\![c]\!]$.

PIR. A PIR protocol allows the client to choose the index i and retrieve the ith record from one (or more) untrusted server(s) while hiding the index value i [47].

Assume that each of the k server has n records $D = (d_1, \ldots, d_n)$ where all items d_i have equal length. A single-round k-server PIR protocol consists of the following algorithms:

- The query algorithms $Q_j(i, r) \to q_j$ for each server $j \in [k]$, which are executed by the client with input index i and randomness r.
- The answer algorithms $A_j(D, q_j) \to a_j$ for each server $j \in [k]$, which is executed by the jth server.
- The reconstruction algorithm $R(i, r, (a_1, \ldots, a_k)) \to d_i$.

The communication complexity of a PIR protocol is defined by the sum of the all query lengths and answer lengths, i.e.,

$$\sum_{j \in [k]} |q_j| + |a_j|.$$

A PIR protocol is correct if for any $D = (d_1, \ldots, d_n)$ with $|d_1| = \cdots = |d_n|$, and for any $i \in [n]$, it holds that

$$\Pr_r\left[R\Big(i, r, \big\{A_j(D, Q_j(i, r))\big\}_{j=1}^{k}\Big) = d_i\right] = 1.$$

A PIR protocol is private if for any $j \in [k]$, for any $i_0, i_1 \in [n]$ with $i_0 \neq i_1$, the following distributions are computationally (or statistically) indistinguishable:

$$\{Q_j(i_0, r)\}_r \approx \{Q_j(i_1, r)\}_r.$$

3.2.1 Bloom Filter

A Bloom filter [25] is a well-known space-efficient data structure that allows a user to insert arbitrary keywords and later to check whether a certain keyword in the filter.

$\mathsf{BF.Init}()$**.** The filter B is essentially an ℓ-bit vector, where ℓ is a parameter, which is initialized with all zeros. The filter is also associated with a set of η different hash functions

$$\mathcal{H} = \{h_q : \{0,1\}^* \to [\ell]\}_{q=1}^{\eta}.$$

$\mathsf{BF.Insert}(\mathsf{B}, \alpha)$**.** To insert a keyword α, the hash results are added to the filter. In particular,

- For $q \in [\eta]$ do the following:

  Compute $j = h_q(\alpha)$ and set $B_j := 1$. Here B_j is the jth bit of B.

$\mathsf{BF.Check}(\mathsf{B}, \beta)$**.** To check whether a keyword β has been inserted to a BF filter B, one can just check the filter with all hash results. In particular,

- For $q \in [\eta]$ do the following:

  Compute $j = h_q(\beta)$ and check if B_j is set.

- If all checks pass output "yes". Otherwise, output "no".

The main advantage of the filter is that it guarantees there will be no false negatives and allows a tunable rate of false positives:

$$\left(1 - \left(1 - \frac{1}{\ell}\right)^{\eta s}\right)^{\eta} \approx \left(1 - e^{-\frac{\eta s}{\ell}}\right)^{\eta},$$

where s is the number of keywords in a Bloom filter.

Random oracle model for hash functions. We show our analysis in the random oracle model. That is, the hash functions are modelled as random functions.

3.2.2 Algebraic Bloom Filter

In this work, we leverage a variant of the Bloom filter where, when inserting an item, the bit-wise OR operation is replaced by addition. There have been works using a similar idea of having each cell hold an integer instead of holding a bit [75, 139].

Moreover, we consider a limited scenario *where the upperbound on the number of keywords to be inserted is known beforehand.* In particular, let s denote such an upperbound.

As before, the filter is also associated with a set of η different hash functions $\mathcal{H} = \{h_q : \{0,1\}^* \to [\ell]\}_{q=1}^{\eta}$. However, now the filter B is not an ℓ-bit vector but a vector where each element is in $[s\eta]$ (i.e., $B \in [s\eta]^{\ell}$) [1]. Therefore, the number of bits to encode B is now blown up by a multiplicative factor $\lceil \lg s\eta \rceil$.

The BF operations are described below where differences are marked by framed boxes.

$\mathsf{BF.Insert}(\mathsf{B}, \alpha)$**.** To insert a keyword α, the hash results are added to the filter. In particular,

- For $q \in [\eta]$ do the following:

  Compute $j = h_q(\alpha)$ and set $\boxed{B_j := B_j + 1}$.

$\mathsf{BF.Check}(\mathsf{B}, \beta)$**.** To check whether a keyword β has been inserted to a BF filter B, one can just check the filter with all hash results. In particular,

- For $q \in [\eta]$ do the following:

  Compute $j = h_q(\beta)$ and check if B_j is $\boxed{\text{greater than } 0}$.

- If all checks pass output "yes". Otherwise, output "no".

It is easy to see that this variant construction enjoys the same properties as the original BF construction.

3.3 Compressed Oblivious Encoding

As our main building block, we introduce a new tool we call Compressed Oblivious Encoding. A compressed oblivious encoding takes as input a large, but sparse, vector and compresses it to a much smaller encoding from which the non-zero entries of the original vector can be recovered. What makes this encoding *oblivious* is that the encoding procedure is oblivious to the original data; in fact, in our constructions the original data will all be encrypted. An efficient encoding must satisfy the following two performance requirements: 1) The size of the encoding must be sublinear in the size of the original array, and 2) constructing the encoding should be computationally cheap. Our constructions only use (homomorphic) addition and multiplication by constant (i.e. plaintext values).

A related notion is that of compaction over encrypted data [7, 23] which aims to put all non-zero entries of a vector to the front of the encoding. Our encoding can be viewed as a form of noisy compaction where, in addition to keeping all the non-zero entries, it allows a small number zero entries to be mixed in with the result. Thus, a compressed encoding trades some inaccuracy in the output for much cheaper construction costs.

We define two variants of compressed oblivious encodings, one that encodes the indices of non-zero entries and one that encodes the actual entries themselves.

[1]We can reduce $s\eta$ further to $\Theta(\eta \cdot (s/\ell) \cdot \log(s/\ell))$ using a Chernoff bound to bound the number of collisions contributing to the sum, but we will use $s\eta$ for the sake of simplicity of presentation.

3.3.1 Compressed Oblivious Index Encoding

A compressed oblivious index encoding (COIE) encodes the indices or locations of all the non-zero entries in the input array. We begin by defining the parameters and syntax for a COIE scheme.

Parameters. A COIE scheme is parametrized as follows.

- n: Input size – The dimension of the input vector v.
- s: Sparsity – Bound on the number on non-zero entries in v.
- c: Compactness – The dimension of the output encoding.
- f_p: False positives – The upperbound on the number of false positives returned by the decoding algorithm.

Syntax. A (n, s, c, f_p)-COIE scheme has the following syntax:

- $[\![\gamma_1]\!], \ldots, [\![\gamma_c]\!] \leftarrow \mathsf{Encode}([\![v_1]\!], \ldots, [\![v_n]\!])$. The Encode algorithm takes as input a vector of ciphertexts with $v_i \in \{0, 1\}$ for all $i \in [n]$. It outputs an encrypted encoding $[\![\gamma_1]\!], \ldots, [\![\gamma_c]\!]$.
- $I \leftarrow \mathsf{Decode}(\gamma_1, \ldots, \gamma_c)$. The Decode algorithm takes the encoding $(\gamma_1, \ldots, \gamma_c)$, in decrypted form, and outputs a set $I \subseteq [n]$

Correctness. Let $(\gamma_1, \ldots, \gamma_c) \leftarrow \mathsf{Dec}([\![\gamma_1]\!], \ldots, [\![\gamma_c]\!])$ denote a correct decryption of the encoding.

Definition 3.2. A (n, s, c, f_p)-COIE scheme is *correct*, if the following conditions are satisfied:

- (No false negatives) For all $v \in \{0, 1\}^n$ with at most s non-zero positions, and for all $i \in \mathsf{idx}(v)$, it should hold

$$i \in \mathsf{Decode}(\mathsf{Dec}(\mathsf{Encode}([\![v_1]\!], \ldots, [\![v_n]\!])))$$

  with probability at least $1 - \mathsf{negl}(\kappa)$ where the random coins are taken from Encode.

- (Few false positives) For all $v \in D^n$ with at most s non-zero positions, consider the set of false positives

$$E = \{i \in [n] : v_i = 0, \text{ but } i \in I\},$$

  where $I = \mathsf{Decode}(\mathsf{Dec}(\mathsf{Encode}([\![v_1]\!], \ldots, [\![v_n]\!])))$.

  We require that $|E| \leq f_p$ with the overwhelming probability over the randomness of Encode.

Efficiency. For efficiency, we look at the following three parameters of a COIE:

- The type and number of operations used by the Encode algorithm.
- The size of the encoding.
- The computation cost of the Decode algorithm.

For an efficient construction, we require that the latter two of these are sublinear in the size of the input vector.

3.3.2 Compressed Oblivious Data Encoding

A Compressed Oblivious Data Encoding (CODE) scheme is very similar to COIE except, rather than encoding the locations of non-zero entries, it encodes the values of these entries. We give a definition of CODE below where differences are marked by framed boxes.

Parameters. A CODE scheme is parametrized by the same four parameters (n, s, c, f_p) as a COIE.

Syntax. A (n, s, c, f_p)-CODE scheme over $\boxed{\text{domain } D}$ has the following syntax:

- $[\![\gamma_1]\!], \ldots, [\![\gamma_c]\!] \leftarrow \mathsf{Encode}([\![v_1]\!], \ldots, [\![v_n]\!])$. The Encode algorithm takes as input a vector of ciphertexts with $v_i \in \boxed{D}$ for all $i \in [n]$. It outputs an encrypted encoding $[\![\gamma_1]\!], \ldots, [\![\gamma_c]\!]$.
- $\boxed{V} \leftarrow \mathsf{Decode}(\gamma_1, \ldots, \gamma_c)$. The Decode algorithm takes the encoding $(\gamma_1, \ldots, \gamma_c)$, in decrypted form, and outputs a set of values $\boxed{V = \{v_i : v_i \neq 0\}}$

Correctness.

Definition 3.3. A (n, s, c, f_p)-CODE scheme over domain D is correct, if the following conditions are satisfied:

- (No false negatives) For all $v \in \{0,1\}^n$ with at most s non-zero positions, and for all $i \in \mathsf{idx}(v)$, it should hold

$$\boxed{v_i \in \mathsf{Decode}(\mathsf{Dec}(\mathsf{Encode}([\![v_1]\!], \ldots, [\![v_n]\!])))}$$

  with probability $1 - \mathsf{negl}(\kappa)$ where the random coins are taken from Encode.

- (Few false positives) For all $v \in D^n$ with at most s non-zero positions, consider the set of false-positive values

$$\boxed{E = \{z \in V : z \neq v_i \text{ for any } i \in \mathsf{idx}(v)\}},$$

  where $V = \mathsf{Decode}(\mathsf{Dec}(\mathsf{Encode}([\![v_1]\!], \ldots, [\![v_n]\!])))$.

  We require $|E| \leq f_p$ with the overwhelming probability over the randomness of Encode.

3.4 COIE Schemes

We assume the input index vector $v \in \{0,1\}^n$ is sparse. In particular, throughout the paper, we assume $s = o(n)$.

3.4.1 A Warm-up construction

Using an algebraic BF, we can create an (n, s, c, f_p)-COIE scheme (the parameters c and f_p will be worked out after the description of the scheme).

$\mathsf{Encode}(\llbracket v_1 \rrbracket, \ldots, \llbracket v_n \rrbracket)$**.** The encoding algorithm works as follows:

1. Initialize a BF $\llbracket B \rrbracket := (\llbracket B_1 \rrbracket, \ldots, \llbracket B_c \rrbracket)$ with $B_j = 0$ for all j. Let $\mathcal{H} = \{h_q : \{0,1\}^* \to [c]\}_{q=1}^{\eta}$ be the associated hash functions.
2. For $i = 1, \ldots, n$:
   (a) For $q = 1, \ldots, \eta$, do the following: Compute $j = h_q(i)$ and set $\llbracket B_j \rrbracket := \llbracket B_j \rrbracket + \llbracket v_i \rrbracket$.

Note that at step 2.a in the above, if $v_i = 0$, then B_j stays the same. On the other hand, if $v_i = 1$, then B_j will be increased by 1. This implies that B will exactly store the results of the operations $\{\mathsf{BF.Insert}(B, i) : i \in \mathsf{idx}(v)\}$.

$\mathsf{Decode}(B_1, \ldots, B_c)$**.** Given the algebraic BF B, we can recover the indices for the nonzero elements as follows:

- Initialize I to be the empty set.
- For $i \in [n]$: if $\mathsf{BF.Check}(B, i) =$ "yes", add i to I.
- return I.

Parameters c and f_p. Since this is a warm-up construction, we perform only a rough estimation on the false positive parameter and the compactness parameter.

For reasons that will become clear later, we wish to keep the upper bound on the number of false positives (f_p) small. In particular, we use a BF with false-positive rate $1/n$. Since there are n operations of $\mathsf{BF.Check}$, the expected number of false positives is 1, and from the Chernoff bound, the number of false positives is bounded by $\Omega(\log \kappa)$ with overwhelming probability in κ. This implies that we have $f_p = \Omega(\log \kappa)$.

The dimension c of the Bloom filter B can be computed using the following equation of BF false positive ratio:

$$\left(1 - e^{-\frac{\eta s}{c}}\right)^{\eta} \leq \frac{1}{n},$$

Setting $c = \eta s \cdot n^{\frac{1}{\eta}}$ will satisfy the equation. This can be verified by using an equality $1 - e^{-x} \leq x$ for $x \in [0,1]$; that is, $1 - e^{-\frac{\eta s}{c}} \leq \frac{\eta s}{c} = 1/n^{1/\eta}$.

Efficiency.

- The encoding algorithm uses $n\eta$ homomorphic addition operations, and $n\eta$ hash functions.
- The dimension c of the encoding is $\eta s \cdot n^{\frac{1}{\eta}}$. Usually, η is set to between 2 and 32.
- The decoding algorithm uses n operations of $\mathsf{BF.Check}$.

In summary, we have reduced the encoding size c to be sub-linear in n as desired. However, we still need to reduce the number $\mathsf{BF.Check}$ operations in Decode to be sub-linear in n. We show how to achieve that in our next construction.

3.4.2 BF-COIE

We now show how to improve the above construction to achieve decoding in time $o(n)$. The main idea of this improvement is to use Bloom filters to represent a binary search tree, one BF per level of the tree. We can then guide the decoding algorithm to avoid decoding branches that do not contain non-zero entries. As most branches can be truncated well before reaching the leaf-level Bloom filter, this results in sublinear total cost.

Example. Before presenting the formal protocol for this construction we convey our idea through an example. Let $n = 32$, and suppose we wish to encode the indices $I = \{1, 15, 16\}$. Denote

$$I^k = \left\{ \left\lceil \frac{i}{2^k} \right\rceil : i \in I \right\}.$$

Intuitively, an element i in I^k can be thought of a range of length 2^k covering $[(i-1) \cdot 2^k + 1, i \cdot 2^k]$. We have:

- $I^4 = \{1\}$.
- $I^3 = \{1, 2\}$.
- $I^2 = \{1, 4\}$.
- $I^1 = \{1, 8\}$.
- $I^0 = \{1, 15, 16\}$.

Now, assume we insert each set I^k into its own BF. We can traverse these BF's to decode the set I as follows:

1. Check I^4 for all possible indices. The only possible indices at this level are 1 and 2, since $n = 32$ and I^4 divides the original indices by $2^4 = 16$.

   In the above example, When we query the BF for I^4, it only contains the index 1, which means that no values greater than 16 are contained in I. We can thus avoid checking any such indices at the lower levels.

Now consider the BF at the next level (i.e., the BF for I^3). The only possible values at this level are 1,2,3,4, but since we already know that there are no values greater than 16 in I, we only need to check for values $1, 2$ (since $3 \cdot 8 > 16$).

2. Check I^3 for indices $1, 2$. The BF will show that indices 1 and 2 are both present, which means that we need to check indices $1, 2$ and $3, 4$ in I^2.

3. Check I^2 for indices $1, 2, 3, 4$. The BF will show that indices 1 and 4 are present, which means that we only need to check indices $1, 2$ and $7, 8$ in I^1, all other indices can be skipped.

4. Check I^1 for indices $1, 2, 7, 8$. The BF will show that indices 1 and 8 are present, which means that we need to check indices $1, 2$ and $15, 16$.

5. Check I^0 for indices $1, 2, 15, 16$, and output the final present indices $1, 15, 16$.

Assuming, for now, that there are no false positives, observe that this approach checks at most $2 \cdot |I|$ values at each level, and there are $\lg n$ levels. Therefore, the decoding algorithm will check $O(|I| \cdot \lg n)$ indices, which is sub-linear in n.

BF-COIE. We now describe our BF-COIE construction. As before, we will work out the parameters after describing our construction. The encoding algorithm is described in Algorithm 1.

Algorithm 1 BF-COIE.Encode($[\![v_1]\!], \ldots, [\![v_n]\!]$)

```
For simplicity, $n$ and $s$ are assumed to be powers of 2.
1. $t := \lg \frac{n}{2s}$
2. For $k = 0, \ldots, t$:
    (a) Initialize $[\![B^k]\!] = ([\![B^k_1]\!], \ldots, [\![B^k_\ell]\!]) := (\mathsf{nil}, \ldots, \mathsf{nil})$.
    (b) Choose $\mathcal{H}^k = \{h^k_q : \{0,1\}^* \to [\ell]\}_{q=1}^{\eta}$ at random.
    (c) For $i \in [n]$ and for $q \in [\eta]$:
            $i' := \lceil i/2^k \rceil$, $j := h^k_q(i')$,
            If $[\![B^k_j]\!]$ is nil, then $[\![B^k_j]\!] := [\![v_{i'}]\!]$
            Otherwise, $[\![B^k_j]\!] := [\![B^k_j]\!] + [\![v_{i'}]\!]$
3. Output $[\![B^0]\!], \ldots, [\![B^t]\!]$.
```

Note that in steps (a) to (c) above, the warm-up construction is used to construct BF B^k for indices I^k.

In order to reduce the size of the output encoding, we set t to be $\lg \frac{n}{2s}$ instead of $\lg n$ as described previously. Note that when t is set in this way, I^t contains at most $n/2^t = 2s$ possible values thus maintaining our invariant.

Algorithm 2 BF-COIE.Decode($B^0, \ldots, B^t$)

```
1. Initialize $I, I^0, \ldots, I^{t-1} := \emptyset$

2. Initialize $I^t := \{1, \ldots, n/2^t\} = [2s]$

3. For $k = t, t-1, \ldots, 1$, and for $i' \in I^k$:
   If BF.Check($B^k, i'$) is "yes", add $2i'-1$, $2i'$ in $I^{k-1}$

4. For $i \in I^0$:
   If BF.Check($B^0, i$) is "yes", add $i$ to $I$

5. Output $I$
```

The decoding algorithm is described in Algorithm 2.

Useful lemma. The following lemma will be useful to analyze the parameters c and f_p.

Lemma 3.4. Consider a Bloom filter with false positive rate $\frac{1}{m}$, where m is an arbitrary positive integer. Suppose at most m BF.Check operations are performed in the BF. Then, for any $\delta > 0$, we have:

$$\Pr[\#\text{ false positives} \geq 1+\delta] \leq \frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}.$$

The proof, by an application of the Chernoff bound, can be found in Appendix 7.1.1.

Regarding the above Lemma, we remark that setting $\delta = \Omega(\log \kappa)$, we have

$$\Pr\left[\sum_{i=1}^{m} X_i \geq 1+\delta\right] = \mathsf{negl}(\kappa).$$

Parameters c and f_p. We set the false positive upperbound $f_p := \Omega(\log \kappa)$ for the BF-COIE scheme. In our experiments, we set $f_p = 16$.

Now, let $m = \max(2s, s+2f_p)$, we set the BF false positive rate to $1/m$. Recall that in the BF-COIE construction, the topmost BF B^t performs the BF.Check operation with $2s$ times; see line (2) in Algorithm 2. Using the above Lemma, the number of false positives in the top level BF B^t is at most f_p with all but negligible probability in κ. Furthermore, the index i in B^t is expanded into two indices $2i-1$ and $2i$ in B^{t-1}. This means that the number of false indices to be checked in B^{t-1} due to the false positives in B^t is at most $2f_p$.

Now consider an index i that belongs to B^t. Algorithm 2 will run BF.Check on the values $2i-1$ and $2i$ in B^{t-1}. Since at least one of these values must actually belong to B^{t-1}, this leads to at most one false index being checked. Thus, the maximum number of false indices that would be checked in B^{t-1} is at most $s+2f_p$ (i.e., $2f_p$ from false positives of B^t and s from true positives of B^t).

The above argument applies inductively all the way to the bottom most level, which means that the maximum number of false indices that would be checked in each level BF B^i will be at most $s + 2f_p$. In the end, the bottom BF will have at most f_p false positives, and the overall BF-COIE scheme will have at most f_p false positives with all but negligible probability in κ.

For the compactness parameter c, we must determine the dimension ℓ of each BF. Recall that we set the BF false positive rate to $1/m$ for $m = \max(2s, s+2f_p)$:

$$\left(1 - e^{-\frac{\eta s}{\ell}}\right)^{\eta} \leq \frac{1}{m}.$$

Setting $\ell = \eta \cdot s \cdot m^{\frac{1}{\eta}}$ would satisfy the above condition, which can be verified using an inequality $1 - e^{-x} \leq x$ for $x \in [0, 1]$; that is, $1 - e^{-\frac{\eta s}{\ell}} \leq \frac{\eta s}{\ell} = (1/m)^{1/\eta}$.

Since the encoding has $t + 1$ BFs, the overall compactness parameter is as follows:

$$c = (t + 1) \cdot \ell = O\left(\eta \cdot s^{1+\frac{1}{\eta}} \cdot \lg \frac{n}{s}\right).$$

Efficiency.

- The size c of encoding is $O\left(\eta \cdot s^{1+\frac{1}{\eta}} \cdot \lg \frac{n}{s}\right)$. In our experiment, we choose $\eta = 2$.
- The encoding algorithm uses $O(\eta \cdot n \cdot \lg \frac{n}{s})$ homomorphic addition operations and hash functions.
- The decoding algorithm uses $\mathsf{BF.Check}$ operations for $O(s \lg \frac{n}{s})$ times.

In summary, assuming $s = o(n)$, we reduced the encoding size c to be sub-linear in n. Moreover, we also reduced the number $\mathsf{BF.Check}$ operations to be sub-linear in n.

Remark. Although this scheme has multiple BFs, the size of encoding c is smaller than that of the warm-up scheme! This is because with multiple levels of BFs, we can relax the false positive ratio for each BF. The encoding computation time was increased by a multiplicative factor of $\lg \frac{n}{s}$.

3.4.3 COIE Scheme Based on Power Sums

Removing false positives using power sums. We offer another encoding scheme using quite different techniques that can eliminate the false positives of the prior construction. To achieve this, we abandon Bloom filters, and instead use a power sum encoding, as has been done in several works using DC-Nets for anonymous broadcast FHE.

PS-COIE. We describe a COIE scheme based on power sums, which we call PS-COIE. As before, we will work out the parameters after describing our construction. The encoding algorithm is shown below.

Note that the values of i^j (modulo the underlying plaintext modulus) are publicly computable, so computing $i^j \cdot [\![v_i]\!]$ only requires scalar multiplication and no homomorphic multiplication.

Algorithm 3 PS-COIE.Encode($[\![v_1]\!], \ldots, [\![v_n]\!]$)

```
1. For $j = 1, \ldots, s$:
   Compute $[\![w_j]\!] = \sum_{i=1}^{n} i^j \cdot [\![v_i]\!]$
2. Output $[\![w_1]\!], \ldots, [\![w_s]\!]$.
```

Recall that $v_i \in \{0, 1\}$. If we let $I = \{i : v_i = 1\}$ denote the indices of the nonzero elements, then note that

$$w_j = \sum_{i=1}^{n} i^j \cdot v_i = \sum_{i \in I} i^j.$$

Therefore, this w_j is the jth power sum of the indices. Using the power sums, we present the decoding algorithm in Algorithm 4.

Algorithm 4 PS-COIE.Decode($[\![w_1]\!], \ldots, [\![w_s]\!]$)

```
1. Recall that we have $w_j = \sum_{x \in I} x^j$, for $j = 1, \ldots, s$, and we would like to
   reconstruct all $x$'s in $I$.
2. Let $f(x) = a_s x^s + a_{s-1} x^{s-1} + \cdots + a_1 x + a_0$ denote the polynomial whose
   roots are the indices in $I$.
3. Use Newton's identities to compute the coefficients of this polynomial $f(x)$:
       $a_s = 1$
       $a_{s-1} = w_1$
       $a_{s-2} = (a_{s-1} w_1 - w_2)/2$
       $a_{s-3} = (a_{s-2} w_1 - a_{s-1} w_2 + w_3)/3$
       $\vdots$
       $a_0 = (a_1 w_1 - a_2 w_2 + \cdots w_s)/s$
4. Extract and output the roots of the polynomial $f(x)$.
```

Parameters c and f_p. This COIE scheme has no false positives; that is, $f_p = 0$. The compactness parameter c is equal to s.

Efficiency.

- The encoding algorithm uses $s \cdot n$ homomorphic addition operations and scalar multiplications[2].
- The encoding consists of s ciphertexts.

[2]We do not count the public multiplications to produce powers of i

- The decoding algorithm computes coefficients in time $O(s^2)$. Roots of degree-s polynomial can be found in time $O(s^3 \log p)$, where p is the plaintext modulus of the underlying FHE, by using the Cantor–Zassenhaus algorithm [36].

3.5 CODE Scheme

In the previous section, we showed two constructions of COIE schemes for encoding a vector of indices using sublinear storage. We now turn to the construction of CODE schemes, which, instead of encoding the indices of non-zero entries, encode the actual data values.

Simplified key-value store. To construct our CODE scheme, we first construct an auxiliary data structure that supports the following operations:

- $\mathsf{Init}()$. Initialize the data structure.
- $\mathsf{Insert}(key, value)$. This operation allows the user to insert an item based on its key and value.
- $\mathsf{Values}()$. Returns all values that have been inserted thus far.

This data structure is simpler than a typical key-value store since it doesn't need to find an individual item by key. Note, however, that this is still sufficient to serve our purpose of constructing a CODE scheme.

3.5.1 BF Set

We now show how to instantiate a simplified key-value store using a data structure we call a Bloom filter set (BFS) that is in turn based on the algebraic Bloom filter presented in Section 3.2.2. To insert a pair $(key, value)$, the Bloom filter set stores the actual *value* rather than an indicator bit. Items are inserted similar to before, by adding their value to the locations indicated by the hashes of the *key*.

Input data format. For our construction we make an assumption on the format of the inserted data. Specifically, we assume that all inserted values contain a unique checksum (e.g., a cryptographic hash of the value). We assume that this checksum is sufficiently long that a random sum of checksums does not give a valid checksum except with negligible probability (as a function of κ).

Construction. We first describe the construction of the data structure. We show below how to choose parameters in such a way that the client can extract all the matched items from this Bloom filter, with overwhelming probability.

- $\mathsf{BFS.Init}() \rightarrow (B, \mathcal{H})$. Create an ℓ-dimensional vector B where each element can store any possible value in the domain D. Choose a set of η different hash functions $\mathcal{H} = \{h_q : \{0,1\}^* \rightarrow [\ell]\}_{q=1}^{\eta}$. Initialize $B_i := 0$ for $i \in [\ell]$.
- $\mathsf{BFS.Insert}(B, \mathcal{H}, key, \alpha)$. To add (key, α), we add α to the values stored at the locations indicated by the hashes of *key*. Specifically,

– For $q \in [\eta]$:

  Compute $j = h_q(key)$ and set $B_j := B_j + \alpha$.

- $\mathsf{BFS.Values}(B)$. Initialize a set V to be the empty set. For $j \in [\ell]$, if B_j has a valid checksum, add B_j to V. Finally, output V.

We note that, as previously proposed by Goodrich [91], it is possible to avoid the checksum by maintaining a counter of the number of values inserted for each location. Then, BFS.Values only returns values at locations with a counter of 1.

Parameters. We show how to set the Bloom filter parameters to guarantee that all values can be recovered with all but negligible probability. We assume that we know the upper bound s on the number of inserted values. We prove the following lemma.

Lemma 3.5. If at most s values have been inserted in the BFS data structure, then by setting η and ℓ such that

$$\ell \geq 2(s\eta - 1),$$

we can recover all s values with probability at least $1 - s \cdot (1/2)^\eta$.

Proof. Consider a (key, value) pair (k_i, α_i). We say that this pair has a *total collision* if every hash position for the pair is also occupied by another inserted key, value pair. In this case, α_i cannot be recovered. On the other hand, if at least one hash position has no collisions, then we can recover the value. Note that the collision depends on the key k_i but not the value α_i.

For a given key k_i, we define the event $\mathsf{TCOL}(k_i)$:

$$\mathsf{TCOL}(k_i) = 1 \text{ if } \forall q \in [\eta], \exists (k', q') \neq (k_i, q) : h_q(k_i) = h_{q'}(k').$$

Here, k' can be the key of any item that has been inserted in the set. Since the set contains at most s items, there are at most s possible keys for k'. Recall also that η hash functions are applied for each item.

Since for each k_i, there are at most $\eta s - 1$ pairs of (k', q')s that are different from (k_i, q), we can bound the collision probability as follows:

$$\Pr[\mathsf{TCOL}(k_i)] \leq \left(\frac{(\eta s - 1)}{\ell}\right)^\eta$$

Thus, if we choose η and ℓ such that $\ell \geq 2(s\eta - 1)$, we have

$$\Pr[\mathsf{TCOL}(k_i)] \leq (1/2)^\eta$$

Taking a union bound over all s inserted values, we have

$$\Pr[\exists k_i : \mathsf{TCOL}(k_i)] \leq s \cdot (1/2)^\eta$$

. ■

3.5.2 CODE Scheme Based on BF Set

In this section, we construct a CODE scheme. Recall that unlike encoding the indices through a COIE scheme, a CODE scheme encodes data in a compressed manner. The main idea of our construction is simulating the operations of BFS; we call our scheme BFS-CODE.

Pre-processing the input data. As mentioned in the description of the BF Set construction, we need to pre-process the input data so that each item is attached with its checksum. Although a data item v is represented as a single number, it is assumed that v can be parsed as $v.val$ for its actual value and $v.tag$ for its checksum. Moreover, we assume that the checksum is long enough, such that a random linear combination of checksums is only negligibly likely to produce a valid checksum (i.e., $|checksum| = \omega(\lambda)$).

We stress that when our CODE scheme is used for secure search, this pre-processing can be performed locally by the client prior to encrypting his data. Moreover, computing checksum adds only a tiny amount of overhead.

BFS-CODE. We now describe our (n, s, c, f_p)-BFS-CODE construction over domain D. As before, we will work out the parameters after describing our construction. The encoding algorithm is shown below.

Algorithm 5 BFS-CODE.Encode($[\![v_1]\!], \dots, [\![v_n]\!]$)

```
1. $\eta = \kappa + \lg s$; $\ell = 2(\eta s - 1)$
2. Initialize $[\![B]\!] = ([\![B_1]\!], \dots, [\![B_\ell]\!]) := ([\![0]\!], \dots, [\![0]\!])$.
3. Choose $\mathcal{H} = \{h_q : \{0,1\}^* \to [\ell]\}_{q=1}^{\eta}$ at random.
4. For $i \in [n]$ and for $q \in [\eta]$:
        $j := h_q(i)$; $[\![B_j]\!] = [\![B_j]\!] + [\![v_i]\!]$
5. Output $[\![B]\!]$.
```

Note that at step 4 in the above, if v_i is 0, then B_j stays the same. On the other hand, if v_i is not 0, B_j will be increased by v_i. This implies that B will exactly hold the result of operations $\{\mathsf{BFS.Insert}(B, \mathcal{H}, i, v_i) : i \in \mathsf{idx}(v)\}$.

The decoding algorithm is simple, and it's described in Algorithm 6.

Algorithm 6 BFS-CODE.Decode(B)

```
1. Output BFS.Values($B$)
```

Correctness. This is immediate from the additive homomorphism of the underlying encryption scheme and the parameters for the BFS. In particular, we set $\eta = \kappa + \lg s$ so that the probability of recovery error is at most 2^{κ}.

Parameters c and f_p. The checksums attached to the data items ensure that we have no false positives with overwhelming probability, that is, $f_p = 0$. The compactness parameter c is the dimension ℓ of the BF, which is $O(\eta s)$.

Efficiency.

- The encoding algorithm uses $\ell = O(\eta s)$ encryption operations, $\eta \cdot n$. addition operations, and ηn hash functions.
- The encoding consists of ℓ ciphertexts.
- The decoding algorithm uses ℓ decryption operations.

Since by Lemma 3.5, the size ℓ of the Bloom filter only depends on the number of matches s and the number of hash function η, we get that the communication complexity of the above protocol is independent of the database size n.

3.6 Secure Search Protocols

We implement secure search protocols by using compressed oblivious encoding schemes. We begin by defining a relaxed notion of correctness that allows for false positives, as is needed in some of our constructions. we then define security of secure search.

3.6.1 (ℓ, f_p)-Relaxed Secure Search

We relax the correctness guarantee to allow the Client to retrieve a superset of the matching records. Specifically, if $\mathcal{S}$ is the set of indexes matching a Client's query q, then at the end of the protocol, we require the Client to obtain a set $\mathcal{S}'$ such that:

- With all but negligible probability, $\mathcal{S} \subseteq \mathcal{S}'$
- With all but negligible probability, $|\mathcal{S}' \setminus \mathcal{S}| \leq f_p$.

We parameterize a secure search scheme by (ℓ, f_p), where ℓ is the *amortized* communication complexity *per* matching record, and f_p is the number of "false positives," as defined above.

3.6.2 Security of Setup-free Secure Search

To define security of our secure search schemes, we use a game-based security definition similar to that of Akavia et al. [3]. The game is between a challenger and an adversary $\mathcal{A}$ with regard to a setup-free search scheme, $\mathsf{sec\text{-}search}$, and an FHE scheme, FHE.

$\mathsf{Game}^{\mathsf{sec\text{-}search}}_{\mathsf{FHE}}(\mathcal{A})$:

1. The challenger runs a key generation algorithm (with computational security parameter κ) and sends the evaluation key to $\mathcal{A}$ so that $\mathcal{A}$ can perform homomorphic additions and multiplications.

2. $\mathcal{A}$ chooses either:
    - Two databases $x^0 = (x_1^0, \ldots, x_n^0)$ and $x^1 = (x_1^1, \ldots, x_n^1)$ of the same length, and a query q, or
    - A single database $x = (x_1, \ldots, x_n)$ and two queries q^0, q^1 of the same circuit size.

    In both cases, we require that the sizes of the two result sets (denoted by s) are equal.
3. The challenger samples $b \leftarrow \{0, 1\}$. Then, either
    - Runs Setup on input x^b and the search protocol from $\mathsf{sec\text{-}search}$ on input q, or
    - Runs Setup on input x, and the search protocol from $\mathsf{sec\text{-}search}$ on input q^b.
4. $\mathcal{A}$ outputs a bit b'
5. We say that $\mathcal{A}$ has advantage

$$\mathsf{Adv}_{\mathsf{FHE}}^{\mathsf{sec\text{-}search}}(\mathcal{A}) = |\Pr[b = b'] - 1/2|.$$

Definition 3.6. A setup-free (ℓ, f_p)-secure search scheme $\mathsf{sec\text{-}search}$ is *fully secure* if every PPT adversary $\mathcal{A}$ controlling the server has a negligible advantage $\mathsf{Adv}_{\mathsf{FHE}}^{\mathsf{sec\text{-}search}}(\mathcal{A}) \leq \mathsf{negl}(\kappa)$ in the game above.

3.6.3 From COIE to Secure Search

We next present our framework for obtaining Secure Search from COIE. The intuition is likely already clear from the previous descriptions: the encrypted client query is applied to the dataset, returning an encrypted bit vector indicating where index matches lie. The server homomorphically computes the hamming weight of this vector, and sends it to the client for decryption. This provides the result set size to the Server, allowing it to encode the result vector in the COIE.[3] The encoding is sent to the client for decryption and decoding.

Because the COIE only encodes the indices, and not the data values, we then add a PIR step to fetch the corresponding data. Note that if the COIE scheme admits false positives, it is possible that the number of false positives, and therefore the number of PIR queries, depends on the data, leaking something to the Server. To fix this problem, the client pads the number of PIR queries as follows. It fixes a bound f_p on the number of false positives, and aborts if the actual number of false positives exceeds this bound. Otherwise, the client uses enough dummy queries to pad the number of PIR queries to $s + f_p$.

[3]We note if we don't wish to reveal this to the server, we can use a fixed, global upper bound, or, if it is appropriate to the application, the client can add noise to provide differential privacy. It is also worth pointing out that prior work leaks the result set size as well.

Algorithm 7 Secure search with a (n, s, c, f_p)-COIE scheme.

1. Client runs the FHE key generation algorithm and encrypts database $x = (x_1, \ldots, x_n)$ with $x_i \in \{0,1\}^m$. It then sends $[\![x]\!] = ([\![x_1]\!], \ldots, [\![x_n]\!])$ and the evaluation key to Server.
2. Client sends an encrypted query $[\![q]\!]$.
3. Server homomorphically evaluates the encrypted query $[\![q]\!]$ on each encrypted record. In particular, let $[\![b]\!] = ([\![b_1]\!], \ldots, [\![b_n]\!])$ where $[\![b_i]\!] = [\![q(x_i)]\!]$. Note that $q(x_i) = 1$ if record i is a match and is equal to 0 otherwise.
4. Server homomorphically computes $[\![s]\!] = \sum_{i=1}^{n} [\![b_i]\!]$, and sends to Client for decryption.
5. Client decrypts $[\![s]\!]$ to obtain s, and sends s to Server.
6. Server calls $\mathsf{COIE.Encode}([\![b]\!])$ with sparsity parameter s, to obtain an encrypted encoding $[\![C]\!]$. It sends $[\![C]\!]$ to Client.
7. Client decrypts $[\![C]\!]$ into C and calls $\mathsf{COIE.Decode}(C)$ to obtain a set $\mathcal{S}'$ of size $s + e$ indexes. If $e > f_p$, Client aborts. Otherwise, Client adds $f_p - e$ number of dummy indexes to $\mathcal{S}'$.
8. Client runs a PIR protocol with the Server to obtain the records corresponding to the indexes in $\mathcal{S}'$.

Theorem 3.7. Given an FHE scheme, a (n, s, c, f_p)-COIE scheme in the random oracle model, and a PIR scheme in the random oracle model with communication complexity ℓ_p for records in $\{0,1\}^m$, the construction in Algorithm 7 yields a (ℓ, f_p)-secure search scheme for records in $\{0,1\}^m$ in the Random Oracle Model, where $\ell = \frac{c \cdot \ell_c + (s+f_p) \cdot \ell_p}{s}$, ℓ_c is the length of an FHE ciphertext, and s is the number of matching records.

Proof. We begin by proving that the adversary cannot distinguish between two different queries. The adversary chooses a database x and two queries q^0 and q^1, with the promise that $s = \sum_{i=1}^{n} q^0(x_i) = \sum_{i=1}^{n} q^1(x_i)$.

The entire view of the adversary during the experiment can be reconstructed efficiently given (1) the encrypted database $[\![x]\!]$ (2) the encrypted query $[\![q]\!]$, (3) $s + f_p$ iterations of the PIR protocol, requesting indexes in $\mathcal{S}'_b$, where s is the number of matching records.

Since the value of s is the same for q^0 and q^1, the two things that change in the view of the adversary when switching from $b = 0$ to $b = 1$ are (1) the encrypted query $[\![q^b]\!]$ (2) the set of indexes $\mathcal{S}'_b$ (but not the number) requested during the PIR step.

We also note that the experiment only aborts when the number of received false positives e is greater than the bound f_p, which only happened with probability $\mathsf{negl}(\kappa)$ for a statistical security parameter κ. Thus, we ignore this possibility in the following.

We can now proceed via a standard hybrid argument:

- We first consider the real experiment with $b = 0$.

- We then switch the encrypted query from q^0 to q^1, but leave the set of indexes in the PIR step as $\mathcal{S}'_0$. Indistinguishability of the adversary's view follows from the IND-CPA security of the FHE scheme.
- Next, we switch the set of indexes in the PIR step from $\mathcal{S}'_0$ to $\mathcal{S}'_1$. Indistinguishability of the adversary's view now follows from the security of the PIR scheme. This is now identical to the real experiment with $b = 1$.

We conclude that the probability the adversary outputs 0 or 1 differs by a negligible amount when $b = 0$ versus $b = 1$. Therefore, the advantage of the adversary in guessing b is negligible.

The proof that the adversary cannot distinguish between the same query applied to two different databases follows nearly identically. ■

3.6.4 From CODE to Secure Search

We next present our framework for obtaining Secure Search from CODE.

Algorithm 8 Secure search with a (n, s, c, f_p)-CODE scheme.

```
1. Client runs the FHE key generation algorithm and encrypts database $x = (x_1, \ldots, x_n)$ with $x_i \in D$. It then sends $[\![x]\!] = ([\![x_1]\!], \ldots, [\![x_n]\!])$ and the evaluation key to Server.
2. Client sends an encrypted query $[\![q]\!]$.
3. Server homomorphically evaluates the encrypted query $[\![q]\!]$ on each encrypted record. In particular, let $[\![b]\!] = ([\![b_i]\!], \ldots, [\![b_n]\!])$ where $[\![b_i]\!] = [\![q(x_i)]\!]$. Note that $q(x_i) = 1$ if record $i$ is a match and is equal to 0 otherwise.
4. Server homomorphically computes $[\![s]\!] = \sum_{i=1}^{n} [\![b_i]\!]$ and sends $[\![s]\!]$ to Client.
5. Client decrypts $[\![s]\!]$ to obtain $s$ and sends it back to the Server.
6. Server computes $[\![d_i]\!] = [\![b_i]\!] \cdot [\![x_i]\!]$ for $i \in [n]$. Then, it applies $\mathsf{CODE.Encode}([\![d_1]\!], \ldots [\![d_n]\!])$ with sparsity parameter $s$, to obtain an encrypted encoding $[\![C]\!]$. It sends $[\![C]\!]$ to Client.
7. Client decrypts $[\![C]\!]$ to $C$ and decodes $C$ to obtain a set $\mathcal{S}$ of size $s$ matching records.
```

Theorem 3.8. Given an FHE scheme, and a (n, s, c, f_p)-CODE scheme over domain D in the random oracle model, the construction in Algorithm 8 yields a (ℓ, f_p)-secure search scheme for records in domain D in the random oracle model, where $\ell = \frac{c(s) \cdot \ell_c}{s}$, ℓ_c is the length of an FHE ciphertext with plaintext space D, and s is the number of matching records.

The proof is similar to the COIE-based scheme and can be found in Appendix 7.1.2.

On the use of homomorphic multiplication. As described, our CODE-based search scheme uses n homomorphic multiplications to create the vector

$[\![d]\!]$. However, it may be the case that this vector is already produced as part of the match step, for example for arithmetic queries. In this case, our CODE scheme requires no further homomorphic multiplications.

On volume attacks. In our secure search schemes, the client sends the number s of matching records to the server so that the server can create an oblivious compress encoding. One recent line of works has developed attacks using volume leakage (e.g., [22, 98, 121]), and these types of attacks can be applied to our scheme in theory.

In our scheme, the volume attacks can be mitigated by hiding s in a differentially private manner. In particular, the client can add a small amount of noise to s before sending it to the server. A similar approach was used in previous work e.g., [148].

3.7 Evaluation

3.7.1 Fetch time

We implemented our search protocols based on BF-COIE, PS-COIE, and BFS-CODE schemes. All protocols were implemented using PySEAL [174], which is a Python wrapper of the Microsoft research SEAL library (version 3.6) [160] using the BFV encryption scheme [74]. We instantiated a single-server PIR protocol in our construction using SealPIR [6]. For the root finding step of the decoding procedure in PS-COIE, we use an implementation based on SageMath 9.2 [171].

Measuring the Fetch step. Our search framework improves the overall search time by executing the Match step only once, while the LEAF protocol must execute the Match step s times. However, since we do not optimize the Match step itself over prior work, we focus on measuring the cost of the Fetch procedure. That is, our experiments measure the time from when the server holds encrypted query results, i.e., $([\![b_1]\!], \dots, [\![b_n]\!])$ with $b_i \in \{0, 1\}$, to when the client recovers all s records matching the query. Specifically, we measure the cost of steps 4 and up in Algorithms 7 and 8. Similarly, for LEAF+, we only measure the cost of the Fetch step.

Database. To measure the performance of our protocols, we run experiments with database size n ranging from 1000 to 100,000 data items and the result set size s set to between 8 and 128. As in the LEAF+ experiments [181], all data items are 16-bit integers.

BF-COIE parameters. For the BF-COIE secure search, we set the parameters as indicated in Section 3.4.2.

- We set the false positive upperbound $f_p = 16$. Recall that the client aborts (without executing the PIR) if the actual number of false positives exceeds this, but this only happens with probability negligible in the security parameter, which we set $\kappa = 40$.
- We set the number of hash function $\eta = 2$ for each Bloom filter, so each BF has size $\ell = 2s \cdot \sqrt{2s}$. (If $2s < s + 2f_p$, we set $\ell = 2s \cdot \sqrt{s + 2f_p}$).

BFS-CODE parameters. For BFS-CODE secure search, with $\kappa = 40$, the number of hash functions η is set to $\kappa + \lg s$, and the Bloom filter size is set to $2(\eta s - 1)$. Additionally, each data item is attached with a 40-bit checksum to guarantee a $2^{-\kappa}$ probability of collision. We used SHA2 to compute a checksum.

Implementing LEAF+. For a comparison we also implemented the fetch step of the LEAF+ protocol [181], since their implementation is not publicly available.

Their protocol has $O(\log \log n)$ depth of multiplications. Therefore, they have to use bootstrapping techniques to reduce the accumulated noise. However, SEAL doesn't provide a method for bootstrapping, and we suspect that they added a customized implementation of bootstrapping on top of SEAL. Unfortunately, their implementation is not available.

We address this issue by choosing to ignore the time for bootstrapping when we measure the running time of our implementation of LEAF+. Of course, our implementation doesn't output the correct results, but the measured running time will be shorter than the actual running time. Therefore, we believe that this measured time serves as a good baseline.

Experiment environments. All our experiments were performed on an Intel®Core 9900k @4.7GHz with 64GB of memory. For fair comparison, the test was performed on a single thread with no batching optimizations for computation. Networking protocol between server and clients is a 1Gbps LAN.

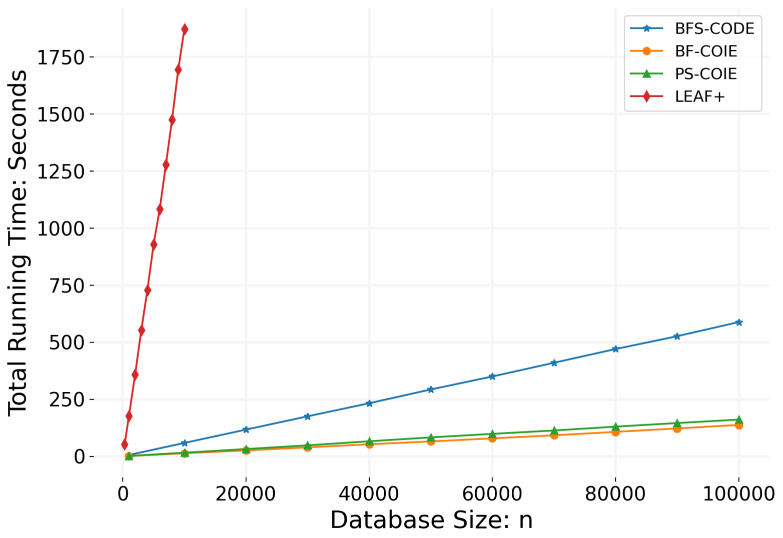
Results: Fetch time vs. database size. Figure 3 shows the performance of our protocols as a function of database size, while the result set size s is fixed to 16. However, for LEAF+, we plot the time for fetching only a **single** record, since fetching s records takes too long. In our implementation of LEAF+, fetching even a single record when $n = 10,000$ requires 1872 seconds. We note that the authors of LEAF+ report about 60 seconds for a single fetch [181]. We conjecture that they parallelize the scheme with 32 threads. Here, we only use a single thread.

All three of our protocols greatly outperform LEAF+. Looking at BF-COIE in particular:

- In BF-COIE search, fetching 16 records with $n = 10,000$ takes 16.7 seconds, compared to 1872 seconds for a single record fetch in LEAF+. We believe that the speed up is due to the fact that LEAF+ (with a single-record fetching) needs $O(n \log n)$ homomorphic additions and $O(n)$ homomorphic multiplications, while BF-COIE search needs only $O(n \log \frac{n}{s})$ homomorphic additions with *no homomorphic multiplications.* In addition, as Figure 4 shows, the overhead of the PIR step to retrieve the actual data is small.

- Due to the sequential limitation in LEAF+, fetching 16 records with LEAF+ is extrapolated to take about $16 \cdot 1872 = 29952$ seconds. Overall, BF-COIE search is about **1800 times faster** than LEAF+.

The time for all three of our protocols is dominated by the server's computation during encode, which grows linearly with the DB size.



For LEAF+, we plot the time for fetching only a **single** record, since fetching s records takes too long.

Figure 3: Fetch time vs. Database size with $s = 16$.

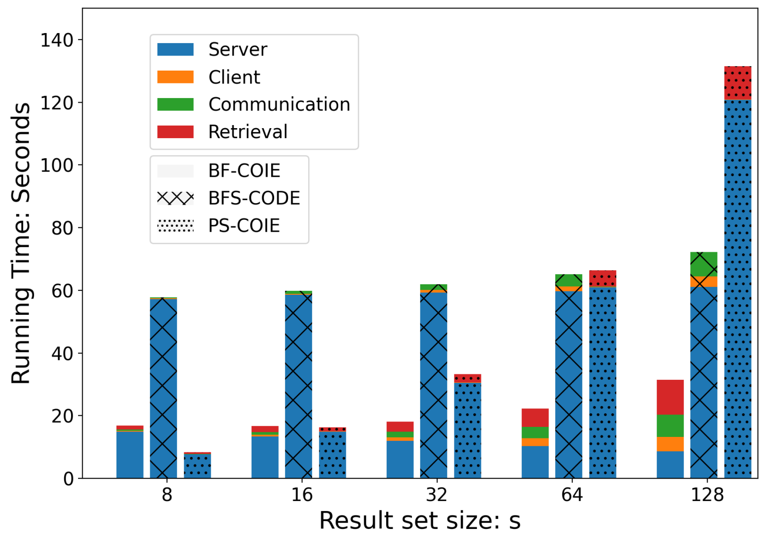


Figure 4: Fetch time vs. Result set size with $n = 10,000$.

Since the number of hash functions η is larger in the BFS-CODE protocol than in BF-COIE protocol, the encoding step of this protocol takes longer.

Results: fetch time vs. the result set size. Figure 4 shows the performance of our protocols as a function of the result set size s while n is fixed to $10,000$. Here, again the performance is dominated by the encoding step, but the relative costs have changed. Due to the need to compute more power sums, the PS-COIE protocol performs worse than BS-COIE and BFS-CODE when s becomes moderately large.

The time used for transmitting the data over network (green in Figure 4) increases for larger s. However, it still remains small for all three schemes. In the scenario of having lower network bandwidth, batching is recommended to pack a vector of ciphertexts into a single ciphertext with relatively low computation overhead. We discuss communication costs further in Section 3.7.3.

3.7.2 Overall Running Time

Although we do not optimize the Match step itself over prior work, we provide an estimated comparison of the running time for the end-to-end flow.

Our search framework improves the overall search time by executing the Match step only once, while the LEAF protocol must execute the Match step s times. Based on this, we can extrapolate the running time as follows:

- The overall running time for LEAF:

$$Time(\mathsf{LEAF}) = s \cdot \mathsf{MT}(\mathsf{LEAF}) + s \cdot \mathsf{FT}(\mathsf{LEAF}).$$

Here, MT and FT denote the match time and fetch time respectively.

- The overall running time for the BF-COIE scheme:

$$Time(\mathsf{BF\text{-}COIE}) = \mathsf{MT}(\mathsf{BF\text{-}COIE}) + \mathsf{FT}(\mathsf{BF\text{-}COIE})$$

Although the implementation (nor the algorithm) of the matching step of LEAF protocol is not available in [181], we expect that it holds $\mathsf{MT}(\mathsf{LEAF}) \approx \mathsf{MT}(\mathsf{BF\text{-}COIE})$. In the experiment performed in LEAF (see Figure 9 in [181]), we have $m = \frac{\mathsf{MT}(\mathsf{LEAF})}{\mathsf{FT}(\mathsf{LEAF})} \approx 1.5$. For $s = 16$, setting $\mathsf{FT}(\mathsf{LEAF}) = 1800 \cdot \mathsf{FT}(\mathsf{BF\text{-}COIE})$ based on the above discussion, we can estimate the speed-up as follows:

$$\frac{Time(\mathsf{LEAF})}{Time(\mathsf{BF\text{-}COIE})} = \frac{s \cdot (m+1)}{m + 1/1800}.$$

Thus, with $s = 16$, we estimate that our BF-COIE scheme has roughly 26X end-to-end speed-up.

3.7.3 Communication

We now look at the communication required by each of our schemes and by LEAF+. Figure 5 shows the network cost of the protocols when the result set size s is 16 and the size of the database is $n = 10,000$. In our implementations, the length of an FHE ciphertext is approximately 103KB and the communication cost of PIR is approximately 369KB.

	LEAF+	BF-COIE	PS-COIE	BFS-CODE
#ct's	704	1323	17	1321
#PIR	0	32	16	0
#ct's (w/ batching)	32	2	2	2

Figure 5: The communication costs ($n = 10,000$ and $s = 16$).

To explain this table, we first need to explain how we determined the costs of LEAF+ and PIR.

- *LEAF+.* Since LEAF+ fetches each data item and the corresponding index one by one, LEAF+ needs to 16 rounds of communication to retrieve 16 data items. Worse yet, LEAF+ requires the client to send the index of the previous match (requiring $\lg n$ bits) in his next query to ensure correctness. Finally, LEAF+ uses bitwise encryption requiring a ciphertext for each bit of the encrypted communication. Thus, in a single round, the client must send $\lg n = 14$ ciphertexts and the server returns $16 + \lg n = 30$ ciphertexts – 16 ciphertexts for returning the matching data item, and $\lg n$ ciphertexts to return its index. This amounts to 704 ciphertexts for fetching 16 items (excluding the query).

- *PIR costs.* We reduce the cost of PIR for the COIE-based schemes by making a slight modification. In addition to storing the FHE-encrypted database, the server also stores a copy of each record encrypted using a symmetric-key encryption scheme (resulting in much shorter ciphertexts). Then, in the PIR step, the client fetches this symmetrically encrypted ciphertext instead of the FHE-encrypted one.

  We use SealPIR for our PIR protocol, which requires 368.6 KB per request. We remark that a very recently introduced SealPIR+ takes 80KB per request (see Table 1 in [4]), using which we can reduce the communication further.

We can now compare the communication costs based on rows 1 and 2 of Figure 5. We see that the communication of BF-COIE and BFS-CODE are roughly twice that of LEAF+, while PS-COIE requires almost 10X less communication. The extra communication needed by BF-COIE and BFS-CODE can likely be offset by the much lower round complexity required by our protocol since the latency costs are likely higher than the cost for the extra bandwidth.

Reducing communication using ciphertext batching. We now describe an optimization to significantly reduce the communication of our protocols at the cost of slightly increased server computation. SEAL allows thousands of encrypted values to be packed together into a single ciphertext. This allows us to pack the ciphertexts in all of our protocols into just one a single ciphertext to be sent from the server to the client. However, this does require the server to do some additional computation to pack the ciphertexts prior to sending them. We experimentally measured this packing, and it requires approximately 3 seconds on a single threaded machine.

LEAF+ can also take advantage of packing to reduce the communication of their protocols. However, since the results must be returned one at a time, the best LEAF+ can do is to pack all ciphertexts that are sent in each round, resulting in a total of 32 ciphertexts.

We note that the cost of PIR is unchanged by this modification. Thus, with the packing optimization, the communication of BFS-CODE is roughly 1/16 of the communication needed by LEAF+, but BF-COIE and PS-COIE require approximately 4X and 2X more communication than LEAF+ respectively when SealPIR is used; however, when SealPIR+ is used, both schemes have slightly less communication than LEAF+.

3.8 Related Work

3.8.1 Techniques for Secure Search

Secure pattern matching (SPM) on FHE-encrypted data. In SPM, given an encrypted query $[\![q]\!]$ and n FHE-encrypted data items $([\![x_1]\!], \ldots, [\![x_n]\!])$, it returns a vector of n ciphertexts $[\![b_1]\!], \ldots, [\![b_n]\!]$, where b_i indicates whether the ith data element is a match [43,44,123,187]. Their works focus on optimizing the search circuits to determine whether a data item matches the query, and therefore the communication complexity and client's running time are proportional to the number of data items. Our work focuses on the orthogonal problem of optimizing the retrieval of the matched data items with sublinear communication and client computation.

Searchable encryption (SE). Searchable encryption [29, 167] allows highly efficient search (usually in $o(n)$ time) over encrypted data. Efficient SE schemes have been proposed for a wide variety of queries including equality queries [41,54], range queries [107, 157], and conjunctive queries [37, 147]. However, to achieve sublinear query performance, SE schemes require significant preprocessing and relax security, allowing some partial information about the queries and data (e.g. access patterns) to leak to the server. For a recent survey on SE constructions and security, see Fuller et al. [82]. In contrast, our work focuses on achieving preprocessing-free secure constructions, leaking nothing about the queries or results other than their sizes.

Property Preserving Encryption (PPE). As a different approach, property-preserving encryption [146] produces ciphertexts that maintain certain relationships (e.g., equality, and order) of the underlying plaintexts. This allows queries to be performed over ciphertexts in the same way that they can be carried out over plaintexts. Examples of PPE include deterministic encryption [19] allowing equality queries, and order-preserving encryption [27, 28] allowing range queries. However, it has been shown [96,97,109] that such property-preserving ciphertexts leak a lot of information about the underlying plaintexts. See [82] for a survey of constructions and attacks.

3.8.2 General Techniques

Private information retrieval (PIR). PIR allows the client to choose the index i and retrieve the ith record from an untrusted server while hiding the index i [47]. However, this protocol by itself provides only a limited search functionality requiring the client to know the index of the data to retrieve. In this work, we aim at protocols supporting any arbitrary search functionality.

Secure multi-party computation (MPC). Secure two-party computation [88, 186] allows players to compute any function of their private inputs without compromising privacy of their inputs. For example, the client and the server can run a protocol for secure two-party computation to solve the secure search problem. While there has been much progress in improving efficiency of MPC protocols, such protocols still require $\Omega(n)$ communication and $\Omega(n)$ client computation per query. In this work, we aim to achieve protocols with sublinear communication and client work.

Oblivious RAM (ORAM) and Oblivious data structure (ODS). ORAM [89] is a protocol which allows a client to store an array of n items on an untrusted server and to access an item obliviously, that is, hiding contents and which item is accessed (i.e., the access pattern). Likewise, ODS [180] allows the client to store and use a data structure obliviously. One could implement secure search by utilizing an ODS for a search tree. However, ODS constructions typically need $\Omega(\log^2 n)$ rounds for each operation. In this work, we aim at achieving a constant round protocol.

3.9 Conclusion

We have presented several new constructions of secure search based on fully homomorphic encryption. Prior constructions were inherently sequential, returning only a single record from the result set, and requiring a new query from the client that depended on the index of the previous match. We have demonstrated several new methods for encoding the entire result set at one time, removing the added rounds, and allowing the server work to be parallelized. Additionally, we have shown that this can be done without homomorphic multiplication, ensuring low computational cost at the server. Finally, we have implemented our constructions, and demonstrated up to three orders of magnitude speed-up over prior work. Additionally, we introduced the notion of compressed oblivious encoding which may be of independent interest.

4 Secure Sampling

4.1 Introduction

Random sampling is an important tool when computing over massive data sets. It has wide application in generating small summaries of data, and serves as a key building block in the design of many algorithms and estimation procedures. In particular, L_p sampling has been used to develop important streaming algorithms such as the heavy hitters, L_p norm estimation, cascaded norm estimation, and finding duplicates in data streams [5, 32, 116, 140].

In this work, we introduce and explore the problem of private two-party sampling. We consider a setting in which two parties would like to sample from a distribution whose probability mass function is distributed across the two parties. Specifically, we assume parties P_1 and P_2 each hold n-dimensional vectors $\mathbf{w}_1 = (w_{1,1}, \ldots, w_{1,n})$ and $\mathbf{w}_2 = (w_{2,1}, \ldots, w_{2,n})$ respectively where every $w_{b,j}$ is non-negative. These vectors each represent a (possibly non-normalized) probability mass function of a distribution. Specifically, for $b \in \{1, 2\}$, $i \in [n]$, the non-negative value $\frac{w_{b,i}}{||\mathbf{w}_b||_1}$ represents the probability mass placed by distribution $\mathcal{D}_b$ on element i. We assume that the dimension n is very large, and our goal is to obtain secure sampling protocols with communication that is *sub-linear* in n.

We consider various ways of deriving the probability mass function $\mathcal{D}$ of the joint distribution from the two individual probability mass functions. Specifically, we consider:

- L_1 distribution: Sample item i with probability $\frac{w_{1,i}+w_{2,i}}{||\mathbf{w}_1+\mathbf{w}_2||_1} = \frac{w_{1,i}+w_{2,i}}{\sum_j (w_{1,j}+w_{2,j})}$.
- L_2 distribution: Sample item i with probability $\frac{(w_{1,i}+w_{2,i})^2}{||\mathbf{w}_1+\mathbf{w}_2||_2^2} = \frac{(w_{1,i}+w_{2,i})^2}{\sum_j (w_{1,j}+w_{2,j})^2}$.

- Product distribution: Sample item i with probability $\frac{w_{1,i} \cdot w_{2,i}}{\langle \mathbf{w}_1, \mathbf{w}_2 \rangle} = \frac{w_{1,i} \cdot w_{2,i}}{\sum_j (w_{1,j} \cdot w_{2,j})}$[4].

Realizing these sampling functionalities securely is immediate via generic 2PC techniques, but the resulting protocols will require communication that is linear in the input length. With *sublinear* communication, however, it is unclear how to perform some of these tasks (or whether it is even possible to do so), even with an *insecure* protocol. We give a (partial) characterization of when such sublinear sampling is possible, and give secure protocols for realizing these functionalities where possible.

Product sampling and the exponential mechanism. While L_1 and L_2 sampling are well-studied, to the best of our knowledge, we are the first to consider the notion of product sampling. We describe a concrete, independent application for this new notion: product sampling can be used to implement a distributed version of the well-known exponential mechanism for differentially-private data release [135].

4.1.1 Our Work

We explore the problems described above, providing multiple two-party protocols, all with sub-linear communication, in the semi-honest security model. We note that our protocol for product sampling has additional leakage, beyond what is revealed by the sampling functionality. We characterize exactly what this leakage is, and provide evidence that similar leakage is necessary to achieve sublinear communication. Specifically, we show the following.

L_1 sampling. We begin by constructing a two-party protocol for L_1 sampling that relies on fully homomorphic encryption (FHE). The main idea behind the protocol is to obliviously sample from each of the two parties inputs independently, and then to securely choose one of the two samples using an appropriately biased coin toss. The results are described in Section 4.2.

L_2 sampling. We also provide a protocol for secure L_2 sampling that relies on fully homomorphic encryption (see Section 4.3). In this case, however, achieving L_2 sampling is non-trivial. In fact, even relying on FHE, it is not immediately clear how to compute $\|\mathbf{w}_1 + \mathbf{w}_2\|_2^2$ with sublinear communication.

Surprisingly, our L_2 sampling protocol runs in constant rounds and with $\widetilde{O}(1)$ communication[5]. Interestingly, it does not require us to compute $\|\mathbf{w}_1 + \mathbf{w}_2\|_2^2$. To achieve this, we developed a novel technique called "corrective sampling", which we overview in the next subsection. We note that our techniques straightforwardly extend to L_p sampling, for constant p.

Product sampling. We then turn to product sampling. We assume, without loss of generality, that the vectors $\mathbf{w}_b$ are normalized (see Section 4.4 for justification).

[4]Of course, if $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle = 0$, the probability space is not well-defined, and in this case, we require the protocol to simply output $\perp$.

[5]Throughout the paper, we will describe the round and communication complexities using the asymptotic notation only based on n. That is, all other parameters (e.g., security parameter) independent on n will be suppressed in the asymptotic expressions.

We first begin with a communication lowerbound, demonstrating that product sampling with sublinear communication is impossible, even without privacy guarantees, if the two input distributions are insufficiently correlated (i.e., $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle = o(\frac{1}{n^2})$). We show this through a reduction from the Set Disjointness problem.

Knowing this lowerbound, we consider the problem under a promise that the input vectors are sufficiently correlated. Assuming that $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle = \omega(\frac{\log n}{n})$, we provide a two-party protocol for secure product sampling leaking (at most) the inner product of the two parties' inputs. We note that the promise itself leaks some information, so some leakage here is inevitable. Interestingly, we observe that the protocol can be modified to provide a trade-off between the communication cost and the leakage. We also discuss why this trade-off is inherent.

Constant round product sampling. Our product sampling protocol has a round complexity that depends on the inner product. In Section 4.5, we show how to make our construction constant round while incurring small additional leakage. Importantly, we must do this *without computing the exact inner product* which itself requires $O(n)$ communication [11].

Two party exponential mechanism. As mentioned previously, one important application of product sampling is the exponential mechanism for providing differential privacy [135]. In Section 4.6, we describe this application in detail.

For this particular application we face an additional challenge: the leakage of $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$ that we relied on for achieving sub-linear communication in product sampling does not preserve differential privacy. To overcome this issue, we construct a new, differentially-private approximation for inner product, and show how to use this for building a sub-linear communication secure computation of the exponential mechanism.

4.1.2 Technical Overview

In the following, we overload notation and let $\mathcal{D}$ denote a distribution as well as its probability mass function. As discussed previously, we consider the case where a probability mass function is distributed across two parties, and the parties would like to securely sample from the corresponding distribution. We consider several ways in which the probability mass function can be distributed across the two parties.

L_1 sampling of convex combinations. In this case, party 1 (resp. party 2) holds a vector $\mathbf{w}_1$ (resp. $\mathbf{w}_2$), indexed from 1 to n. For $i \in [n]$, $w_{1,i}/||\mathbf{w}_1||_1$ (resp. $w_{2,i}/||\mathbf{w}_2||_1$) corresponds to the probability mass of i under distribution $\mathcal{D}_1$ (resp. $\mathcal{D}_2$). The goal of the parties is to sample from the distribution $\mathcal{D}$, defined as follows for $i \in [n]$:

$$\begin{aligned}\mathcal{D}[i] &:= \frac{||\mathbf{w}_1||_1}{||\mathbf{w}_1||_1 + ||\mathbf{w}_2||_1} \cdot \frac{w_{1,i}}{||\mathbf{w}_1||_1} + \frac{||\mathbf{w}_2||_1}{||\mathbf{w}_1||_1 + ||\mathbf{w}_2||_1} \cdot \frac{w_{2,i}}{||\mathbf{w}_2||_1} \\ &= \frac{||\mathbf{w}_1||_1}{||\mathbf{w}_1||_1 + ||\mathbf{w}_2||_1} \cdot \mathcal{D}_1[i] + \frac{||\mathbf{w}_2||_1}{||\mathbf{w}_1||_1 + ||\mathbf{w}_2||_1} \cdot \mathcal{D}_2[i]\end{aligned}$$

Note that the target distribution $\mathcal{D}$ is a convex combination of the distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ held by the two parties.

A potentially straightforward sampling protocol is to therefore have party 1 locally draw a sample i_1 from $\mathcal{D}_1$, party 2 locally draw a sample i_2 from $\mathcal{D}_2$, and then run a secure two party computation that outputs i_1 with probability $\frac{||\mathbf{w}_1||_1}{||\mathbf{w}_1||_1+||\mathbf{w}_2||_1}$ and i_2 with probability $\frac{||\mathbf{w}_2||_1}{||\mathbf{w}_1||_1+||\mathbf{w}_2||_1}$.

This protocol clearly has sublinear communication, but it unfortunately does not securely realize the ideal functionality. The reason is as follows: conditioned on the ideal functionality outputting a certain index i^*, the probability that i^* was drawn by party 1 (resp. party 2) is $\frac{w_{1,i^*}}{w_{1,i^*}+w_{2,i^*}}$ (resp. $\frac{w_{2,i^*}}{w_{1,i^*}+w_{2,i^*}}$). Thus, if the simulator receives i^* from the ideal functionality and has to simulate the view of party 1, it needs to set $i_1 = i^*$ with probability $\frac{w_{1,i^*}}{w_{1,i^*}+w_{2,i^*}}$ and set $i_1 \neq i^*$ with probability $\frac{w_{2,i^*}}{w_{1,i^*}+w_{2,i^*}}$. However, the simulator is not able to simulate these probabilities correctly, since it does not know w_{2,i^*}.

To get around this issue we therefore have the parties sample i_1 and i_2 *obliviously.* To do this with sublinear communication, we can use fully homomorphic encryption (FHE). Specifically, to sample i_1, player 1 first encrypts his input $\mathbf{w}_1$ using an FHE scheme for which he does not know the secret key. The players then jointly choose a random value $r \in [0, ||\mathbf{w}_1||_1)$. Player 1 then uses the homomorphic operations to find the value i_1 chosen by this r, and the parties use threshold decryption to recover a secret sharing of i_1. The parties reverse roles to sample i_2. Details of this construction are provided in Section 4.2.

Additionally, an alternative construction that uses sub-linear OT for the oblivious sampling is provided in Section 7.2.3.

L_2 Sampling of component-wise sum. In this case, party 1 (resp. party 2) holds a vector $\mathbf{w}_1$ (resp. $\mathbf{w}_2$), indexed from 1 to n. For $i \in [n]$. The goal of the parties is to sample from the distribution $\mathcal{D}$ defined as follows for $i \in [n]$:

$$\mathcal{D}[i] := \frac{(w_{1,i} + w_{2,i})^2}{||\mathbf{w}_1 + \mathbf{w}_2||_2^2}.$$

We present a protocol that samples from this distribution with $\tilde{O}(1)$ communication. This protocol relies on a novel technique that we call "corrective sampling", which is an interesting type of rejection sampling. In what follows, we describe an insecure version of our protocol to give the intuition behind it. To make it secure, we carry out the corrective sampling under FHE as described in Protocol 12.

The main challenge that we face here, unlike in the case of L_1 sampling, is that it is impossible to compute $||\mathbf{w}_1 + \mathbf{w}_2||_2^2$ (and therefore impossible to compute $\mathcal{D}[i]$ for each i) with sublinear communication [11]. Instead, we sample index i from a different, related, distribution, which is easy to sample with sub-linear communication. We then show that we can efficiently correct this distribution by rejecting with the appropriate probability. Interestingly, we show that corrective rejection, which depends on the index i, doesn't require us to explicitly compute $||\mathbf{w}_1 + \mathbf{w}_2||_2^2$. In fact, the parties never learn the corrective term at all!

First, as in rejection sampling, corrective sampling proceeds in trials and in each trial, for every i, the probability that the protocol successfully samples index i is $\alpha \cdot \mathcal{D}[i]$ for some unknown constant $0 < \alpha < 1$. Since the same constant α is applied to every index i, by repeating the trials, the protocol samples index i correctly without skewing the distribution $\mathcal{D}$. The expected number of trials is $1/\alpha$. We therefore need to keep $1/\alpha \in O(1)$ to reach our target communication complexity.

As mentioned above, we observe that *the protocol never has to explicitly compute* α. Towards describing how this is done, first note that in $\mathcal{D}[i]$, the denominator, $||\mathbf{w}_1 + \mathbf{w}_2||_2^2$ – which we assume for purposes of this exposition is at least 1 – is the same for every i, so it can be pushed into α without impacting the discussion above: letting $\alpha' = \alpha/(||\mathbf{w}_1 + \mathbf{w}_2||_2^2)$, it suffices to implement rejection sampling with a protocol that samples index i with probability $\alpha' \cdot (w_{i,1} + w_{i,2})^2 = \alpha \cdot \mathcal{D}[i]$. This protocol would only need to explicitly compute $(w_{i,1} + w_{i,2})^2$ (which can be done efficiently given i), but not α'.

Unfortunately, this does not quite work. $||\mathbf{w}_1 + \mathbf{w}_2||_2^2$ can be very large, which would then make $1/\alpha'$ large. We therefore must combine the above with another idea to ensure that our corrective term introduces at most a $O(1)$ overhead.

We achieve this by having each trial of the protocol work as follows:

1. It samples index i from distribution $\mathcal{D}_{\text{ignore}}$, which is easy to sample. We note that the contribution of this distribution will be eventually canceled out through rejection. In particular, we choose the following distribution for $\mathcal{D}_{\text{ignore}}$:
$$\mathcal{D}_{\text{ignore}}[i] := \frac{w_{1,i}^2 + w_{2,i}^2}{\mathsf{denom}},$$
where we set $\mathsf{denom} = ||\mathbf{w}_1||_2^2 + ||\mathbf{w}_2||_2^2$ to make the distribution well-defined. Note that denom can be computed with $\tilde{O}(1)$ communication.

2. After sampling i from $\mathcal{D}_{\text{ignore}}$, the protocol computes a "corrective bias" for a coin flip that is dependent on $(w_{1,i} + w_{2,i})^2$. We stress that once i is determined, computing $(w_{1,i} + w_{2,i})^2$ is easy. In particular, a coin is flipped with the following bias:
$$\Pr[coin|i] := \frac{(w_{1,i} + w_{2,i})^2}{2 \cdot \mathcal{D}_{\text{ignore}}[i] \cdot \mathsf{denom}}$$

Overall, this makes sure that the probability that each trial outputs index i is
$$\mathcal{D}_{\text{ignore}}[i] \cdot \Pr[coin|i] = \frac{(w_{1,i} + w_{2,i})^2}{2 \cdot \mathsf{denom}} = \alpha \mathcal{D}[i],$$

where $\alpha = \frac{||\mathbf{w}_1+\mathbf{w}_2||_2^2}{2\cdot\mathsf{denom}}$.

To conclude that this is a valid and efficient sampling procedure, we need to show the following:

- α must be less than 1 for the procedure to be valid. This is implied by the fact that $\|\mathbf{w}_1 + \mathbf{w}_2\|_2^2 \leq 2 \cdot \mathsf{denom}$.
- $1/\alpha$ must be in $\tilde{O}(1)$ so that the procedure is efficient. We have $2 \cdot \mathsf{denom} \leq 2\|\mathbf{w}_1 + \mathbf{w}_2\|_2^2$, which implies that α is at least $1/2$. So, the expected number of trials is at most 2.

We extend our techniques to the setting of L_p sampling for constant p in Section 4.3.3.

Product sampling. In this case, party 1 (resp. party 2) holds a normalized vector $\mathbf{w}_1$ (resp. $\mathbf{w}_2$), indexed from 1 to n. For $i \in [n]$, $w_{1,i}$ (resp. $w_{2,i}$) corresponds to the probability mass of i under distribution $\mathcal{D}_1$ (resp. $\mathcal{D}_2$).[6] The goal of the parties is to sample from the distribution $\mathcal{D}$ defined as follows for $i \in [n]$:

$$\mathcal{D}[i] := \frac{w_{1,i} \cdot w_{2,i}}{\langle \mathbf{w}_1, \mathbf{w}_2 \rangle}.$$

We begin by noting (via a simple reduction from Set Disjointness) that it is impossible to achieve sublinear product sampling when no restrictions are placed on the inputs $\mathbf{w}_1, \mathbf{w}_2$. We further show (via a more complex reduction from Set Disjointness) that for every protocol Π (parametrized by dimension n) that correctly samples from $\mathcal{D}$, there are inputs $\mathbf{w}_1 := \mathbf{w}_1(n), \mathbf{w}_2 := \mathbf{w}_2(n)$, with $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle \in \Omega(1/n^2)$, that require linear communication complexity. See Section 4.4.1 for details.

This means that in order to achieve sublinear communication complexity, we would need–at the minimum–a promise on the inputs that guarantees that $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle \in \omega(1/n^2)$. We then present a protocol that has the following properties:

- When $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle \in \omega(\log n/n)$, the protocol achieves *expected* communication $\frac{\log n}{\langle \mathbf{w}_1, \mathbf{w}_2 \rangle}$.
- The execution of the protocol leaks nothing more than the sampled output, and $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$. This is formalized via an Ideal/Real paradigm simulation, in which the simulator receives leakage of $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$ in the Ideal world.

The idea for the protocol is the following. The protocol proceeds in rounds: in round j, party 1 and 2 obliviously sample values i_1, i_2 from $\mathcal{D}_1, \mathcal{D}_2$, respectively (as described for L_1 sampling). Then the parties run a secure protocol that checks whether $i_1 = i_2$. If yes, they output i_1. Otherwise, the parties repeat the process in the next round.

The main technical portion of our security analysis is to show that the number of rounds (which is the only information leaked) is distributed as a geometric distribution with success probability $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$. This implies that the expected

[6]Here the assumption that $\mathbf{w}$ are normalized is without loss of generality.

number of rounds is $1/\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$, and furthermore, it implies that a simulator who knows $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$ can simulate the terminating round by making a draw from this geometric distribution. See Section 4.4.2 for more details. There, we also describe how we can pad the communication cost to the worst-case, which depends on the given promise, thereby removing the leakage of $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$.

Product sampling in constant rounds. The protocol presented above for product sampling required a large number of rounds stemming from the iterative rejection sampling procedure. We now consider how to parallelize this process. To do so, we need to compute the inner product in order to determine, a priori, how many samples will suffice. However, computing this value requires $O(n)$ communication [11]!

The natural thing to do is therefore to use an approximation to the inner product that can be computed with sublinear communication. However, when replacing an exact computation of a function $f(\mathbf{w}_1, \mathbf{w}_2)$ with an approximation $\tilde{f}(\mathbf{w}_1, \mathbf{w}_2; r)$, one needs to be careful that *more* information is not leaked by the output. Specifically, Ishai et al. [76, 77] introduced the notion of secure multiparty computation of approximations and, loosely speaking, their security definition says that the approximate computation is secure if its output can be simulated from the exactly correct output. While our result falls slightly short of that definition, we are still able to give a rigorous guarantee on the amount of additional information leaked by our approximate functionality. Specifically, we present an approximate functionality $\tilde{f}$ and prove that the output of $\tilde{f}(\mathbf{w}_1, \mathbf{w}_2; r)$ can be simulated given both the exactly correct output $f(\mathbf{w}_1, \mathbf{w}_2)$ (where f is the inner product), as well as the L_2 norms of the individual inputs.

To achieve this, we use a sublinear protocol from the Johnson-Lindenstrauss Transform (JLT) to approximate the dot product of the input vectors. This can be done with sublinear communication by having the parties jointly sample a $k \times n$ JLT matrix $\mathbf{M}$ for $k \ll n$ by choosing a short seed and expanding it under FHE. The rest of the computation is then done by communicating vectors $\mathbf{M}\mathbf{w}_b$, which are of length k rather than n. Based on this approximation, the parties can obliviously pre-sample a number of inputs that is sufficient with all but negligible probability, and then input them into a constant round secure computation protocol.

Our contribution here, is to show that this variant protocol only requires additional leakage of $||\mathbf{w}_1||_2^2$, $||\mathbf{w}_2||_2^2$, beyond what is already leaked by the original protocol (i.e., the inner product). Our analysis may be of independent interest, since it shows that given $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$, $||\mathbf{w}_1||_2^2$, $||\mathbf{w}_2||_2^2$, the values $\mathbf{M}\mathbf{w}_1$ and $\mathbf{M}\mathbf{w}_2$ can be efficiently sampled from exactly the correct distribution, when $\mathbf{M}$ is a JLT matrix, and is kept private from both parties. We prove this result by analyzing the underlying joint multivariate normal distributions corresponding to $\mathbf{M}\mathbf{w}_1$ and $\mathbf{M}\mathbf{w}_2$, and showing that the mean and covariance (which fully determine the distribution) depend *only* on the values $||\mathbf{w}_1||_2, ||\mathbf{w}_2||_2$, and $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$ See Section 4.5 for more details.

Applications to distributed exponential mechanism. We first briefly describe the connection between product sampling and the exponential mechanism.

Ignoring many details, the joint exponential mechanism M outputs a value i on input $X = (x_1, \ldots, x_n)$ with probability proportional to

$$w_i = e^{c \cdot f(x_i)} = e^{c \cdot f(x_{1,i} + x_{2,i})},$$

where c is some constant, f is some scoring function, and the data values x_i are partitioned between the two parties (as $x_{1,i}, x_{2,i}$). If the scoring function f is linear, it holds that $f(x_{1,i} + x_{2,i}) = f(x_{1,i}) + f(x_{2,i})$, and, letting $w_{b,i} = e^{c \cdot f(x_{b,i})}$, we can rewrite w_i as follows:

$$w_i = w_{1,i} \cdot w_{2,i}.$$

Therefore, using product sampling, the parties can sample each item i with probability proportional to w_i.

Based on this connection, we present an application of our constant-round, product sampling protocol to realize a two-party exponential mechanism in Section 4.6. However, to use our sampling protocol in this application, we must show that the leakage of our protocols preserves the differential privacy guarantee. We indeed prove that our constant-round JLT-based protocol can achieve differential privacy—even when the JLT matrix $\mathbf{M}$ is public—by adding correctly distributed noise to $\langle \mathbf{M}\mathbf{w}_1, \mathbf{M}\mathbf{w}_2 \rangle$. This allows parties to execute the exponential mechanism when the cost function is additively distributed across the two parties, with sublinear communication, in the case that $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle \in \omega(\log n/n)$.

4.1.3 Related Work

Sampling from streaming data. Many prior papers (e.g. [49,83,115,140,182]) have studied the problem of sampling data from a data stream. In this setting the goal is to achieve L_p sampling for arbitrary p without having to process or store all the streaming data, thus requiring sublinear computation. These works generally operate in the one-party setting and do not consider privacy.

Secure multiparty sampling. A few prior works [153, 154] have investigated the problem of two and multi-party private sampling in the information theoretic setting. These works focus on identifying the necessary setup to enable sampling from various distributions. We instead focus on the computational setting, and focus on reducing communication. Recently, Champion et al. [39] also considered the computational setting, but they focus on sampling from a publicly-known distribution whereas we sample from a private one.

Secure multiparty computation of differentially private functionalities. Starting with the work of Dwork et al. [62] there has been a good amount of work (e.g. [1, 48, 70, 93, 149, 152]) on using MPC to realize differentially private functionalities to protect the privacy of individual inputs given the output of the MPC. These works have focused on building efficient, private applications in machine learning and other fields, whereas we focus on reducing the communication necessary for the specific functionalities of sampling.

Secure sketching. A long line of work [45, 69, 112, 136, 178] has investigated building secure sketches for securely estimating statistics of Tor usage, web traffic, and other applications. These works focus on building sublinear communication and computation protocols for computing specific statistics such as unique count, median, etc.

4.2 Two-party L_1 Sampling

In this section, we describe a secure two-party L_1 sampling protocol. Given two n-dimensional vectors $\mathbf{w}_1 = (w_{1,1}, \ldots, w_{1,n})$ and $\mathbf{w}_2 = (w_{2,1}, \ldots, w_{2,n})$ as the private inputs from parties P_1 and P_2 respectively, the protocol samples from the L_1 distribution according to $\mathbf{w}_1 + \mathbf{w}_2$.

Notation: L_p norm. Let $\mathbf{w} = (w_1, \ldots, w_n) \in \mathbb{R}^n$ be a non-zero vector. *The L_p norm $\|\mathbf{w}\|_p$ of* $\mathbf{w}$ is defined as $\|\mathbf{w}\|_p := \left(\sum_j |w_j|^p\right)^{1/p}$. When there is no subscript, it means L_2 norm; that is, $\|\mathbf{w}\| := \|\mathbf{w}\|_2$

Assumptions. Throughout the paper, we assume that the values $w_{b,i}$ are represented by fixed-point precision numbers, and consider the cost of communicating such a number to be independent of n. We assume all weights in vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ are non-negative.

Ideal functionality. We first define an ideal functionality for the two-party L_1 sampling. Slightly abusing the notation, let $L_1(\mathbf{w}_1, \mathbf{w}_2)$ be a two-input sampling procedure based on the L_1 distribution of $\mathbf{w}_1 + \mathbf{w}_2$:

$$\Pr[L_1(\mathbf{w}_1, \mathbf{w}_2) \text{ samples } i] = \frac{w_{1,i} + w_{2,i}}{\|\mathbf{w}_1 + \mathbf{w}_2\|_1}.$$

We give a more formal description of the functionality $\mathcal{F}_{L_1}$ in the figure below. In Section 4.2.2, we present a protocol that securely realizes this functionality.

$\mathcal{F}_{L_1}$: Ideal functionality for two-party L_1 sampling

The functionality has the following parameter:

- $n \in \mathbb{N}$. The dimension of the input weight vectors $\mathbf{w}_1$ and $\mathbf{w}_2$.

The functionality proceeds as follows:

1. Receive inputs $\mathbf{w}_1$ and $\mathbf{w}_2$ from P_1 and P_2 respectively.
2. Sample $i \in [n]$ with probability $\frac{w_{1,i}+w_{2,i}}{\|\mathbf{w}_1+\mathbf{w}_2\|_1}$
3. Send i to P_1 and P_2.

4.2.1 A toy protocol towards securely realizing $\mathcal{F}_{L_1}$

We describe our first attempt, which is insecure, but provides good intuition on how we construct a secure protocol. In fact, the attack on this broken protocol, as well as the fix presented in the next sub-section, remain relevant when we move to product sampling and L_2 sampling as well. Since we assume that all the weights are non-negative, we observe that letting $p = \frac{\|\mathbf{w}_1\|_1}{\|\mathbf{w}_1\|_1 + \|\mathbf{w}_2\|_1}$, the above measure can be re-written as follows:

$$\Pr[L_1(\mathbf{w}_1, \mathbf{w}_2) \text{ samples } i] = \frac{w_{1,i}}{\|\mathbf{w}_1\|_1} \cdot p + \frac{w_{2,i}}{\|\mathbf{w}_2\|_1} \cdot (1-p). \tag{1}$$

Equation (1) leads us to the following natural approach.

1. Party P_1 samples i_1 from the L_1 distribution according to $\mathbf{w}_1$, such that $\Pr[P_1 \text{ samples } i_1] = \frac{w_{1,i_1}}{\|\mathbf{w}_1\|_1}$.
2. Party P_2 samples i_2 from the L_1 distribution according to $\mathbf{w}_2$, such that $\Pr[P_2 \text{ samples } i_2] = \frac{w_{2,i_2}}{\|\mathbf{w}_2\|_1}$.
3. Then, P_1 and P_2 execute a secure protocol for the following procedure:
   (a) Execute a coin toss protocol with bias p. Let b be the output of the coin-flip.
   (b) If $b = 0$ (resp., $b = 1$), output i_1 (resp., i_2).

The output of the protocol will achieve correct sampling.

Insecurity of the protocol. However, this protocol has a subtle security issue. For example, let i be the eventual output index of the protocol. Then, we have the following:

- If the coin flip b is 0, which happens with probability p, it holds that i is always the same as i_1.
- On the other hand, if the coin flip b is 1, then i will be the same as i_1 if and only if $i_2 = i_1$, which happens with probability $\frac{w_{2,i_1}}{\|\mathbf{w}_2\|_1}$.

This implies that we have

$$\Pr[i = i_1 | i_1] = p + (1-p) \cdot \frac{w_{2,i_1}}{\|\mathbf{w}_2\|_1}$$

Now consider a distinguisher that corrupts P_1, chooses inputs $\mathbf{w}_1$ and $\mathbf{w}_2$, and checks the above conditional probability, which is possible *since the distinguisher can also see* i_1 *through the corrupted* P_1. To prove security, we should be able to construct a simulator for P_1 that fools this distinguisher. However, a simulator for P_1 doesn't know $\mathbf{w}_2$, which causes the above conditional probability to be unsimulatable.

In a sense, by having P_1 choose i_1, the protocol allows P_1 to measure the conditional probability $\Pr[i = i_1 | i_1]$, which depends on the value w_{2,i_1} thereby leaking information about P_2's input to P_1.

4.2.2 Secure L_1 sampling protocol

Oblivious sampling. We address the insecurity of the toy protocol by having the parties sample *obliviously* from $\mathbf{w}_1$, $\mathbf{w}_2$. This way, each party would not know whether the final output index matches the sample taken from its own vector, or the sample taken from the other party's vector. Specifically, we will construct our protocol under the framework described below:

1. The parties obliviously sample i_1 according to L_1 distribution of $\mathbf{w}_1$. The output index i_1 is secret shared between the two parties. Let $\langle i_1 \rangle$ denote the secret share of i_1. Likewise, they obliviously sample $\langle i_2 \rangle$ from L_1 distribution of $\mathbf{w}_2$.
2. Execute a secure two-party protocol to compute the following:
   (a) Flip a coin b with bias p.
   (b) If $b = 0$, output the decryption of i_1; otherwise output the decryption of i_2.

Ideal functionalities. Formally, we define an ideal functionality $\mathcal{F}_{\mathsf{osample(L_1)}}$ as follows:

$\mathcal{F}_{\mathsf{osample(L_1)}}$: Ideal functionality for oblivious L_1 sampling.

The functionality considers two participants, the sender and the receiver. The functionality is parameterized with a number n.

Inputs: The sender has an n-dimensional weight vector $\mathbf{w}$. The receiver has no input.

The functionality proceeds as follows:

1. Receive $\mathbf{w}$ from the sender.
2. Sample $i \in [n]$ with probability $\frac{w_i}{\|\mathbf{w}\|_1}$
3. Choose a random pad $\pi \in \{0,1\}^\ell$, where $\ell = \lceil \log_2 n \rceil$.
4. Send π to the sender and $i \oplus \pi$ to the receiver.

We also give an ideal functionality $\mathcal{F}_{\mathsf{biasCoin}}$ for the biased coin tossing.

$\mathcal{F}_{\mathsf{biasCoin}}$: Ideal functionality for biased coin tossing.

The functionality considers two participants P_1 and P_2 and proceeds as follows:

1. Receive a number s_1 as input from P_1 and s_2 from P_2.
2. Flip a coin b with bias $p = \frac{s_1}{s_1+s_2}$.
3. Choose a random bit $r \in \{0,1\}$.
4. Send r to P_1 and $r \oplus b$ to the receiver.

L_1 sampling protocol. Based on the above functionalities, we describe a protocol securely realizing $\mathcal{F}_{L_1}$ in the $(\mathcal{F}_{\mathsf{osample(L_1)}}, \mathcal{F}_{\mathsf{biasCoin}})$-hybrid.

Protocol 9 Two-party L_1 sampling in the $(\mathcal{F}_{\mathsf{osample(L_1)}}, \mathcal{F}_{\mathsf{biasCoin}})$-hybrid.

Inputs: Party P_b has input $\mathbf{w}_b$.

1. Execute $\mathcal{F}_{\mathsf{osample(L_1)}}$ with P_1 as a sender with input $\mathbf{w}_1$ and P_2 as a receiver. Let $\langle i_1 \rangle$ be the secret share of the output index.
2. Execute $\mathcal{F}_{\mathsf{osample(L_1)}}$ with P_2 as a sender with input $\mathbf{w}_2$ and P_1 as a receiver. Let $\langle i_2 \rangle$ be the secret share of the output index.
3. Execute $\mathcal{F}_{\mathsf{biasCoin}}$ where P_1 has input $\|\mathbf{w}_1\|_1$ and P_2 has input $\|\mathbf{w}_2\|_1$. Let $\langle b \rangle$ be the secret share of the output bit.
4. Execute $\mathcal{F}_{2PC}$ for the following circuit:
   (a) Input: $\langle i_1 \rangle, \langle i_2 \rangle, \langle b \rangle$.
   (b) Output: $i_1 \cdot (1-b) + i_2 \cdot b$.

Theorem 4.1. Protocol 9 securely realizes $\mathcal{F}_{L_1}$ with semi-honest security in the $(\mathcal{F}_{\mathsf{osample(L_1)}}, \mathcal{F}_{\mathsf{biasCoin}})$-hybrid.

The proof is found in Section 7.2.2.

Securely realizing $\mathcal{F}_{\mathsf{osample(L_1)}}$ with threshold FHE. The main idea of the protocol is having the parties securely sample a random number r from $[s]$, where $s := \|\mathbf{w}\|_1$. Our construction is found in Protocol 10.

Theorem 4.2. Assuming the existence of threshold FHE with IND-CPA security, Protocol 10 securely realizes $\mathcal{F}_{\mathsf{osample(L_1)}}$ in the semi-honest security model.

The proof is found in Section 7.2.2.

We note that we give another construction that relies on sub-linear 1-out-of-m oblivious transfer (OT), but requires computation that is exponential in the bit precision in Section 7.2.3.

Securely realizing $\mathcal{F}_{\mathsf{biasCoin}}$. The secure construction for $\mathcal{F}_{\mathsf{biasCoin}}$ is straightforward and can be found in Section 7.2.1.

4.3 Two Party L_2 Sampling

In this section we consider the two-party L_2 sampling functionality. Given input vectors $\mathbf{w}_1, \mathbf{w}_2$, this functionality samples from the distribution $D_{L_2}(\mathbf{w}_1, \mathbf{w}_2)$ with the following probability mass function:

$$\Pr[D_{L_2}(\mathbf{w}_1, \mathbf{w}_2) \text{ samples } i] = \frac{(w_{1,i} + w_{2,i})^2}{\sum_j (w_{1,j} + w_{2,j})^2} = \frac{(w_{1,i} + w_{2,i})^2}{\|\mathbf{w}_1 + \mathbf{w}_2\|_2^2}.$$

We begin by presenting a non-private protocol for two-party L_2 sampling with $\tilde{O}(1)$ communication in Section 4.3.1, the construction is found in Protocol 11. We then show how to implement the protocol securely in Section 4.3.2.

Protocol 10 Oblivious sampling from threshold FHE

Inputs: The sender has input $\mathbf{w} = (w_1, \ldots, w_n)$.

1. The sender computes $s := \|\mathbf{w}\|_1$.
2. The sender and the receiver execute $\mathcal{F}_{2PC}$ to uniformly sample r from the range $[0, s)$. This is possible, since s has a fixed point representation. Let r_1 and r_2 be the secret share of r given to P_1 and P_2 respectively.
3. The sender and the receiver set up a threshold FHE scheme. The plaintext space of the FHE is $GF(2)$, which allows homomorphic bitwise-xor and bitwise-AND operations. Let $[\![m]\!]$ denote an FHE encryption of plaintext m which can be a bit or bits depending on the context.
4. The receiver sends $[\![r_2]\!]$ so that the sender can compute $[\![r]\!] := [\![r_1]\!] \oplus [\![r_2]\!]$.
5. The sender homomorphically evaluates the following circuit:
   (a) Let $cnt_0 = 0$. For $j = 1, ..., n$, let $cnt_j = cnt_j + w_j$.
   (b) Output $i \in [1, n]$ such that $r \in [cnt_{i-1}, cnt_i]$.

   Let $[\![i]\!]$ be the output encryption from the above homomorphic evaluation.
6. The sender chooses a random pad π, and then it sends $[\![c]\!] = [\![i]\!] \oplus [\![\pi]\!]$ to the receiver.
7. The two parties perform threshold decryption so that c is decrypted to the receiver.
8. The sender outputs π and the receiver outputs the decryption of c.

4.3.1 A non-private L_2 sampling protocol with $\tilde{O}(1)$ communication

We begin by defining and showing how to sample from a helper distribution D_{ignore}.

Definition 4.3. For input vectors $\mathbf{w}_1, \mathbf{w}_2$, let $D_{\mathsf{ignore}}(\mathbf{w}_1, \mathbf{w}_2)$ be the distribution that "ignores" the cross term in $D_{L_2}(\mathbf{w}_1, \mathbf{w}_2)$. I.e. $D_{\mathsf{ignore}}(\mathbf{w}_1, \mathbf{w}_2)$ samples index $i \in [n]$ with probability $\frac{w_{1,i}^2 + w_{2,i}^2}{||\mathbf{w}_1||_2^2 + ||\mathbf{w}_2||_2^2}$.

Lemma 4.4. There exists a protocol Π_{ignore} for sampling from $D_{\mathsf{ignore}}(\mathbf{w}_1, \mathbf{w}_2)$ with $\tilde{O}(1)$ communication.

Proof. Let $\mathbf{w'}_b = (w_{b,1}^2, \ldots, w_{b,n}^2)$. The lemma follows by observing the following:

$$D_{\text{ignore}}(\mathbf{w}_1, \mathbf{w}_2) = D_{L_1}(\mathbf{w'}_1, \mathbf{w'}_2).$$

$\blacksquare$

Definition 4.5. For $i \in [n]$, let the corrective parameter function be defined as

$$f_c(\mathbf{w}_1, \mathbf{w}_2, i) := \frac{w_{1,i}^2 + 2w_{1,i}w_{2,i} + w_{2,i}^2}{||\mathbf{w}_1||_2^2 + ||\mathbf{w}_2||_2^2}.$$

Definition 4.6. The constant $c := c(\mathbf{w}_1, \mathbf{w}_2)$ is defined as

$$c(\mathbf{w}_1, \mathbf{w}_2) := \frac{||\mathbf{w}_1 + \mathbf{w}_2||_2^2}{||\mathbf{w}_1||_2^2 + ||\mathbf{w}_2||_2^2}$$

This ensures that for every i, $f_c(\mathbf{w}_1, \mathbf{w}_2, i) = c \cdot \Pr_{D_{L_2}(\mathbf{w}_1, \mathbf{w}_2)}[i]$.

The following lemma will be useful for arguing the validity of the final protocol.

Lemma 4.7. For all $i \in \mathsf{supp}(D_{L_2}(\mathbf{w}_1, \mathbf{w}_2))$, $\Pr_{D_{L_2}(\mathbf{w}_1, \mathbf{w}_2)}[i] \leq 2/c \cdot \Pr_{D_{\mathsf{ignore}}(\mathbf{w}_1, \mathbf{w}_2)}[i]$.

Proof.

$$\begin{aligned}
\Pr_{D_{L_2}(\mathbf{w}_1, \mathbf{w}_2)}[i] &= \frac{w_{1,i}^2 + 2w_{1,i}w_{2,i} + w_{2,i}^2}{||\mathbf{w}_1||_2^2 + 2\langle \mathbf{w}_1, \mathbf{w}_2 \rangle + ||\mathbf{w}_2||_2^2} \\
&= \frac{w_{1,i}^2 + 2w_{1,i}w_{2,i} + w_{2,i}^2}{c \cdot (||\mathbf{w}_1||_2^2 + ||\mathbf{w}_2||_2^2)} \\
&\leq \frac{2 \cdot (w_{1,i}^2 + w_{2,i}^2)}{c \cdot ||\mathbf{w}_1||_2^2 + ||\mathbf{w}_2||_2^2} \\
&= \frac{2}{c} \cdot \Pr_{D_{\mathsf{ignore}}(\mathbf{w}_1, \mathbf{w}_2)}[i]
\end{aligned}$$

The inequality holds since

$$\begin{aligned}2(w_{1,i}^2 + w_{2,i}^2) - (w_{1,i}^2 + 2w_{1,i}w_{2,i} + w_{2,i}^2) &= w_{1,i}^2 - 2w_{1,i}w_{2,i} + w_{2,i}^2 \\ &= (w_{1,i} - w_{2,i})^2 \\ &\geq 0.\end{aligned}$$

■

We now present the L_2 sampling protocol Π_{L_2}, which is described in Protocol 11. We show the correctness and efficiency of the protocol.

Protocol 11 Protocol for exact L_2 sampling (Π_{L_2})

Inputs: Parties P_1 and P_2 have inputs $\mathbf{w}_1$ and $\mathbf{w}_2$ respectively.

The protocol proceeds as follows:

1. Parties run Π_{ignore} with inputs $\mathbf{w}_1, \mathbf{w}_2$ that samples from $D_{\mathsf{ignore}}(\mathbf{w}_1, \mathbf{w}_2)$ and obtain output i.
2. For $b \in \{1, 2\}$, P_b sends $w_{b,i}, ||\mathbf{w}_b||_2^2$. Both parties compute

$$\Pr_{D_{\mathsf{ignore}}(\mathbf{w}_1,\mathbf{w}_2)}[i] = \frac{w_{1,i}^2 + w_{2,i}^2}{||\mathbf{w}_1||_2^2 + ||\mathbf{w}_2||_2^2} \quad \text{and} \quad f_c(\mathbf{w}_1, \mathbf{w}_2, i) = \frac{w_{1,i}^2 + 2w_{1,i}w_{2,i} + w_{2,i}^2}{||\mathbf{w}_1||_2^2 + ||\mathbf{w}_2||_2^2}$$

3. Parties output i with probability

$$\begin{aligned}\frac{f_c(\mathbf{w}_1, \mathbf{w}_2, i)}{2 \cdot \Pr_{D_{\mathsf{ignore}}(\mathbf{w}_1,\mathbf{w}_2)}[i]} &= \frac{c \cdot \Pr_{D_{L_2}(\mathbf{w}_1,\mathbf{w}_2)}[i]}{2 \cdot \Pr_{D_{\mathsf{ignore}}(\mathbf{w}_1,\mathbf{w}_2)}[i]} \\ &= \frac{\Pr_{D_{L_2}(\mathbf{w}_1,\mathbf{w}_2)}[i]}{2/c \cdot \Pr_{D_{\mathsf{ignore}}(\mathbf{w}_1,\mathbf{w}_2)}[i]}\end{aligned}$$

and otherwise return to step 1.

Lemma 4.8. With all but negligible probability, on inputs $\mathbf{w}_1, \mathbf{w}_2$, Π_{L_2} samples exactly correctly from $D_{L_2}(\mathbf{w}_1, \mathbf{w}_2)$, and has communication $\tilde{O}(1)$.

Proof. Note that Π_{L_2} simply performs rejection sampling in a distributed setting where sampling from $D_{\mathsf{ignore}}(\mathbf{w}_1, \mathbf{w}_2)$ and computing the probabilities is done in a distributed manner. It is therefore well-known that as long as for all $i \in [n]$,

$$\Pr_{D_{L_2}(\mathbf{w}_1,\mathbf{w}_2)}[i] \leq 2/c \cdot \Pr_{D_{\mathsf{ignore}}(\mathbf{w}_1,\mathbf{w}_2)}[i], \tag{2}$$

then Π_{L_2} samples from the exact correct distribution, and the number of samples required from $D_{\mathsf{ignore}}(\mathbf{w}_1, \mathbf{w}_2)$ in protocol Π_{L_2} follows a geometric distribution with probability $c/2$. Thus, if condition (2) is met, the protocol samples exactly correctly and completes in an expected $2/c$ (with $2/c \leq 2$, since $c \geq 1$) number of rounds. Further, it can be immediately noted that condition (2) is met due to Lemma 4.7. Finally, each round has $\tilde{O}(1)$ communication, since Π_{ignore} has communication $\tilde{O}(1)$ (by Lemma 4.4) and since, in addition to that, only a constant number of length $\tilde{O}(1)$ values are exchanged in each round. Combining the above, we have that Π_{L_2} has expected communication $\tilde{O}(1)$ and worst case (with all but negligible probability) communication $\tilde{O}(1)$. ■

Remark 4.9. Note that the protocol and analysis above did not require that vectors $\mathbf{w}_1, \mathbf{w}_2$ are normalized. I.e. we do not require that $||\mathbf{w}_1||_1$ or $||\mathbf{w}_2||_1$ are equal to 1 or to each other.

4.3.2 Secure L_2 Sampling From FHE

L_2 sampling protocol. We present our secure L_2 sampling protocol in Protocol 12. For two n-dimensional vectors $\mathbf{w}_1$ and $\mathbf{w}_2$, we denote by $\mathbf{w}_1 \odot \mathbf{w}_2$ the n-dimensional vector whose i-th entry is equal to $w_{1,i} \cdot w_{2,i}$.

Our L_2 sampling protocol uses ideal functionality $\mathcal{F}_{L_1}^{ss}$, which works essentially the same as $\mathcal{F}_{L_1}$ except that the output index is secret shared among both parties. We can securely realize this functionality with semi-honest security through a trivial change in the protocol Π_{L_1}; for the sake of completeness, we provide the details in Section 7.2.4.

Efficiency and correctness. It is clear that the total communication complexity of the protocol is $\tilde{O}(1)$, since each step in the loop has complexity $\tilde{O}(1)$ and the loop iterates $B \in \tilde{O}(1)$ number of times. Correctness is also immediate, since the protocol simply implements the Π_{L_2} sampling procedure, which was proven in Section 4.3.1 to be correct, and to require at most $B \in \tilde{O}(1)$ samples, with all but negligible probability,

Security. Security of our protocol is stated through the following theorem.

Theorem 4.10. Assuming the existence of threshold FHE with IND-CPA security, Protocol 12 securely realizes the L_2 sampling functionality in the $\{\mathcal{F}_{L_1}^{ss}, \mathcal{F}_{2PC}\}$-hybrid model with semi-honest security.

We provide the proof in Section 7.2.2.

4.3.3 A non-private L_p sampling protocol with $\tilde{O}(1)$ communication

In this section we present a $\tilde{O}(1)$ sampling protocol for L_p sampling for constant p. We present only the insecure version, extending it to a secure sampling protocol can be done entirely analogously to the construction for L_2 sampling given in Section 4.3.2.

Given input vectors $\mathbf{w}_1, \mathbf{w}_2$, L_p sampling refers to sampling from the distribution $D_{L_p}(\mathbf{w}_1, \mathbf{w}_2)$ with the following probability mass function:

$$\Pr[D_{L_p}(\mathbf{w}_1, \mathbf{w}_2) \text{ samples } i] = \frac{(w_{1,i} + w_{2,i})^p}{\sum_j (w_{1,j} + w_{2,j})^p} = \frac{(w_{1,i} + w_{2,i})^p}{\|\mathbf{w}_1 + \mathbf{w}_2\|_p^p}.$$

Protocol 12 Two-party L_2 sampling in the $(\mathcal{F}_{L_1}, F_{2PC})$-hybrid.

Inputs: Party P_b has input $\mathbf{w}_b$.

1. Let $B \in \tilde{O}(1)$. The parties perform the following steps for $j \in [B]$:
   (a) Sample from $D_{\mathsf{ignore}}(\mathbf{w}_1, \mathbf{w}_2)$ by doing the following: Invoke ideal functionality $\mathcal{F}^{ss}_{L_1}$ with P_1's input set to $\mathbf{w}_1 \odot \mathbf{w}_1$ and P_2's input set to $\mathbf{w}_2 \odot \mathbf{w}_2$. Let $\langle i_j \rangle$ be the secret share of the output index.

   (b) Parties compute encryptions of w_{1,i_j}, w_{2,i_j} using a threshold FHE scheme as follows.
      - Parties compute an encryption of i_j by exchanging encryptions of their shares and adding them.
      - Party b encrypts $\mathbf{w}_b$ and uses FHE to locally compute an encryption of w_{b,i_j}.
      - The parties then send these ciphertexts to each other.

   (c) Rejection Sampling. Compute a threshold FHE ciphertext $\widehat{\mathsf{bias}}_j$ that encrypts

$$\frac{f_c(\mathbf{w}_1, \mathbf{w}_2)_{i_j}}{2 \cdot \Pr_{D_{\mathsf{ignore}}(\mathbf{w}_1, \mathbf{w}_2)}[i_j]} = \frac{w^2_{1,i_j} + 2w_{1,i_j}w_{2,i_j} + w^2_{2,i_j}}{2(w^2_{1,i_j} + w^2_{2,i_j})}.$$

   Invoke ideal functionality $\mathcal{F}_{2PC}$ that takes encrypted bias $\widehat{\mathsf{bias}}_j$, the threshold decryption keys, index i_j, and random bits. The functionality executes a circuit that flips a coin with bias $\widehat{\mathsf{bias}}_j$ and returns a ciphertext $\widehat{\mathsf{out}}_j$, which is an encryption of i_j if the coin evaluates to 1 and an encryption of 0 otherwise.

2. Execute $\mathcal{F}_{2PC}$ for the following circuit:
   (a) Input: $(\widehat{\mathsf{out}}_1, \dots, \widehat{\mathsf{out}}_B)$ and threshold decryption keys.

   (b) Output: i_j corresponding to the minimum j such that $\widehat{\mathsf{out}}_j$ decrypts to $i_j \neq 0$. Or $\perp$ if no such $j \in [B]$ exists.

Protocol 13 Protocol for exact L_p sampling (Π_{L_p})

Inputs: Parties P_1 and P_2 have inputs $\mathbf{w}_1$ and $\mathbf{w}_2$ respectively.

The protocol proceeds as follows:

1. Parties run Π_{ignore} with inputs $\mathbf{w}_1, \mathbf{w}_2$ that samples from $D_{\mathsf{ignore},p}(\mathbf{w}_1, \mathbf{w}_2)$ and obtain output i.

2. For $b \in \{1, 2\}$, P_b sends $w_{b,i}, ||\mathbf{w}_b||_p^p$. Both parties compute

$$\Pr_{D_{\mathsf{ignore},p}(\mathbf{w}_1,\mathbf{w}_2)}[i] = \frac{w_{1,i}^p + w_{2,i}^p}{||\mathbf{w}_1||_p^p + ||\mathbf{w}_2||_p^p} \quad \text{and} \quad f_c(\mathbf{w}_1, \mathbf{w}_2, i) = \frac{(w_{1,i} + w_{2,i})^p}{||\mathbf{w}_1||_p^p + ||\mathbf{w}_2||_p^p}$$

3. Parties output i with probability

$$\begin{aligned} \frac{f_c(\mathbf{w}_1, \mathbf{w}_2, i)}{2^{p-1} \cdot \Pr_{D_{\mathsf{ignore},p}(\mathbf{w}_1,\mathbf{w}_2)}[i]} &= \frac{c \cdot \Pr_{D_{L_2}(\mathbf{w}_1,\mathbf{w}_2)}[i]}{2^{p-1} \cdot \Pr_{D_{\mathsf{ignore},p}(\mathbf{w}_1,\mathbf{w}_2)}[i]} \\ &= \frac{\Pr_{D_{L_2}(\mathbf{w}_1,\mathbf{w}_2)}[i]}{2^{p-1}/c \cdot \Pr_{D_{\mathsf{ignore},p}(\mathbf{w}_1,\mathbf{w}_2)}[i]} \end{aligned}$$

and otherwise return to step 1.

We begin by defining and showing how to sample from a helper distribution $D_{\mathsf{ignore},p}$.

Definition 4.11. For input vectors $\mathbf{w}_1, \mathbf{w}_2$, let $D_{\mathsf{ignore},p}(\mathbf{w}_1, \mathbf{w}_2)$ be the distribution that "ignores" the cross term in $D_{L_p}(\mathbf{w}_1, \mathbf{w}_2)$. I.e. $D_{\mathsf{ignore},p}(\mathbf{w}_1, \mathbf{w}_2)$ samples index $i \in [n]$ with probability $\frac{w_{1,i}^p + w_{2,i}^p}{||\mathbf{w}_1||_p^p + ||\mathbf{w}_2||_p^p}$.

Lemma 4.12. There exists a protocol Π_{ignore} for sampling from $D_{\mathsf{ignore},p}(\mathbf{w}_1, \mathbf{w}_2)$ with $\tilde{O}(1)$ communication.

Proof. Let $\mathbf{w}'_b = (w_{b,1}^p, \ldots, w_{b,n}^p)$. The lemma follows by observing the following:

$$D_{\text{ignore}}(\mathbf{w}_1, \mathbf{w}_2) = D_{L_1}(\mathbf{w}'_1, \mathbf{w}'_2).$$

■

Definition 4.13. For $i \in [n]$, let the corrective parameter function be defined as

$$f_c(\mathbf{w}_1, \mathbf{w}_2, i) := \frac{(w_{1,i} + w_{2,i})^p}{||\mathbf{w}_1||_p^p + ||\mathbf{w}_2||_p^p}.$$

Definition 4.14. The constant $c := c(\mathbf{w}_1, \mathbf{w}_2)$ is defined as

$$c(\mathbf{w}_1, \mathbf{w}_2) := \frac{||\mathbf{w}_1 + \mathbf{w}_2||_p^p}{||\mathbf{w}_1||_p^p + ||\mathbf{w}_2||_p^p}$$

This ensures that for every i, $f_c(\mathbf{w}_1, \mathbf{w}_2, i) = c \cdot \Pr_{D_{L_p}(\mathbf{w}_1,\mathbf{w}_2)}[i]$.

The following lemma will be useful for arguing the validity of the final protocol.

Lemma 4.15. For all $i \in \mathsf{supp}(D_{L_p}(\mathbf{w}_1, \mathbf{w}_2))$,

$$\Pr_{D_{L_p}(\mathbf{w}_1,\mathbf{w}_2)}[i] \leq 2^{p-1}/c \cdot \Pr_{D_{\mathsf{ignore},p}(\mathbf{w}_1,\mathbf{w}_2)}[i].$$

The proof is found in Section 7.2.2.

We now present the L_p sampling protocol Π_{L_p} in Protocol 13. We show the correctness and efficiency of the protocol below.

Lemma 4.16. With all but negligible probability, on inputs $\mathbf{w}_1$ and $\mathbf{w}_2$, protocol Π_{L_p} samples exactly correctly from $D_{L_p}(\mathbf{w}_1, \mathbf{w}_2)$. Further, for any constant p, the protocol has communication $\tilde{O}(1)$.

The proof is found in Section 7.2.2. We note that this result strictly generalizes Lemma 4.8. In particular, setting $p = 2$ in the above protocol yields a protocol with exactly the same parameters as the L_2 sampling protocol.

4.4 Two-party Product Sampling

We next consider the problem of two-party sampling from a product distribution. Specifically, given n-dimensional vectors $\mathbf{w_1} = (w_{1,1}, \ldots, w_{1,n})$ and $\mathbf{w_2} = (w_{2,1}, \ldots, w_{2,n})$ as the private inputs from P_1 and P_2 respectively, we wish to sample from the distribution D_{prod} defined by

$$\Pr[D_{\mathsf{prod}}(\mathbf{w}_1, \mathbf{w}_2) = i] = \frac{w_{1,i} \cdot w_{2,i}}{\sum_{j=1}^{n} w_{1,j} \cdot w_{2,j}} = \frac{w_{1,i} \cdot w_{2,i}}{\langle \mathbf{w}_1, \mathbf{w}_2 \rangle}$$

Of course, if $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle = 0$, the probability space is not well-defined, and in this case, we require the protocol to simply output $\perp$.

As before, we assume that all weights in $\mathbf{w}_1$ and $\mathbf{w}_2$ are non-negative.

Ideal functionality. We now define an ideal functionality $\mathcal{F}_{\mathsf{prod}}$ for two-party product sampling. This functionality is parametrized by a function f_{Leak} capturing the leakage that the functionality gives to the adversary.

$\mathcal{F}_{\mathsf{prod}}$: Ideal functionality for two-party product sampling

The functionality has the following parameters:

- $n \in \mathbb{N}$. The dimension of the input weight vectors $\mathbf{w}_1$ and $\mathbf{w}_2$.
- A function f_{Leak} describing the leakage.

The functionality proceeds as follows:

1. Receive inputs $\mathbf{w}_1$ and $\mathbf{w}_2$ from P_1 and P_2 respectively.
2. Compute $\mathsf{leak} = f_{\mathsf{Leak}}(\mathbf{w}_1, \mathbf{w}_2)$
3. If $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle = 0$, send leak to the adversary and $\perp$ to P_1 and P_2.
4. Otherwise, sample i with probability $\frac{w_{1,i} \cdot w_{2,i}}{\langle \mathbf{w}_1, \mathbf{w}_2 \rangle}$, send leak to the adversary, and send i to P_1 and P_2.

4.4.1 Impossibility of sublinear product sampling

Our goal is to find a protocol for two-party sampling with sublinear (in n) communication. However, unlike the case for L_1 sampling, we show that this goal is actually impossible. Roughly speaking, if parties are allowed to have arbitrary input vectors, then a sublinear communication solution to product sampling implies a sublinear communication solution to the disjointness problem, which is known to be impossible.

For our impossibility result, we first define the two-party disjointness problem.

Disjointness problem. The disjointness problem checks if two input sets S and T are disjoint (i.e., $S \cap T = \emptyset$). Specifically, we consider a function $\mathsf{DISJ}^n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ defined as:

$$\mathsf{DISJ}^n(v_S, v_T) = \begin{cases} 1 \text{ if } \langle v_S, v_T \rangle = 0 \\ 0 \text{ otherwise} \end{cases}$$

In the above, v_S and v_T are the characteristic vectors of S and T respectively. The communication complexity of the solution to the disjointness problem is known to have a linear lowerbound, as shown in the following Theorem:

Theorem 4.17 ([12, 118, 155]). For any (even non-private) two-party protocol Π where each party holds v_S and v_T respectively, if Π computes $\mathsf{DISJ}^n(v_S, v_T)$ correctly with probability at least $2/3$, the communication complexity of Π is $\Theta(n)$.

Our impossibility result. We first observe that a simple reduction from Disjointness gives us that is impossible to achieve sublinear product sampling. Specifically, disjointness can be directly learned from whether the product sampling protocol outputs $\perp$ or not.

Our impossibility result is stronger. We show that it is impossible to achieve sublinear product sampling *even when the product sampling protocol is executed with input vectors* $\mathbf{w}_1$ *and* $\mathbf{w}_2$ *in which all coordinates are bounded away from* 0, which in particular guarantees that $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$ is bounded away from 0.

Before stating a formal theorem below, for $0 < \gamma < 1$, we first define γ-heaviness; we say that a vector $\mathbf{w}$ is *γ-heavy* when each coordinate of $\mathbf{w}$ is a number contained in $[\gamma, 1]$.

Theorem 4.18. Let $\mathbf{w}_1$ and $\mathbf{w}_2$ be γ-heavy vectors of length n, each respectively held by P_1 and P_2. Assume there exists a two-party protocol Π_{prod} for the product sampling from $\mathbf{w}_1$ and $\mathbf{w}_2$, with communication at most $C := C(n, \gamma)$.

Then, for any $\gamma \leq 1/2n$, there exists a constant ρ and a probabilistic protocol computing DISJ^n correctly with probability at least $2/3$ that has communication at most $\log(n) + 1 + \rho \cdot (C + 1)$.

Proof of Theorem 4.18

We construct a protocol computing DISJ^n by taking advantage of Π_{prod} as follows:

The protocol for DISJ^n

Parties $\mathbf{A}$ and $\mathbf{B}$ each get as input a vector $\tilde{\mathbf{a}}$, $\tilde{\mathbf{b}} \in \{0,1\}^n$. The goal is to output 1 if the vectors are "disjoint" and 0 otherwise.

Edge Case: If one of the parties' inputs has Hamming weight 0, then they output 1 and send 1 to the other party. From now on, we assume that the Hamming weight of each party's input is at least 1.

Preamble: We call the party with the lower Hamming weight input the *designated party*. To determine this, **A** sends to **B** the Hamming weight of its input vector $\tilde{\mathbf{a}}$. If **B**'s input has higher Hamming weight, it sends back the bit 1 to **A**; otherwise it sends 0.

Input Transformation: Let $g_\gamma : \{0,1\} \to \mathbb{R}$ be a boosting function defined as $g_\gamma(0) = \gamma$ and $g_\gamma(1) = 1$. Each party **A**, **B** locally transforms their input vector $\tilde{\mathbf{a}}$, $\tilde{\mathbf{b}}$ to $\mathbf{a}$, $\mathbf{b}$ by applying the boosting function in order to ensure γ-heaviness. That is, for $i \in [n]$, set $a_i = g_\gamma(\tilde{a}_i)$ and $b_i = g_\gamma(\tilde{b}_i)$.

Sampling Protocol: The parties run the sampling protocol $\Pi_{\mathsf{prod}}(\mathbf{a}, \mathbf{b})$ and both receive some output i^*.

Output Computation: The *designated party* checks the i^*th bit of its input by which we denote x (i.e., $x = \tilde{a}_{i^*}$ or $x = \tilde{b}_{i^*}$ depending on which party is the designated party). It sends $1 - x$ to the other party. Both parties output $1 - x$.

The following lemmas give the completeness and soundness of the protocol.

Lemma 4.19. If $\tilde{\mathbf{a}}$, $\tilde{\mathbf{b}}$ are disjoint, then the parties both output 1 with probability at least $\frac{1}{2+n\cdot\gamma}$.

Lemma 4.20. If $\tilde{\mathbf{a}}$, $\tilde{\mathbf{b}}$ are not disjoint, then the parties both output 1 with probability at most $1 - \frac{1}{1+n\cdot\gamma}$.

Before we prove the lemmas, we briefly describe how we can use these lemmas to achieve a protocol that correctly computes DISJ with probability at least 2/3. Note that we can get a gap by setting $\gamma = \frac{1}{2n}$. In other words, parties output 1 when disjoint with probability at least $\frac{2}{5}$. Parties output 1 when not disjoint with probability at most $\frac{1}{3}$. Since we have a constant gap between completeness and soundness, this can be amplified to 2/3 and 1/3 by running the protocol a constant number of times.

Remarks. We would like to characterize the sublinearity condition for product sampling protocols using the normalized input vectors. We can do this since without loss of generality we can assume that input vectors to the product sampling protocols are normalized; in particular, for any (non-normalized) vectors $\mathbf{w}_1$ and $\mathbf{w}_2$, we have

$$\Pr\left[D_{\mathsf{prod}}\left(\frac{\mathbf{w}_1}{\|\mathbf{w}_1\|_1}, \frac{\mathbf{w}_2}{\|\mathbf{w}_2\|_1}\right) = i\right] = \frac{\frac{w_{1,i}}{\|\mathbf{w}_1\|_1} \cdot \frac{w_{2,i}}{\|\mathbf{w}_2\|_1}}{\langle \frac{\mathbf{w}_1}{\|\mathbf{w}_1\|_1}, \frac{\mathbf{w}_2}{\|\mathbf{w}_2\|_1} \rangle} = \Pr[D_{\mathsf{prod}}(\mathbf{w}_1, \mathbf{w}_2) = i].$$

Specifically, we show below that the impossibility theorem implies that in order to achieve sublinear communication complexity for product sampling, we would need, at the minimum, a promise on the inputs that guarantees that

$$\langle \mathbf{w}_1, \mathbf{w}_2 \rangle \in \Omega(1/n^2),$$

when $\mathbf{w}_1, \mathbf{w}_2$ are normalized vectors.

To do this, first note that the theorem implies that sublinear communication product sampling needs to have $\gamma \in \Omega(1/n)$. Now, in the proof, any non-disjoint binary vectors $\tilde{\mathbf{a}}$, $\tilde{\mathbf{b}}$ to the DISJ problem has $\langle \tilde{\mathbf{a}}, \tilde{\mathbf{b}} \rangle \geq 1$, and these vectors are transformed to $g_\gamma(\tilde{\mathbf{a}})$ and $g_\gamma(\tilde{\mathbf{b}})$. Let $\mathbf{w}_1$ and $\mathbf{w}_2$ be the normalized vectors $g_\gamma(\tilde{\mathbf{a}})$ and $g_\gamma(\tilde{\mathbf{b}})$; that is, $\mathbf{w}_1 := g_\gamma(\tilde{\mathbf{a}})/\|g_\gamma(\tilde{\mathbf{a}})\|_1$ and $\mathbf{w}_2 = g_\gamma(\tilde{\mathbf{b}})/\|g_\gamma(\tilde{\mathbf{b}})\|_1$. Since each entry of $g_\gamma(\tilde{\mathbf{a}})$ and $g_\gamma(\tilde{\mathbf{a}})$ is at most 1, we have $\|g_\gamma(\tilde{\mathbf{a}})\|_1 \leq n$ and $\|g_\gamma(\tilde{\mathbf{b}})\|_1 \leq n$. Therefore, we have

$$\langle \mathbf{w}_1, \mathbf{w}_2 \rangle \geq \frac{\langle g_\gamma(\mathbf{a}), g_\gamma(\mathbf{b}) \rangle}{n \cdot n} \geq \frac{1}{n^2}.$$

Proof of Lemma 4.19. Assume that $\tilde{\mathbf{a}}$, $\tilde{\mathbf{b}}$ are disjoint, and moreover, assume WLOG that $\mathbf{A}$ is the designated party, and its input vector has Hamming weight w. Recall that $a_i = g_\gamma(\tilde{a}_i)$ and $b_i = g_\gamma(\tilde{b}_i)$. Let

$$W_{0,0} := \sum_{i:\tilde{a}_i=0,\tilde{b}_i=0} a_i \cdot b_i, \qquad W_{1,0} := \sum_{i:\tilde{a}_i=1,\tilde{b}_i=0} a_i \cdot b_i$$
$$W_{0,1} := \sum_{i:\tilde{a}_i=0,\tilde{b}_i=1} a_i \cdot b_i, \qquad W_{1,1} := \sum_{i:\tilde{a}_i=1,\tilde{b}_i=1} a_i \cdot b_i$$

Note that $W_{0,0} \leq n \cdot \gamma^2$. Further, $W_{1,1} = 0$, since the vectors are disjoint, and $W_{1,0} = w \cdot \gamma$ since the Hamming weight of $\tilde{\mathbf{a}}$ is exactly w. Additionally, note that $W_{0,1} \geq W_{1,0}$, since $\mathbf{A}$ is the designated party, so the Hamming weight of $\tilde{\mathbf{a}}$ is less than or equal to the Hamming weight of $\tilde{\mathbf{b}}$.

Note that when the designated party is $\mathbf{A}$, then the output of the protocol is $1 - a_{i^*}$. Using the above facts, the probability of outputting 1 is

$$\begin{aligned}\frac{W_{0,0} + W_{0,1}}{W_{1,1} + W_{0,0} + W_{0,1} + W_{1,0}} &\geq \frac{W_{0,1}}{W_{0,0} + W_{0,1} + W_{1,0}} \\ &\geq \frac{W_{0,1}}{n\gamma^2 + 2W_{0,1}} \\ &= \frac{w \cdot \gamma}{n\gamma^2 + 2w \cdot \gamma} \\ &= \frac{w}{n\gamma + 2w} \\ &\geq \frac{1}{n\gamma + 2},\end{aligned}$$

where the last inequality follows since $w \geq 1$, due to the Edge Case step of the protocol. ■

Proof of Lemma 4.20. Assume that $\tilde{\mathbf{a}}$, $\tilde{\mathbf{b}}$ are not disjoint. As before, consider $W_{0,0}$, $W_{1,0}$, $W_{0,1}$, and $W_{1,1}$. Note that $W_{1,1} \geq 1$ since the inputs are not disjoint. We also have $W_{0,0} + W_{0,1} + W_{1,0} \leq n \cdot \gamma$, since a_i or b_i is γ in these cases.

Using the above facts, the probability of outputting 0 is

$$\begin{aligned}\frac{W_{1,0} + W_{1,1}}{W_{1,1} + W_{0,0} + W_{0,1} + W_{1,0}} &\geq \frac{W_{1,1}}{W_{1,1} + n \cdot \gamma} \\ &\geq \frac{1}{1 + n \cdot \gamma}.\end{aligned}$$

■

4.4.2 Product sampling while leaking at most the inner product

Assumptions. As before, we assume that all weights in $\mathbf{w}_1$ and $\mathbf{w}_2$ are non-negative. As discussed in the previous subsection, we also assume, without loss of generality, that

$$\|\mathbf{w}_1\|_1 = \|\mathbf{w}_2\|_1 = 1.$$

Overview. We now show that the impossibility result of Section 4.4.1 can be bypassed if we make some assumptions on the inputs. Specifically, if we restrict ourselves to the case when $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle = \omega\left(\frac{\log n}{n}\right)$, then we can achieve a sublinear communication protocol for product sampling on inputs $\mathbf{w}_1, \mathbf{w}_2$[7]. Of course, by observing that the protocol uses sub-linear communication, due to our lower-bound, both parties will learn that such a promise on the inputs is satisfied; the lower bound implies that some leakage about the inputs is necessary. In our protocol, we show that the information leaked is at most the inner product $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$. (Formally, we set $f_{\mathsf{Leak}}(\mathbf{w}_1, \mathbf{w}_2) = \langle \mathbf{w}_1, \mathbf{w}_2 \rangle$.) Interestingly, we show that this is the case even though our protocol does not, and cannot,[8] actually compute $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$.

Product sampling protocol. Roughly, the protocol works as follows. The protocol proceeds in rounds where in each round P_1 and P_2 use the oblivious L_1 sampling with a single input vector ($\mathcal{F}_{\mathsf{osample(L_1)}}$) to produce two secret-shared sampled indices, one from P_1's input vector, and one from P_2's input vector. The parties then run a secure 2-PC protocol to securely compare these values, and if they are equal, output the sampled index. If the two sampled indices are not equal, the parties move to the next round.

We describe a private two-party protocol for product sampling leaking at most the inner product (see Protocol 14). This protocol is in the $\{\mathcal{F}_{\mathsf{osample(L_1)}}, \mathcal{F}_{2PC}\}$-hybrid model.

Security. We will prove the following theorem.

[7]Regarding $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$, there is a gap between the lowerbound result (i.e., $\Omega(\frac{1}{n^2})$) and our construction (i.e., $\omega(\frac{\log n}{n})$). Resolving the gap is left as an interesting open problem.

[8]This can be shown by a simple modification of the lower bound proof from Section 4.4.1.

Protocol 14 Product sampling (Π^{IP}_{prod}) in the $\{\mathcal{F}_{\mathsf{osample(L_1)}}, \mathcal{F}_{2PC}\}$-hybrid.

Inputs: Party P_b has input $\mathbf{w}_b$ of length n.

1. Invoke the $\mathcal{F}_{\mathsf{osample(L_1)}}$ ideal functionality with P_1 as the sender with input $\mathbf{w}_1$ and P_2 as the receiver. Let $i_{1,1}$ and $i_{1,2}$ be the output from the ideal functionality to P_1 and P_2 respectively.
2. Invoke the $\mathcal{F}_{\mathsf{osample(L_1)}}$ ideal functionality with P_2 as the sender with input $\mathbf{w}_2$ and P_1 as the receiver. Let $i_{2,1}$ and $i_{2,2}$ be the output from the ideal functionality to P_1 and P_2 respectively.
3. Invoke the $\mathcal{F}_{2PC}$ ideal functionality with the following circuit:

   Input: $(i_{1,j}, i_{2,j})$ for $j = 1, 2$.

   (a) Let $i_1 = i_{1,1} \oplus i_{1,2}$, $i_2 = i_{2,1} \oplus i_{2,2}$.

   (b) If i_1 is equal to i_2, output i_1 to both P_1 and P_2. Otherwise, output $\perp$.
4. If the output from the ideal functionality is $\perp$, go back to Step 1. Otherwise, output whatever $\mathcal{F}_{2PC}$ outputs.

Output: Both parties output the sampled value i.

Theorem 4.21. Protocol Π^{IP}_{prod} securely realizes $\mathcal{F}_{\mathsf{prod}}$ with leakage $f_{\mathsf{Leak}}(\mathbf{w}_1, \mathbf{w}_2) = \langle \mathbf{w}_1, \mathbf{w}_2 \rangle$ in the $\{\mathcal{F}_{\mathsf{osample(L_1)}}, \mathcal{F}_{2PC}\}$-hybrid model with semi-honest security.

Proof. We describe the simulator Sim in the $\{\mathcal{F}_{\mathsf{osample(L_1)}}, \mathcal{F}_{2PC}\}$-hybrid model for the case that Party 1 is corrupted. The simulator and proof of security are analogous in the case that Party 2 is corrupted.

Sim receives as input $\mathbf{w}_1$, the output i^*, and $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$. Sim samples r^* from a geometric distribution with success probability $p = \langle \mathbf{w}_1, \mathbf{w}_2 \rangle$.

Sim invokes Party 1 on input $\mathbf{w}_1$. For $i \in [r^* - 1]$, Party 1 sends its input to the first invocation of $\mathcal{F}_{\mathsf{osample(L_1)}}$ and Sim returns to it a random value in $\mathbb{Z}_n$. Party 1 sends its input to the second invocation of $\mathcal{F}_{\mathsf{osample(L_1)}}$ and Sim returns to it a random value in $\mathbb{Z}_n$. Party 1 sends its input to the $\mathcal{F}_{2PC}$ functionality and Sim returns to it $\perp$. For $i = r^*$, Party 1 sends its input to the first invocation of $\mathcal{F}_{\mathsf{osample(L_1)}}$ and Sim returns to it a random value in $\mathbb{Z}_n$. Party 1 sends its input to the second invocation of $\mathcal{F}_{\mathsf{osample(L_1)}}$ and Sim returns to it a random value in $\mathbb{Z}_n$. Party 1 sends its input to the $\mathcal{F}_{2PC}$ functionality and Sim returns to it i^*.

It is clear that the view of Party 1 is identical in the ideal and real world, assuming that Sim samples the first succeeding round, r^*, from the correct distribution. In the following, we argue that this is indeed the case.

First, note that on any given round, we have

$$p_c := \Pr[\text{collision}] = \sum_i \Pr[i_i = i \wedge i_2 = i] = \sum_i w_{1,i} \cdot w_{2,i} = \langle \mathbf{w}_1, \mathbf{w}_2 \rangle.$$

Let $\mathsf{FirstSuccess}(r)$ denote an event in which the protocol succeeds for the first time on the r-th round. Now, for $r \in \mathbb{N}$, we have

$$\begin{aligned}
&\Pr[\mathsf{FirstSuccess}(r) \text{ AND the output is } i^*] \\
&= \Pr[\text{no collision in first } r-1 \text{ rounds}] \cdot \Pr[i_1 = i^* \wedge i_2 = i^* \text{ on the } r\text{th round}] \\
&= (1-p_c)^{r-1} \cdot \Pr[i_1 = i^* \wedge i_2 = i^*]
\end{aligned}$$

Now, the probability that the protocol eventually outputs i^* is:

$$\begin{aligned}
&\Pr[\text{protocol eventually outputs } i^* \text{ after some number of rounds}] \\
&= \sum_{j=1}^{\infty} \Pr[\mathsf{FirstSuccess}(j) \text{ AND the output is } i^*] \\
&= \Pr[i_1 = i^* \wedge i_2 = i^*] \sum_{j=1}^{\infty} (1-p_c)^{j-1} = \Pr[i_1 = i^* \wedge i_2 = i^*] \cdot \frac{1}{p_c}.
\end{aligned}$$

Thus, the probability of $\mathsf{FirstSuccess}(r)$ conditioned on the output being i^* is:

$$\begin{aligned}
&\Pr[\mathsf{FirstSuccess}(r) | \text{ the output is } i^*] \\
&= \frac{\Pr[\mathsf{FirstSuccess}(r) \text{ AND the output is } i^*]}{\Pr[\text{protocol eventually outputs } i^* \text{ after some number of rounds}]} \\
&= \frac{(\Pr[i_1 = i^* \wedge i_2 = i^*]) \cdot (1-p_c)^{r-1}}{\Pr[i_1 = i^* \wedge i_2 = i^*] \cdot \frac{1}{p_c}} \\
&= p_c \cdot (1-p_c)^{r-1}.
\end{aligned}$$

The above is exactly the probability of the number of Bernoulli trials (with probability $p_c = \langle \mathbf{w}_1, \mathbf{w}_2 \rangle$) needed to get one success. Sampling the number of rounds is therefore equivalent to sampling the random variable corresponding to the number of rounds from a geometric distribution with success probability $p_c = \langle \mathbf{w}_1, \mathbf{w}_2 \rangle$, which is exactly what Sim does. ■

Performance. As shown above, the number of rounds r needed by this protocol is distributed as the number of Bernoulli trials (with probability $p = \langle \mathbf{w}_1, \mathbf{w}_2 \rangle$) needed to get one success. Thus, the expected number of rounds is $r = \frac{1}{\langle \mathbf{w}_1, \mathbf{w}_2 \rangle}$. In each round, the communication consists of a secure 2-PC of equality on $O(\log n)$-bit inputs, which can be done in $O(\log n)$ communication and $O(1)$ rounds. Thus, in total, this protocol has expected communication $O(\frac{\log n}{\langle \mathbf{w}_1, \mathbf{w}_2 \rangle})$ and $O(\frac{1}{\langle \mathbf{w}_1, \mathbf{w}_2 \rangle})$ rounds. This communication is sublinear in n when $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle = \omega\left(\frac{\log n}{n}\right)$.

Trading efficiency for privacy. In the proof above, the simulator requires the value of $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$, which is not revealed by the output. However, a slight modification to the protocol allows us to remove this leakage at the cost of additional, though still sub-linear, communication. Instead of terminating the protocol the first time there is a collision in the L_1 samples, we can pad the communication cost by making $O(\frac{n}{\log n})$ calls to $\mathcal{F}_{\mathsf{osample(L_1)}}$. Under the promise of $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle = \omega(\frac{\log n}{n})$, this ensures a collision in the outputs (with all but negligible probability). The parties can then use $O(\frac{n}{\log n})$ communication to obliviously find and output the collision, without revealing the index, and avoiding the leakage of $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$.

Generalizing this idea, we arrive at a set of similar protocol modifications that support a continuous set of tradeoffs: instead of choosing between leaking $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$ to the simulator, or padding to the maximum communication, we can choose to leak some lower bound on $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$, and modify the protocol to make a proportionate number of calls to $\mathcal{F}_{\mathsf{osample(L_1)}}$, search (obliviously) for a collision, and repeat if necessary.

Without a full proof, we provide some intuition for the fact that this tradeoff between leakage and communication is inherent. We can do that by generalizing the statement of Theorem 4.18. We first modify the definition of γ-heavy defined previously: for any $t(n) = O(n)$, we say that a vector $\mathbf{w}$ of length n is $\gamma_{t,n}$-heavy if each of the $t := t(n)$ coordinates of $\mathbf{w}$ is a number contained in $[\gamma, 1]$. In particular, we now allow $t(n) = o(n)$. Then, with a small modification to the reduction, we can prove that if $\mathbf{w}_1$ and $\mathbf{w}_2$ are $\gamma_{t,n}$-heavy, and if there exists a protocol Π_{prod} for product sampling with communication at most $C := C(n, \gamma)$, then there exists a protocol for computing DISJ^t with communication $\log(n) + O(C)$. In the modified reduction, the parties simply increase the weights of the t input slots (as before), and append $n - t$ entries containing 0 at the end. Since we know that DISJ^t requires $O(t)$ communication, the implication is that we have increasingly weaker communication bounds as we are provided increasingly strong promises on the inner product. Conversely, for a certain set of input vectors, observing the communication of the sampling protocol gives you a bound on the inner product of the inputs. The less communication observed, the tighter that bound, and the greater the leakage.

4.5 Product Sampling in Constant Rounds

Achieving constant rounds through parallel repetition. In Sections 4.4, we showed a sublinear communication protocol for product sampling when $\langle \mathbf{w_1}, \mathbf{w}_2 \rangle$ is sufficiently large. Moreover, this protocol provably leaked no more information than the inner product. However, this protocol required $O(1/\langle \mathbf{w_1}, \mathbf{w}_2 \rangle)$ rounds of communication. This raises the question of whether constant-round sublinear product sampling is possible under the same restrictions on the inputs.

Our protocol to achieve this takes a relatively standard approach. Suppose that we are given the value of $\langle \mathbf{w_1}, \mathbf{w}_2 \rangle$. Then, since the expected number of samples until a collision is a function of $\langle \mathbf{w_1}, \mathbf{w}_2 \rangle$, we can just run the inner loop of protocol Π_{prod} in parallel sufficiently many times to guarantee that the protocol would terminate with all but negligible probability.

How many times to repeat? However, there is one catch. It is not actually possible to compute $\langle \mathbf{w_1}, \mathbf{w}_2 \rangle$ in sublinear communication! One simple solution is to use our promise on the input: we could run the inner loop enough times to guarantee termination for any inputs satisfying the promise (e.g. $\omega(\frac{n}{\log n})$ times). However, this forces us to adopt the worst-case communication cost, which might be undesirable. (Recall, it also offers the least leakage, which might be desirable.) Instead, we re-establish the trade-off between leakage and efficiency as follows. We begin by computing an approximation of the inner product in sublinear communication (see Section 4.5.1). Using this approximation, we can then realize our sublinear communication, constant round protocol for product sampling as follows in the next subsection.

4.5.1 Secure approximation of the inner product

We achieve a protocol that securely approximates the inner product with sublinear communication. In particular, we take advantage of the well known Johnson–Lindenstrauss Transform (JLT) [105, 114] sketch.

Additional assumptions about $\mathbf{w_1}$ and $\mathbf{w_2}$. We assumed that $\mathbf{w_1}$ and $\mathbf{w_2}$ are normalized and correlated such that $\langle \mathbf{w_1}, \mathbf{w_2} \rangle = \omega(\log n/n)$. In a similar vein, we assume that the cosine similarity of the two vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ is not small, e.g., $\omega(1/\log n)$.

Recall the cosine similarity between the two vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ is defined as $\cos(\mathbf{w}_1, \mathbf{w}_2) = \frac{\langle \mathbf{w}_1, \mathbf{w}_2 \rangle}{\|\mathbf{w}_1\|_2 \cdot \|\mathbf{w}_2\|_2}$. Since the L_1 norm of each vector is equal to 1, their L_2 norms will typically much smaller than 1, which implies that the cosine similarity is usually much larger than $\langle \mathbf{w_1}, \mathbf{w_2} \rangle$.

Approximating the inner product using JLT sketches. The JLT sketch of $\mathbf{x}$ is equal to $\mathbf{Mx}$, where $\mathbf{M}$ is a random $k \times n$ matrix with $k \ll n$. More specifically, the inner product of the two vectors is approximated as follows:

$\mathsf{approxIP}(\mathbf{w}_1, \mathbf{w}_2)$: $\triangleright$ $\mathbf{w}_1$ and $\mathbf{w}_1$ are n dimensional vectors.

1. Choose $k \times n$ matrix $\mathbf{M}$ such that each entry $M_{i,j}$ is chosen from an independent Gaussian distribution of mean 0 and variance 1.
2. Output $\frac{1}{k} \cdot \langle \mathbf{M}\mathbf{w}_1, \mathbf{M}\mathbf{w}_2 \rangle$. (Here, we slightly abuse the notation and treat the vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ as column vectors.)

Lemma 4.22. (cf. [117, Corollary 3.1]) For all $\mathbf{w}_1, \mathbf{w}_2$ such that $\cos(\mathbf{w}_1, \mathbf{w}_2) \geq t$, the procedure $\mathsf{approxIP}(\mathbf{w}_1, \mathbf{w}_2)$ approximates $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$ up to a $1 \pm \epsilon$ approximation factor with all but negligible probability (over the choice of the JLT matrix), using JLT dimension $k = \omega\left(\frac{\log(n)}{t^2 \cdot \epsilon^2}\right)$.

Privacy of the approximate output. What is interesting is that the approximate inner product doesn't reveal anything more than the inner product itself. In this sense, it satisfies the notion of private approximation introduced in [78]. In particular, we prove the following:

Lemma 4.23. The output of $\mathsf{approxIP}(\mathbf{w}_1, \mathbf{w}_2)$ can be simulated perfectly given only $\langle \mathbf{w}_1, \mathbf{w}_1 \rangle$, $\langle \mathbf{w}_2, \mathbf{w}_2 \rangle$, and $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$.

The proof is found in Section 7.2.2.

Private protocol via JLT. Using the JLT sketch, we can design a private protocol approximating the inner product. See Protocol 15. The protocol uses threshold FHE (e.g., [141]).

Protocol 15 Private protocol for computing approximate inner product

---

Inputs: Parties P_1 and P_2 has inputs $\mathbf{w}_1$ and $\mathbf{w}_2$ respectively.

The protocol proceeds as follows:

1. Parties set up a threshold FHE scheme.
2. They securely sample $k \times n$ matrix $\mathbf{M}$ described in the above with in the threshold FHE. In particular, they jointly generate an encrypted random seed $[\![s]\!]$. Using this randomness, parties homomorphically evaluates $[\![PRG(s)]\!]$, where PRG is a pseudorandom generator, to obtain the JLT matrix $[\![\mathbf{M}]\!]$.
3. Each party P_b homomorphically evaluates $[\![\tilde{\mathbf{w}}_b]\!] = [\![\mathbf{M}\mathbf{w}_b]\!]$.
4. Party P_1 sends $[\![\tilde{\mathbf{w}}_1]\!]$ to P_2.
5. Party P_2 homomorphically evaluates $[\![\langle \tilde{\mathbf{w}}_1, \tilde{\mathbf{w}}_2 \rangle]\!]$ and sends it to P_1.
6. Parties execute threshold decryption to obtain and output $\frac{1}{k} \cdot \langle \tilde{\mathbf{w}}_1, \tilde{\mathbf{w}}_2 \rangle$.

Security. Since every protocol message is a ciphertext, based on semantic security of the threshold FHE, it is easy to see that the protocol securely realizes a functionality for computing $\mathsf{approxIP}$. Based on Lemma 4.23, the leakage profile of the functionality is $\langle \mathbf{w}_1, \mathbf{w}_1 \rangle$, $\langle \mathbf{w}_2, \mathbf{w}_2 \rangle$, and $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$.

4.5.2 Constant-round protocol for product sampling

Note that the Protocol 14 has the following structure. In particular:

- The probability that Protocol 14 samples a good index and halts in a given trial is $p = \langle \mathbf{w}_1, \mathbf{w}_2 \rangle$.

We need to repeat r trials in parallel so that the probability that all r trials fail is negligible. In other words, we should have

$$(1-p)^r \leq e^{-p \cdot r} \leq e^{-\omega(\log \kappa)}.$$

This means that we should have $r > \frac{\omega(\log \kappa)}{p}$.

Moreover, in the previous subsection, we discussed how to obtain a good estimate $\tilde{p} = (1 \pm \epsilon)p$. Therefore, we should have

$$r > \frac{(1+\epsilon) \cdot \omega(\log \kappa)}{\tilde{p}} > \frac{\omega(\log \kappa)}{p}.$$

In summary, by running $\frac{(1+\epsilon)\cdot\omega(\log\kappa)}{\tilde{p}}$ instances in parallel, we achieve constant round protocols for product sampling with negligible failure probability. The final protocol should perform extra steps to hide from which trial the output comes from, and these changes can made in a straightforward way.

4.6 Two-party Exponential Mechanism

Recall that one of our main motivations for this work is to instantiate a two-party version of the exponential mechanism to achieve differential privacy. We observe that for many natural loss functions (i.e., when the loss function is additive across the two parties), the exponential mechanism on two parties is essentially equivalent to product sampling. We explain this further with a concrete example in Section 4.6.1.

4.6.1 A concrete example

Suppose we want to choose a classifier minimizing the L_2 error over a test dataset while preserving differential privacy of the labeled examples. Suppose there are n machine learning classifiers $(c_1, \ldots, c_n)$, and a test dataset $D = (d_1, \ldots, d_{|D|})$ consists of $|D|$ rows. Let $\ell_j \in \{0, 1\}$ be the label of the j-th row d_j of the dataset. For a machine learning classifier c_i, we define its L_2 loss function as follows:

$$f^{c_i}_{\mathsf{loss}}(D) := \sum_{j\in|D|} (c_i(d_j) - \ell_j)^2/|D|.$$

Now, consider a two-party federated setting in which the parties would like to perform computation on the aggregation of their local datasets. In particular, we assume party P_1 (resp., party P_2) holds dataset D_1 (resp., D_2) with $|D_1| = |D_2|$. Let $D = D_1||D_2$.

DP mechanism in the central curator model. In our mechanism, the central curator would receive input from parties P_1 and P_2 and choose classifier c_i with a $(\epsilon, 0)$-DP guarantee using the exponential mechanism.

We observe that the L_2 loss function $f^{c_i}_{\mathsf{loss}}(D)$ over the entire dataset D can be computed by each party P_b *first locally computing*

$$f^{c_i}_{\mathsf{loss}}(D_b) := \sum_{j\in|D_b|} (c_i(d_{b,j}) - \ell_{b,j})^2/|D|,$$

and then computing $f^{c_i}_{\mathsf{loss}}(D) = f^{c_i}_{\mathsf{loss}}(D_1) + f^{c_i}_{\mathsf{loss}}(D_2)$.

Based on the above observation, in our mechanism, each party P_b computes a vector $\mathbf{v}_b$ as follows:

> For $b \in \{1, 2\}$, let $\mathbf{v}_b = (v_{b,1}, \ldots, v_{b,n})$, where $v_{b,i} = e^{-\epsilon\cdot\frac{f^{c_i}_{\mathsf{loss}}(D_b)}{2\Delta u}}$ and $\Delta u = \frac{\epsilon}{40(\log n+t)}$, and κ is the security parameter.

Then, each party P_b computes $\mathbf{w}_b := \frac{\mathbf{v}_b}{||\mathbf{v}_b||_1}$ (i.e., the normalization of $\mathbf{v}_b$), and sends $\mathbf{w}_b$ to the central curator. Finally, the curator will choose classifier c_i with the probability $\frac{w_{i,i}\cdot w_{i,2}}{\langle\mathbf{w}_1,\mathbf{w}_2\rangle}$.

Lemma 4.24. If $|D| \geq 40(\log n + \kappa)/\epsilon$, our mechanism provides $(\epsilon, 0)$-DP.

Proof. Let $q_i(D) = \frac{-f_{\mathsf{loss}}^{c_i}(D)}{2\Delta u}$. We first show that drawing a sample from the product distribution of $\mathbf{w}_1, \mathbf{w}_2$ is identical to running the exponential mechanism to select classifier c_i with probability

$$\frac{e^{\epsilon q_i(D)}}{\sum_{j \in [n]} e^{\epsilon q_j(D)}}.$$

Recall that product sampling on inputs $\mathbf{w}_1, \mathbf{w}_2$ returns index i with probability $\frac{w_{1,i} \cdot w_{2,i}}{\langle \mathbf{w}_1, \mathbf{w}_2 \rangle}$. Since $w_{b,i} = \frac{v_{b,i}}{||\mathbf{v}_b||_1}$, we can rewrite the previous equation as

$$\frac{w_{1,i} \cdot w_{2,i}}{\langle \mathbf{w}_1, \mathbf{w}_2 \rangle} = \frac{v_{1,i} \cdot v_{2,i}}{\langle \mathbf{v}_1, \mathbf{v}_2 \rangle} \quad = \frac{e^{\epsilon \cdot q_i(D_1)} \cdot e^{\epsilon \cdot q_i(D_2)}}{\sum_{j \in [n]} e^{\epsilon \cdot q_j(D_1)} \cdot e^{\epsilon \cdot q_j(D_2)}} = \frac{e^{\epsilon \cdot q_i(D)}}{\sum_{j \in [n]} e^{\epsilon \cdot q_j(D)}}$$

Note that the loss functions have global sensitivity of $1/|D|$ since a change to the j-th row of D_b can cause $f_{\mathsf{loss}}^{c_i}(D_b)$ to change by $\pm 1/|D|$. Instead of using $1/|D|$, our mechanism uses a larger value Δu. Since the Δu is larger than the sensitivity whenever $|D| \geq 40(\log n + \kappa)/\epsilon$, and our mechanism is a simple exponential mechanism, $(\epsilon, 0)$-DP holds. ∎

Utility. Although using a larger value Δu deteriorates the utility of the mechanism, we show that the utility is still acceptable. Applying Theorem 3.11 in [65] to our setting, we have

$$\Pr\left[f_{\mathsf{loss}}^{c_i}(D) \leq f_{\mathsf{loss}}^{c_{opt}}(D) - \frac{2\Delta u}{\epsilon} \cdot (\log n + \kappa)\right] \leq e^{-\kappa}.$$

Noting that $\Delta u = \frac{\epsilon}{40(\log n + \kappa)}$, we have

$$\Pr\left[f_{\mathsf{loss}}^{c_i}(D) \leq f_{\mathsf{loss}}^{c_{opt}}(D) - 1/20\right] \leq e^{-\kappa}.$$

Viewing $f_{\mathsf{loss}}^{c_{opt}}(D)$ as the optimal accuracy for the chosen classifier. This implies that our mechanism returns a classifier that is at most 5% less accurate than the optimal classifier. We note that an even smaller loss in accuracy can be achieved by increasing Δu and the minimum size of D accordingly.

Jumping ahead, we use this larger Δu in order to achieve differential privacy of the approximate inner product evaluation to be described in the next section.

4.6.2 Differentially-Private Inner Product for the Exponential Mechanism

Issue: DP is broken due to leakage $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$. Based on the result of the previous subsection, we can simply run the product sampling protocol to achieve a two-party exponential mechanism without the central curator. However, there is one issue we need to address. In particular, the leakage from the previously described protocols for product sampling violates the DP guarantee of the exponential mechanism; the leakage $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$ is clearly not differentially private with respect to P_2's input.

Thus, to instantiate the exponential mechanism, we give an alternative inner product approximation protocol that achieves differential privacy. Using this approximation, we can build a protocol that is able to sample from exactly the product distribution while additionally leaking a value leak that is differentially-private and thus does not violate the DP guarantee of the exponential mechanism. We build such a protocol based on the approximate inner product using the JLT given in Section 4.5.1.

Approximating the inner product differentially privately. We now describe a mechanism, executed by a trusted curator, to approximate the inner product on inputs $\mathbf{w}_1$ and $\mathbf{w}_2$ with $w_{b,i} = \frac{v_{b,i}}{||\mathbf{v}_b||_1}$ for $b \in \{1, 2\}$ and $i \in [n]$ as described above. This mechanism is essentially the $\mathsf{approxIP}$ algorithm described in Section 4.5.1 with noise added in the exponent to guarantee differential privacy.

$\mathsf{DP\text{-}approxIP}(\mathbf{w}_1, \mathbf{w}_2)$:

1. Choose $k \times n$ matrix $\mathbf{M}$ such that each entry $M_{i,j}$ chosen from an independent Gaussian distribution of mean 0 and variance 1. The dimension k (with $k \ll n$) is determined appropriately according to Lemma 4.22.
2. Choose a value x from the Laplace distribution $\mathsf{Lap}(1/(\Delta u \cdot |D|))$.
3. Output $e^x \cdot \langle \mathbf{M}\mathbf{w}_1, \mathbf{M}\mathbf{w}_2 \rangle$.

Public sampling of M. Contrary to Section 4.5 where $\mathbf{M}$ was sampled inside the FHE, here, the matrix $\mathbf{M}$ can be publicly sampled (e.g., through a commonly chosen random PRG seed), since DP is achieved through adding a Laplace noise.

Differential privacy. We say that two datasets D and D' are *neighboring* if they differ in exactly one row.

Recall that, as in the application described in Section 4.6.1, the parties' inputs to the product sampling are of the form $\mathbf{w}_b$ where $w_{b,i} \propto e^{-\epsilon \cdot \frac{f_{\mathsf{loss}}^{c_i}(D_b)}{2\Delta u}}$ for some additive loss functions $f_{\mathsf{loss}}^{c_i}$.

Theorem 4.25. If $|D| \geq 40 \cdot (\log n + \kappa)/\epsilon$, the mechanism $\mathsf{DP\text{-}approxIP}$ is ϵ-differentially private w.r.t a database D_1 (resp., D_2) when the loss functions $f_{\mathsf{loss}}^{c_i}(D)$ have low sensitivity $1/|D|$.

Proof. We prove differential privacy with respect to D_1 (i.e., the adversary knows D_2 and the differential privacy guarantee holds relative to rows of D_1). The reverse case is analogous.

Note that the change of a single row in D_1 causes the values $v_{1,i}$ (for $i \in [n]$) and the value $\|\mathbf{v}_1\|_1$ to change by at most a multiplicative factor of $\alpha := e^{\pm\epsilon/(2\Delta u \cdot |D|)}$. Thus, letting $\mathbf{M}_\ell$ be the ℓ-th row of the JLT matrix (which is fixed and public), for each $\ell \in [k]$, the value $\langle \mathbf{M}_\ell, \mathbf{w}_b \rangle$ can change by at most a multiplicative factor of α^2. Therefore, the dot product $\langle \mathbf{Mw}_1, \mathbf{Mw}_2 \rangle$ can change by at most a multiplicative factor of α^2.

Ultimately, this means that we have additive sensitivity of $\epsilon/(\Delta u \cdot |D|)$ in the *exponent*. To achieve differential privacy, we thus need to multiply the dot product estimate by e^x, where x is drawn from $\mathsf{Lap}(1/(\Delta u \cdot |D|))$. ■

Correctness. We briefly analyze the estimation error due to the added noise. Since x is drawn from $\mathsf{Lap}(1/(\Delta u \cdot |D|))$, the probability that $|x| \geq \epsilon$ is at most $e^{-\Delta u \cdot |D|}$. When $|D| > \Delta u \cdot \kappa/\epsilon$, this probability is at most $e^{-\kappa}$, which is negligible.

In other words, when D is a sufficiently large dataset, with overwhelming probability, the incurred multiplicative error is $e^{|x|} \leq e^{\epsilon} < 1 + 2\epsilon$. Thus, differential privacy adds at most a $1 \pm 2\epsilon$ multiplicative error on top of the error of the approximation algorithm.

Removing the curator. We described the inner product approximation protocol as being run by a trusted curator. As is standard, we can replace this curator with a secure 2-PC evaluating the mechanism to achieve computational DP.

4.6.3 Instantiating the Exponential Mechanism

We now have all the necessary pieces to instantiate a sublinear communication protocol to evaluate the exponential mechanism for a database D held jointly by two parties.

The two-party exponential mechanism protocol. We now describe the distributed exponential mechanism where P_b has input D_b and the loss functions have low sensitivity. This protocol is in the $\mathcal{F}_{\mathsf{osample(L_1)}}$-hybrid model.

Security. We will prove the following theorem.

Theorem 4.26. If $|D| \geq 40 \cdot \kappa(\log n + \kappa)/\epsilon^2$, protocol Π_{EM} is $(2\epsilon, \mathsf{negl}(\kappa))$-DP.

It is easy to see that this protocol runs an enough number of product samplings in parallel so that it does not output abort, except with negligible probability (see Section 4.5.2). Therefore, for the proof, we assume that the protocol does not output abort.

Proof. We first consider where P_1 is corrupted by the adversary A. Let $\mathsf{view}_A(D)$ be the view of the protocol to A (consisting of input, output and transcript) in the $\{\mathcal{F}_{2PC}, \mathcal{F}_{\mathsf{osample(L_1)}}\}$-hybrid model. In the j-th invocation of $\mathcal{F}_{\mathsf{osample(L_1)}}$, the outputs $\{i^j_{1,1}, i^j_{2,1}\}_j, i)$ sent to A by the ideal functionality are uniformly distributed and independent of D. Thus, WLOG, we assume that A's view consists of its input, transcript η, and output i. Let $\mathsf{view}^{\mathsf{trans}}_A(D)$ (resp., $\mathsf{view}^{\mathsf{out}}_A(D)$) be the transcript (resp., output) contained in A's view during a random execution of the protocol with input D.

Protocol 16 Exponential Mechanism Protocol (Π_{EM}) in the $\{\mathcal{F}_{2PC}, \mathcal{F}_{\mathsf{osample}(\mathsf{L}_1)}\}$-hybrid model

Inputs: Party P_b has input D_b

1. P_b computes $\mathbf{v}_b = (v_{b,1}, \ldots, v_{b,n})$ such that $v_{b,i} = e^{-\epsilon \cdot \frac{f_{\mathsf{loss}}^{c_i}(D^b)}{2\Delta u}}$, where $\Delta u = \frac{\epsilon}{40(\log n + \kappa)}$. Let $\mathbf{w}_b = \frac{\mathbf{v}_b}{||\mathbf{v}_b||_1}$.
2. Invoke the $\mathcal{F}_{2PC}$ ideal functionality to evaluate $\eta = \mathsf{DP\text{-}approxIP}(\mathbf{w}_1, \mathbf{w}_2)$. Note that η is an $(1 + 2\epsilon)$-approximation of the inner product.
3. The parties execute the following steps $m = \frac{(1+2\epsilon)\cdot\omega(\log\kappa)}{\eta}$ times in parallel.
   (a) Invoke the $\mathcal{F}_{\mathsf{osample}(\mathsf{L}_1)}$ ideal functionality with P_1 as the sender with input $\mathbf{w}_1$ and P_2 as the receiver. Let $i_{1,1}^j$ and $i_{1,2}^j$ be the output of the jth execution to P_1 and P_2 respectively.
   (b) Invoke the $\mathcal{F}_{\mathsf{osample}(\mathsf{L}_1)}$ ideal functionality with P_2 as the sender with input $\mathbf{w}_2$ and P_1 as the receiver. Let $i_{2,1}^j$ and $i_{2,2}^j$ be the output of the jth execution to P_1 and P_2 respectively.
4. Invoke the $\mathcal{F}_{2PC}$ ideal functionality for the following circuit:

   Input: $(i_{1,1}^j, i_{2,1}^j, i_{1,2}^j, i_{2,2}^j)$ for $j = 1, \ldots, m$.

   (a) Let $i_1^j = i_{1,1}^j \oplus i_{1,2}^j$, $i_2^j = i_{2,1}^j \oplus i_{2,2}^j$.
   (b) Find the smallest j such that i_1^j equals i_2^j, and output i_1^j to both P_1 and P_2. If no such j exists, output abort.

Output: Both parties output the sampled index i or abort.

For all neighboring database D and D', and for all η and i, we have:

$$\begin{aligned}
&\Pr[\mathsf{view}_A^{\mathsf{trans}}(D) = \eta, \mathsf{view}_A^{\mathsf{out}}(D) = i] \\
&= \Pr[\mathsf{view}_A^{\mathsf{trans}}(D) = \eta] \cdot \Pr[\mathsf{view}_A^{\mathsf{out}}(D) = i|\eta] \\
&\le e^{\epsilon} \Pr[\mathsf{view}_A^{\mathsf{trans}}(D') = \eta] \cdot \Pr[\mathsf{view}_A^{\mathsf{out}}(D) = i|\eta] \\
&\le e^{\epsilon} \Pr[\mathsf{view}_A^{\mathsf{trans}}(D') = \eta] \cdot e^{\epsilon} \Pr[\mathsf{view}_A^{\mathsf{out}}(D') = i|\eta] \\
&= e^{2\epsilon} \Pr[\mathsf{view}_A^{\mathsf{trans}}(D') = \eta, \mathsf{view}_A^{\mathsf{out}}(D') = i],
\end{aligned}$$

The first inequality holds form the DP of protocol DP-approxIP, and the second inequality holds from the DP of the exponential mechanism.

The case when P_2 is corrupted can be proved similarly. ■

5 MinHash

5.1 Introduction

Min-hash sketch. The min-hash sketch is a simple and well-known technique to produce an unbiased estimate of the Jaccard index [33, 126]. The Jaccard index [111] is a similarity measure between two sets A and B, denoted $J(A, B)$, defined as the fraction of the elements in the intersection of A and B divided by the number of elements in their union. That is, $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$. The Jaccard index has seen wide application for clustering of websites and documents [33, 34], community identification [173], DNA matching [52], and machine learning [113, 183].

Computing the Jaccard index exactly, especially when the input sets are large, can be costly. The min-hash sketch allows communication-efficient approximation [33]. The basic idea behind the min-hash sketch is to apply a random hash function h to both sets A and B and then compare the minimum hashes (denoted $\min h(A)$, $\min h(B)$) in both sets. If $\min h(A) = \min h(B)$, it means that an element in $A \cap B$ has been hashed to the minimum value among elements in $A \cup B$. This occurs with probability $J(A, B)$. Thus, to get an unbiased approximation of the Jaccard index, it suffices to repeat this procedure with sufficiently many random hashes.

Private Jaccard index via min-hash. Due to its simplicity and efficiency, the min-hash sketch has become a popular tool to approximate the Jaccard index. Moreover, since the min-hash sketch only needs to compare the minimum hashes, it has been a key building block when maintaining privacy of the input sets is important, e.g., if the input sets represent fingerprints, DNA, or medical records.

There are two classes of solutions for privacy-preserving min-hash. The first class of solutions (e.g. [26, 52, 72, 156]) considers how to compute the min-hash and Jaccard index in a two-party setting, where the parties do not trust each other with their private inputs. The goal of these works is to design secure two-party computation protocols for computing the min-hash sketch as efficiently as possible, but they generally do not consider the privacy implications of revealing the output. The second line of work (e.g. [10, 184, 185]) considers how to make the min-hash approximation privacy-preserving by adding noise to the local min-hash sketches.

Our work. These works serve as the starting point for our study. In particular, we first present a protocol that addresses the privacy of min-hash-based approximations from two perspectives. Similar to the first class of solutions, our protocol ensures that no private information about the input is revealed beyond the Jaccard index. Additionally, in line with the second class of solutions, our protocol guarantees that even the Jaccard index output satisfies differential privacy, which is achieved through adding a small amount of noise. Next, we explore whether any variant of differential privacy can be achieved without adding noise to the protocol, which would improve its accuracy. Interestingly, we demonstrate that under specific constraints on the inputs, the resulting protocol still provides a certain level of privacy guarantees.

More formally, we define three ideal functionalities to capture flavors of min-hash. $\mathcal{F}_{\mathsf{minH}}$ computes the min-hash and then outputs *both the min-hash count and the random hashes used.* On the other hand, $\mathcal{F}_{\mathsf{privH}}$ computes the min-hash functionality and outputs *only the min-hash count.* This corresponds to a setting where the min-hash is computed by a trusted curator who does not disclose the hashes used. Finally, we define $\mathcal{F}_{\mathsf{noisy\text{-}minH}}$ which adds noise to the min-hash count computed by $\mathcal{F}_{\mathsf{minH}}$. For our first result we show that for appropriate noise levels, the $\mathcal{F}_{\mathsf{noisy\text{-}minH}}$ functionality achieves both high accuracy and differential privacy, and design a secure two-party computation of $\mathcal{F}_{\mathsf{noisy\text{-}minH}}$ that is both computation and communication-efficient. For our second result, we consider a setting in which the outputs of $\mathcal{F}_{\mathsf{minH}}$ or $\mathcal{F}_{\mathsf{privH}}$ have already been released *without added noise* and show that, under certain conditions on the inputs, this setting also provides privacy guarantees for individuals' inputs.

Differentially-private and secure computation of min-hash. To build a protocol for differentially-private min-hash we observe that the min-hash count has low global sensitivity. This allows us to define a functionality $\mathcal{F}_{\mathsf{noisy\text{-}minH}}$, parameterized by (ϵ, δ) which adds (properly-tuned) Laplace noise to the output of the min hash (See Figure 18 for details.) We then prove the following theorem about this functionality.

Theorem 5.1 (Informal)**.** $\mathcal{F}_{\mathsf{noisy\text{-}minH}}$ is (ϵ, δ)-DP against an adversary corrupting either party.

To realize a protocol for DP estimation of the Jaccard index, we now just need to instantiate this functionality. We show how this can be done efficiently using a PSI-CA functionality in Section 5.4. In Section 5.9, we evaluate the performance when instantiating the PSI-CA protocol [53, 172] in the semi-honest setting. The resulting protocol has better accuracy compared to the prior work [10, 101]. We recommend this protocol to compute differentially-private estimates of the Jaccard index.

Privacy after leakage of min-hash output. While ideally, parties should follow recommendations to add noise to the output of the min-hash count before releasing it (as in functionality $\mathcal{F}_{\textsf{noisy-minH}}$), in practice, this may not happen. Further, there may be historical counts that have already been released without added noise. We refer to settings in which such output is released as "output leakage."

We ask whether any privacy for an individual can be salvaged in this case. Somewhat surprisingly, we show that under certain conditions on the inputs to $\mathcal{F}_{\textsf{minH}}$ or $\mathcal{F}_{\textsf{privH}}$, the error of the min-hash approximation itself is sufficient to achieve (variants of) differential privacy–meaning that the presence of an individual element in one of the two input sets cannot be inferred given the output of $\mathcal{F}_{\textsf{minH}}$ or $\mathcal{F}_{\textsf{privH}}$. Essentially, the error of the sketch acts as noise to protect the privacy of the inputs. Similar observations that sketching algorithms inherently preserve privacy under certain input restrictions have previously been shown for the Johnson-Lindenstrauss sketch [24], the LogLog sketch [46, 165], and other sketches [179].

We first consider the simpler case of the privacy of an individual once the output of $\mathcal{F}_{\textsf{privH}}$ has been released. Recall that in this setting a set of private hashes is chosen by the functionality and these hashes are not returned as output of the functionality. Standard differential privacy in this setting requires that *conditioned on knowledge of A and all but one element of B (denoted by x^*)*, the probability that the functionality outputs any value out when $x^* \in B$ versus when $x^* \notin B$ differs by a factor of at most e^ϵ with all but negligible probability.

We note that min-hash is *not* differentially private in this setting if $A \cap B$ is either too large or too small. For example, if $|A \cap B| = 0$ when $x^* \notin B$ and 1 when $x^* \in B$, then min-hash always outputs 0 in the first case and outputs a count ≥ 1 with noticeable probability in the second. We prove the following theorem showing that when this is not the case the min-hash output is differentially private:

Theorem 5.2 (Informal)**.** If the size of the intersection is a constant fraction of the size of A and B, then the output of $\mathcal{F}_{\textsf{privH}}$ is (ϵ, δ)-DP for negligible δ.

We stress that this theorem crucially relies on the fact that the parties, and the adversary, do not have any information about the chosen hashes, and cannot learn the evaluation of the hashes on their own inputs. Note that for this theorem to be useful in a two-party protocol, the parties must compute the hashes under a 2-PC or FHE. This is unlikely to be done in practice. Thus, typically, the parties will locally store the hashes[9] during the computation. To understand the privacy of this approach, we consider the case of the $\mathcal{F}_{\textsf{minH}}$ functionality where the output leakage includes the hash functions as well as the counts.

Unfortunately, in this case there is a problem when trying to argue privacy. In the standard DP setting, we assume that the adversary knows all of the inputs (in this case, all entries in both sets A and B) except for some input x^* and wants to determine, from the output of the computation, whether x^* was in the other party's set. If the hashes are known, then the output of min-hash is

[9]As noted previously, it is sufficient to store a short seed to identify the hash.

deterministic: The adversary can exactly reconstruct the min-hash execution for the case when x^* is in the set and when it is not, and then see which of these matches the output it received. Since the min-hash protocol provides a good approximation of the Jaccard index, the adversary will be able to exactly determine whether or not $x^* \in B$ with noticeable probability.

Note that the above attack works only if the adversary knows the entirety of both sets A, B and just tries to distinguish whether $x^* \in B$ or not. Realistically, especially when the inputs are large, the adversary would not know the entire input of the honest party. More precisely, we assume that given the adversary's set (and even the intersection between the two sets), the honest party's set still has sufficiently high min-entropy. With this assumption, we turn to the tool of distributional DP (DDP) [14] which allows us to analyze differential privacy when the distribution of inputs has sufficient uncertainty.

We begin with a relatively strong assumption on the amount of uncertainty the adversary has about the honest set. Specifically, we assume that every element that is not in the intersection is highly unpredictable (i.e., has a high amount of min-entropy), even conditioned on all the other set elements. Under this assumption, we prove the following theorem:

Theorem 5.3 (Informal)**.** If each non-intersecting item has sufficiently high min-entropy, revealing the hash functions[10] together with the min-hash counts (as in the $\mathcal{F}_{\mathsf{minH}}$ functionality) preserves (ϵ, δ)-DDP for negligible δ, as long as the size of the intersection is a constant fraction of the size of A and B.

Not surprisingly, the proof of this Theorem (given in Section 5.6) leverages the fact that when each element has individual high min-entropy, hashing each element acts as a strong randomness extractor, thus resulting in sufficient random noise for privacy.

DDP over a polynomial-size universe. However, this assumption that every item has high min-entropy is quite strong. For example, consider the setting where each item in B is chosen from a polynomial-size universe. In this case, while individual items cannot have much min-entropy, the honest party's set may still collectively have high min-entropy as long as it is large enough. Thus, for our third result, we analyze what happens under this weaker assumption that *only the full honest set, instead of each individual item, has high min-entropy.*

Note that in this case, we cannot apply the hash function as randomness extractor technique. This is because in order to guarantee that the randomness extractor yields output that is negligibly close to uniform, we must lose super-logarithmic in n bits of entropy from each input. However, in the case we are currently considering, each element has at most $O(\log n)$ bits of min-entropy. Further, we in fact have no guarantee that each element has individually high min-entropy (since the elements are not necessarily independent), but only that the total min-entropy of the non-intersection items is high. Nevertheless, we show $\mathcal{F}_{\mathsf{minH}}$ still achieves DDP, by proving a new strong chain rule for min-entropy (see Section 5.8.5).

[10]We use cryptographic hash functions to instantiate the hashes in the random oracle model.

Specifically, we consider the following class of distributions $\mathcal{C}$ over secret sets R of size n:

- Let $\mathcal{U}$ be a universe of polynomial size $n \cdot \ell$, where $\ell = \Omega(n^3)$.
- R is chosen uniformly from all subsets of $\mathcal{U}$ of size n.
- In general, to relax the uniformity above, we additionally allow arbitrary leakage $L = L(R)$ computed on R, such that the length of the leakage L is at most $|L| \leq c \cdot n \log \ell$, for a fixed constant $c \in (0, 1)$.
- We consider the resulting conditional distribution $\mathcal{D}$ on R given leakage L.

Theorem 5.4 (Informal)**.** Assume the set R is drawn from a distribution $\mathcal{D} \in \mathcal{C}$. Then the min-hash protocol in the random oracle model (corresponding to functionality $\mathcal{F}_{\mathsf{minH}}$) preserves (ϵ, δ)-DDP for negligible δ, as long as the size of the intersection is a constant fraction of the size of A and B.

On spoiling bits and leakage resilience. Consider a distribution over sets of n elements $R = R_1, \ldots, R_n$, where each R_i is chosen from a universe of size $\ell \in \Omega(n)$. Note that the set R can have min-entropy $\Omega(n \lg(\ell))$ while it can still be possible that for every i, the marginal distribution over R_i has only *constant* min-entropy (see Example 1.1 in [67]). To deal with such situations, Skórski [164] proves a theorem showing the existence of "spoiling bits." Namely, given $R_1, \ldots, R_n$, some additional information known as spoiling bits can be released such that, conditioned on this information, for each $i \in [n]$, the distribution of R_i conditioned on $R_{<i}$, where $R_{<i}$ denotes $(R_1, \ldots, R_{i-1})$, is nearly flat (in the sense that the min/max entropy gap is at most a small additive constant). Further, the total number of spoiling bits that are released is small.

It is not hard to use Skórski's result to show that if R starts out with sufficiently high min-entropy then for a large fraction of i (those in the set $V \subseteq [n]$), the distribution of R_i conditioned on $R_{<i}$ has high min-entropy of at least $\Omega(\log(n))$, while the remaining indices (those in the set $W = [n] \setminus V$)) may have low min-entropy.

Unfortunately, this result is very brittle in the sense that the flatness conditions hold only for this particular distribution of R conditioned on the spoiled bits. Specifically, despite the flatness condition being satisfied for this distribution, the random variables R_i are *not* independent of one another. Thus, if additional information is leaked on R_j after the spoiling bits are computed, then the flatness guarantees may no longer hold for R_i.

In our setting, we require additional leakage $\{\ell_i\}_{i \in W}$ on the elements $\{R_i\}_{i \in W}$. One issue is that the set W (i.e., low min-entropy elements conditioned on the spoiling bits) is only known *after* the spoiling bits are computed. This leaves us with a dilemma:

- Leaking $\{\ell_i\}_{i \in W}$ additionally *after* the spoiling leakage can destroy the flatness property.

- On the other hand, we cannot leak $\{\ell_i\}_{i \in W}$ *before* computing the spoiling bits, since we don't know the set W yet! We could leak from all the blocks $(R_1, \ldots, R_n)$, but this may deplete the entropy needed from the random variables $\{R_i\}_{i \in V}$.

To solve this problem, we prove a new variant of the spoiling lemma that computes the spoiling bits *at the same time* as the additional leakage ℓ_i for $i \in W$ is computed so that the spoiling bits also contain $\{\ell_i\}_{i \in W}$, while still maintaining the flatness condition. The types of leakage that can be captured are essentially those such that the leakage ℓ_i for $i \in W$ can be expressed as a function of R_i and the leakages $\{\ell_j : j > i, j \in W\}$. It turns out that the leakage we need for our result has this form.

We state our theorem in general terms as we believe it may find further applications in leakage resilient cryptography. For the formal theorem statement see Theorem 5.14.

A note on composition. One known weakness of the DDP definition is the lack of a general composition theorem [14]. However, for the specific setting of our min-hash protocols we can leverage the small output of min-hash to argue composition properties after leakage of several outputs. Specifically, suppose that the adversary executes a min-hash protocol with (ϵ, δ)-DDP security twice with the same honest party's input both times. Since each min-hash protocol outputs a single number between 0 and k (i.e., $\lg k$ bits long), when we apply Theorem 3, the leakage profile increases to a total of at most $L + 2 \cdot \lg k$ bits. However, according to Theorem 3, as long as $|L| + 2 \lg k \leq c \cdot n \lg \ell$, each protocol execution will preserve DDP, and therefore the composition of the two protocol executions will preserve $(2\epsilon, 2\delta)$-DDP. In general, assuming that the initial leakage $|L|$ is a small constant, this type of DDP composition will hold for $O(n \cdot \frac{\lg \ell}{\lg k})$ executions.

Comparison to other approaches. We note that an alternative approach to get a differentially-private estimate of the Jaccard index is via mergeable cardinality estimation sketches (e.g. [101]) to compute (an approximation of) the set intersection cardinality and use this via the inclusion-exclusion principle to compute the Jaccard index. We give a detailed comparison of error from our protocol vs. the best known cardinality estimator [101] in Section 5.9.

5.2 Related Works

Differential privacy (DP). Differential privacy protects the privacy of individuals by limiting an adversary's ability to learn information about an individual input from the output of a computation [60, 63]. For a good overview of differential privacy and many of the algorithms to achieve it, both in the standard curator setting and in distributed settings, we refer the reader to the book by Dwork and Roth [66].

Optimizing secure computation using differential privacy. Another direction of work has considered how to use DP to reduce the cost of secure computation, especially when we aim for DP-style guarantees from the final output. [17] first proposed such optimization for the problem of secure summation. [100] and [94] applied the differential privacy relaxation to improve efficiency of set-intersection protocols. [133], and [134] consider graph-parallel computations and design more efficient solutions with differential private leakages. [40] consider classic tasks like sorting, merging, and range-query data structures with differential privacy relaxation. [92] consider multiparty shuffle that allows a differentially private leakage and shows that it suffices to achieve end-to-end differential privacy in the shuffle model of DP.

Private sketching. Sketching algorithms, or "sketches" are sublinear space algorithms for approximating certain properties of large inputs or data streams. The main idea behind sketching algorithms is to generate a compact summary data structure that allows for efficient storage, merging, and processing.

Some recent works [15,16,24,46,56,101,104,127,137,138,145,165,179,189] have additionally observed that sketches can often also aid in achieving privacy as the inherent loss of information in the sketch can essentially make the sketch itself be differentially private or to only require a little additional noise.

A line of research pertinent to our work involves constructing private sketches for set cardinality estimations [124,142,144,168,169]. Recently, [101] proposed a private mergeable sketch that can be used to estimate the size of the intersection and union of sets.

Secure approximation. Secure approximation studies what functions can be securely approximated without revealing anything beyond the true output [76, 99]. While this notion is quite different from that of differentially private approximation that we consider here, we note that our FHE-based protocol described in Section 5.5 additionally achieves this.

Adversarially robust property-preserving hash functions and robust sketching. Property-preserving hash (PPH) functions allow compressing large input x into a short digest $h(x)$ such that some property $P(x, y)$ can be computed given only $h(x)$ and $h(y)$. Adversarially-robust PPH [30,80,81,103] aim to further guarantee that $P(x, y)$ is correctly computed (i.e., robust) even if the inputs x and y are chosen after the hash function h is fixed. A related concept of robust sketching, e.g. [9,20] aims to construct sketches that provide good approximations even when inputs are chosen after the randomness of the sketch is fixed.

Both of these approaches are similar to our work in that they also study the consequences of making the choice of hash (or sketch) known to the adversary. However, these works focus on robustness to adversarial inputs, while we instead focus on the privacy of the output when the adversary additionally sees the hash functions.

Differentially private min-hash DP min-hash aims to make min-hash approximation differentially private by adopting standard DP mechanisms such as adding DP noises to the output to hide individual items in the input sets. In particular, [10] achieves local DP (LDP) min-hash by either adding Laplacian noise, or using generalized randomized response to perturb the minhash vectors. Other than this, there are also other earlier efforts. For example, [185] attempts to use a flawed exponential mechanism to achieve DP. This leads to a faulty claim of ϵ-DP, as pointed out in [10]. [184] correctly applies exponential mechanism. However, this results in a large amount of noise being added to the results.

5.3 Preliminaries

A function g is *negligible*, denoted $\mathsf{negl}(\cdot)$, if for every positive integer c, there is an integer n_c such that for all $n \geq n_c$ we have $g(n) \leq 1/n^c$. Let κ denote the security parameter.

Range of hash functions and the random oracle model. We model each hash function as a random oracle that maps each item to a real value in $[0, 1]$, and the output of the hash function is long enough to ensure that the probability of any two different items having a hash collision is negligible.

Notation. Let $\mathcal{U}$ denote the universe of input elements. In this paper, we will consider two input sets $A, B \subseteq \mathcal{U}$. Let $n_A = |A|, n_B = |B|$. Let $I = A \cap B$, $n_I = |I|$. We will also let $B_{+x^*} = B \cup \{x^*\}$.

Let Eq be an equality function; i.e., $\mathsf{Eq}(a, b) = 1$ if $a = b$ and 0 otherwise. For a hash function h and a set A, we let $h(A) := \{h(a) : a \in A\}$. Let $\mathrm{B}(m, p)$ be the binomial distribution with m trials and each trial having success probability p.

Basic min-hash functionality. We describe the basic min-hash functionality in Figure 17. In this work, we will consider several variants and consider privacy implications.

Protocol 17 The Basic Min-Hash Functionality $\mathcal{F}_{\mathsf{minH}}$

The functionality is parameterized with a random oracle $\mathcal{O}$.

Input: P_1 and P_2's input vectors $A = (x_1^A, \ldots, x_{n_A}^A)$ and $B = (x_1^B, \ldots, x_{n_B}^B)$.

Minhash:

1. Randomly sample prefix pre, which is used to define hash functions $h_1, h_2, \ldots, h_k$, where for $i \in [k]$, $h_i(\cdot) := \mathcal{O}(\mathsf{pre}||i||\cdot)$.
2. For input A, compute the min-hash vector $(u_1^A, u_2^A, \ldots, u_k^A)$ as follows:

   For each iteration $j \in [k]$:

   i. For each item $x_i^A \in A$, compute $y_{i,j}^A = h_j(x_i^A)$.

   ii. Compute the min-hash for iteration j; that is, $u_j^A = \min\{y_{i,j}^A : i \in [n_A]\}$.
3. Likewise, compute another min-hash vector $(u_1^B, u_2^B, \ldots, u_k^B)$ for input B similarly.
4. Compute $c = \sum_{j=1}^{k} \mathsf{Eq}(u_j^A, u_j^B)$.

Output: Return (pre, c) to P_1 and P_2.

Differential privacy. The definitions of (ϵ, δ)-differential privacy are given below.

Definition 5.5 ((ϵ, δ)-indistinguishablity)**.** Two random variables X and Y are (ϵ, δ)-indistinguishable (denoted as $X \approx_{\epsilon,\delta} Y$) if, for all events S, we have

$$\Pr[X \in S] \leq e^{\epsilon} \cdot \Pr[Y \in S] + \delta, \quad \Pr[Y \in S] \leq e^{\epsilon} \cdot \Pr[X \in S] + \delta.$$

Definition 5.6 (Computational (ϵ, δ)-indistinguishablity)**.** Two random variables X and Y are computationally (ϵ, δ)-indistinguishable (denoted as $X \overset{c}{\approx}_{\epsilon,\delta} Y$) if, for any polynomial time adversary $\mathcal{A}$, it holds

$$\Pr[\mathcal{A}(X) = 1] \leq e^{\epsilon} \cdot \Pr[\mathcal{A}(Y) = 1] + \delta, \quad \Pr[\mathcal{A}(Y) = 1] \leq e^{\epsilon} \cdot \Pr[\mathcal{A}(X) = 1] + \delta.$$

Definition 5.7 ((Computational) (ϵ, δ)-differential privacy)**.** Let X be an input space and $\simeq_X$ be a relation capturing the notion of neighboring inputs. Let $\mathcal{M} : X \to Z$ be a randomized algorithm that takes input $x \in X$ and outputs a value over Z. We say that the mechanism $\mathcal{M}$ is (ϵ, δ)-differentially private if the following holds:

$$\forall x, x' \in X \text{ s.t. } x \simeq_X x' : \ \mathcal{M}(x) \approx_{\epsilon,\delta} \mathcal{M}(x').$$

The mechanism $\mathcal{M}$ is (ϵ, δ)-computationally differentially private if $\forall x, x' \in X$ s.t. $x \simeq_X x' : \ \mathcal{M}(x) \overset{c}{\approx}_{\epsilon,\delta} \mathcal{M}(x')$.

Definition 5.8. The global sensitivity of a function $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$ is:

$$\Delta f = \max_{X,Y \in \mathbb{N}^{|\mathcal{X}|}, \|X-Y\|_1 = 1} \|f(X) - f(Y)\|_1$$

Definition 5.9. The Laplace Distribution (centered at 0) with scale b is the distribution with probability density function: $Lap(x|b) = \frac{1}{2b} e^{-|x|/b}$.

We will write $Lap(b)$ to denote the Laplace distribution with scale b. Given any function $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$, the Laplace mechanism that adds noise drawn from Laplace distribution; that is, given an input database X, the mechanism outputs $f(X) + (Y_1, \ldots, Y_k)$, where Y_i are i.i.d. random variables drawn from $Lap(\Delta f/\epsilon)$. It is known that the Laplace mechanism achieves $(\epsilon, 0)$-differential privacy [66, Theorem 3.6].

Distributional differential privacy (DDP). We adapt the original definition [14] for our purpose to consider a two-party protocol that takes sets as input more explicitly. Specifically, we consider a computational indistinguishability variant for our DDP definition.

Definition 5.10 (View of a party in a two-party functionality)**.** Given a two-party functionality $\mathcal{F}$ with parties P_1 and P_2, let $\mathsf{view}_{P_1}^{\mathcal{F}}(A, B)$ denote the view of P_1 for the execution of functionality $\mathcal{F}$ with A and B being the input of P_1 and P_2 respectively. In particular, $\mathsf{view}_{P_1}^{\mathcal{F}}(A, B)$ consists of the following (the view of P_2 is defined similarly):

- The input A of P_1, the private random coins of P_1, and the output of the functionality.
- If the functionality is in the random oracle model, we allow a semi-honest P_1 to make a polynomial number of arbitrary queries to the random oracle and to add the input/output information to its view.

Definition 5.11 (DP and DDP of a two-party functionality)**.** A two party functionality $\mathcal{F}$ is (computationally) (ϵ, δ)**-DP** against an adversary corrupting P_1, if for every (A, B) and every $x^* \in \mathcal{U}$, it holds that $\mathsf{view}_{P_1}^{\mathcal{F}}(A, B)$ is (compuatationally) (ϵ, δ)-indistinguishable from $\mathsf{view}_{P_1}^{\mathcal{F}}(A, B_{+x^*})$.

Let $\mathcal{X}$ denote a random variable for two sets over universe $\mathcal{U}$. Let $\mathcal{Z}$ denote the random variable measuring the additional auxiliary information known to the adversary. A two party functionality $\mathcal{F}$ is (computationally) $(\epsilon, \delta, \Delta)$**-DDP** against an adversary corrupting P_1, if for every distribution $\mathcal{D} \in \Delta$ on $(\mathcal{X}, \mathcal{Z})$, every $(X = (A, B), Z)$ in the support of $(\mathcal{X}, \mathcal{Z})$ and every $x^* \in \mathcal{U}$, it holds that $\big(\mathsf{view}_{P_1}^{\mathcal{F}}(A, B), Z\big)$ is (computationally) (ϵ, δ)-indistinguishable from $\big(\mathsf{view}_{P_1}^{\mathcal{F}}(A, B_{+x^*}), Z\big)$. Here, (A, B) and Z are sampled from $\mathcal{D}$, and each party may use additional randomness.

DP and DDP against an adversary corrupting P_2 is defined symmetrically.

Tail bound for a Binomial distribution. We will use this well-known inequality.

Lemma 5.12 ([59])**.** Consider a Binomial distribution $B(n, p)$. We have

$$\Pr_{X \sim B(n,p)}[X \geq k] \leq \binom{n}{k} p^k.$$

5.4 Min-Hash with DP

Since the noiseless min-hash functionality cannot achieve DP as discussed above, we consider a noisy variant that provides DP. We first consider the global sensitivity of $\mathcal{F}_{\mathsf{minH}}$ and use the standard Laplace mechanism to provide DP.

5.4.1 Sensitivity

Let $B = (x_1^B, \ldots, x_{n_B}^B)$ and $B_{+x^*} = (x_1^B, \ldots, x_{n_B}^B, x^*)$, and WLOG, we consider two neighboring inputs (A, B) and (A, B_{+x^*}); the case in which x^* is added into A can be shown symmetrically.

We show how changing the input sets from B to B_{+x^*} affects the final count. Let x^* be the $(n_B + 1)$-th element of B_{+x^*}. Consider iteration j of Step 2 in Figure 17. Since we model each hash function h_j as a random oracle, $(y_{1,j}^B, \ldots, y_{n_B+1,j}^B)$ will be uniformly distributed. Now, consider how the min-hash u_j^B is computed. *The value x^* from B_{+x^*} can affect the min-hash u_j^B (and thereby the final count c), only if $y_{n_B+1,j}^B$ is smaller than $(y_{1,j}^B, \ldots, y_{n_B,j}^B)$.*

The probability that $y^B_{n_B+1,j}$ will be less than all $y^B_{i,j}$s is at most $1/(n_B+1)$ by a symmetry argument. Note the final output is computed as the sum of k of these trials. Let

$$S_{x^*} = \left\{ j \in [k] : y^B_{n_B+1,j} < \min_{i \in [n_B]} \{y^B_{i,j}\} \right\}.$$

Therefore, we consider a binomial distribution $|S_{x^*}| \sim \mathrm{B}(k, 1/(n_B+1))$, which represents how many iterations j cause x^* to be the min-hash u^B_j. In other words, $|S_{x^*}|$ captures the sensitivity of min-hash. Therefore, given the failure probability δ, the following measure can be used as the global sensitivity:

$$\sigma(\delta, k, n_B) := \arg\min_s \ \{s : \Pr_{h_1,\ldots,h_k}[|S_x| \geq s] \leq \delta\}$$

Lemma 5.13. For any $\{x^B_i\}_{i \in [n_B]}$ and $x \in \mathcal{U}$, we have $\sigma(\delta, k, n_B) \leq \binom{k}{s} \cdot \left(\frac{1}{n_B+1}\right)^s$.

Proof. The result immediately follows from Lemma 5.12. ■

According to the above lemma, Asymptotically, with $k = \Omega(\kappa)$, we have $\sigma(\delta = \mathsf{negl}(\kappa), k, n = \Theta(k^2)) = O(\lg \lg k)$.

5.4.2 Noisy Min-Hash

We consider a variant $\mathcal{F}_{\mathsf{noisy\text{-}minH}}$ of $\mathcal{F}_{\mathsf{minH}}$ described in Figure 18.

Protocol 18 Noisy Min-Hash Functionality $\mathcal{F}_{\mathsf{noisy\text{-}minH}}$

1. Run $\mathcal{F}_{\mathsf{minH}}$ and let (pre, c) be the output from $\mathcal{F}_{\mathsf{minH}}$.

2. Sample ζ^A and ζ^B from Laplace distributions $Lap(\frac{\sigma(\delta/2,k,n_A)}{\epsilon})$ and $Lap(\frac{\sigma(\delta/2,k,n_B)}{\epsilon})$ respectively. Let coins_A and coins_B the random coins that the functionality used to sample them.

3. Let ℓ_A and ℓ_B be the minimum integers satisfying $\Pr[|\zeta^A| > \ell_A] \leq \delta/2$ and $\Pr[|\zeta^B| > \ell_B] \leq \delta/2$. If $|\zeta_A| > \ell_A$, then truncate it so that it holds $|\zeta_A| = \ell_A$. Likewise, truncate ζ_B if necessary.

4. Let $\mathsf{r}(\cdot)$ be a rounding function. Output $(\mathsf{pre}, \mathsf{coins}_A, \mathsf{r}(c + \zeta^B))$ to P_1 and $(\mathsf{pre}, \mathsf{coins}_B, \mathsf{r}(c + \zeta^A))$ to P_2.

Theorem 5.1. $\mathcal{F}_{\mathsf{noisy\text{-}minH}}$ is (ϵ, δ)-DP against an adversary corrupting either party.

Proof. Based on the definition, $\sigma(\cdot)$ works as the upperbound on the sensitivity with probability $1-\delta/2$. For the honest party's noise (i.e., ζ_A or ζ_B), truncation takes place with probability at most $\delta/2$. Therefore, using the standard Laplace mechanism [66, Theorem 3.6], and since DP is preserved even with post-processing, $\mathcal{F}_{\textsf{noisy-minH}}$ provides (ϵ, δ)-DP. ∎

A two party protocol $\pi_{\textsf{NMH}}$ securely realizing $\mathcal{F}_{\textsf{noisy-minH}}$. We construct a two party protocol that securely realizes functionality $\mathcal{F}_{\textsf{noisy-minH}}$. The protocol takes advantage of an ideal functionality $\mathcal{F}_{\textsf{psi-ca}}$ of private set intersection cardinality (PSI-CA) [53] that computes the exact cardinality of the intersection of the two input sets, as described in Figure 19.

Protocol 19 Functionality of Private Set Intersection Cardinality $\mathcal{F}_{\textsf{psi-ca}}$

Input: P_1 has a set A and P_2 has a set B.
Output: Return $|A \cap B|$ to P_1 and $\perp$ to P_2.

In particular, in order to compute the noisy min-hash match counts, the parties construct two sets consisting min-hash values and additional dummy elements and then run $\mathcal{F}_{\textsf{psi-ca}}$ on these sets. To reflect the Laplace noise into elements of a set, the protocol uses unary encoding, which introduces some inefficiency. However, as the tail probability of Laplace noise decreases exponentially, the unary encoding length can be bounded with a small value, and the protocol's overall efficiency is still maintained. Detailed steps of the protocol are provided in Figure 20.

It is worth noting that the above task could also be implemented using a generic two-party computation (2PC) protocol. However, [53] proposed an efficient PSI-CA protocol that outperforms 2PC protocols for small input sizes (using the start-to-finish comparison including the 2PC preprocessing steps). See Section 5.9.1 for more details of this PSI-CA protocol. Since our input sets are small, we chose to present protocol $\pi_{\textsf{NMH}}$ using the PSI-CA functionality.

We will prove below that protocol $\pi_{\textsf{NMH}}$ securely realizes $\mathcal{F}_{\textsf{noisy-minH}}$. It implies that protocol $\pi_{\textsf{NMH}}$ is also (ϵ, δ)-computational-DP [163]. The main benefit of the protocol is that the hash computations can be computed locally and the communication complexity of the protocol is *sub-linear in n_A and n_B* even when the protocol implementing $\mathcal{F}_{\textsf{psi-ca}}$ has a linear communication complexity.

Proposition 5.2. Protocol $\pi^{\mathcal{O}}_{\textsf{NMH}}$ described in Figure 20 securely realizes $\mathcal{F}_{\textsf{noisy-minH}}$ in the semi-honest model.

Proof. First note that the protocol will correctly compute $c_A = \mathsf{r}(c + \zeta^B)$ and $c_B = \mathsf{r}(c + \zeta^A)$ as in $\mathcal{F}_{\textsf{noisy-minH}}$. For privacy, when P_1 is corrupted, the only message to simulate is c_A, the output from $\mathcal{F}_{\textsf{psi-ca}}$. Since the protocol is in the $\mathcal{F}_{\textsf{psi-ca}}$ hybrid, this message c^A can be perfectly simulated by using the output from $\mathcal{F}_{\textsf{noisy-minH}}$. The simulator can also make sure that pre and ζ^A are correctly sampled by using pre and coins_A from $\mathcal{F}_{\textsf{noisy-minH}}$. For corrupted P_2, first the simulator makes sure that pre and ζ^B are correct by using pre and coins_B from

Protocol 20 **Two-party Noisy Min-hash Protocol** $\pi^{\mathcal{O}}_{\mathsf{NMH}}$

Input: P_1 and P_2's input vectors $A = (x^A_1, \ldots, x^A_{n_A})$ and $B = (x^B_1, \ldots, x^B_{n_B})$.

Protocol:

1. P_1 samples prefix pre and sends it to P_2. This prefix is used to define hash functions $h_1, h_2, \ldots, h_k$, where for $i \in [k]$, $h_i(\cdot) := \mathcal{O}(\mathsf{pre}||i||\cdot)$.

2. P_1 computes the min-hash vector $(u^A_1, u^A_2, \ldots, u^A_k)$ locally exactly as described in $\mathcal{F}_{\mathsf{minH}}$. Likewise, P_2 locally computes $(u^B_1, u^B_2, \ldots, u^B_k)$.

3. P_1 (resp. P_2) samples ζ^A (resp. ζ^B) from Laplace distribution $Lap(\frac{\sigma(\delta/2,k,n_A)}{\epsilon})$ (resp. $Lap(\frac{\sigma(\delta/2,k,n_B)}{\epsilon})$). Let coins_A (resp. coins_B) be the random coins that P_1 (resp. P_2) used in sampling the noise ζ^A (resp. ζ^B). As in $\mathcal{F}_{\mathsf{noisy\text{-}minH}}$, parties truncate ζ^A and ζ_B based on ℓ_A and ℓ_B, if necessary.

   Let Z^B be a $2\ell_B$-bit vector representing the unary encoding of $\mathsf{r}(\zeta^B + \ell_B)$. That is, the first $\mathsf{r}(\zeta^B + \ell_B)$ bits are 1's and the remaining bits are 0's. We let Z^B_j denote the jth bit of Z^B.

4. P_1 and P_2 invokes $\mathcal{F}_{\mathsf{psi\text{-}ca}}$ with the following inputs:

   - P_1's input: $\{(i, u^A_i) : i \in [k]\} \cup \{(j + k, 1) : j \in [2\ell_B]\}$
   - P_2's input: $\{(i, u^B_i) : i \in [k]\} \cup \{(j + k, Z^B_j) : j \in [2\ell_B]\}$

   Let *out* be the output to P_1 from functionality $\mathcal{F}_{\mathsf{psi\text{-}ca}}$. Set $c_A = out - \ell_B$.

5. P_1 computes $c^+ = c^A + \mathsf{r}(\zeta^A)$ and sends c^+ to P_2. P_2 computes $c^B = c^+ - \mathsf{r}(\zeta^B)$.

Output: P_1 and P_2 output $(\mathsf{pre}, \mathsf{coins}_A, c^A)$ and $(\mathsf{pre}, \mathsf{coins}_B, c^B)$ respectively.

$\mathcal{F}_{\mathsf{noisy\text{-}minH}}$. The message $c^+ = c^B + \mathsf{r}(\zeta^B)$ can also be perfectly simulated, since the simulator can obtain c^B from $\mathcal{F}_{\mathsf{noisy\text{-}minH}}$. ■

5.5 Noiseless Protocol in the Private Hash Setting

In Figure 21, we describe the min-hash protocol $\mathcal{F}_{\mathsf{privH}}$ in the private hash setting. We show that if $J(A, B)$ is a constant, there exist parameter regimes where $\mathcal{F}_{\mathsf{privH}}$ without noise satisfies differential privacy. Our observation is that the final count c follows a binomial distribution in the private hash setting, which can be treated as noise to obscure the sensitivity.

Theorem 5.3. For any constant $\epsilon > 0$, if $k = k(\epsilon, \kappa) \in \Omega(\kappa), n_A/k \in \Omega(\kappa), n_B/k \in \Omega(\kappa)$, and $J(A, B) \in (0, 1)$ is a constant independent of κ, then $\mathcal{F}_{\mathsf{privH}}$ is (ϵ, δ)-DP with $\delta \in \mathsf{negl}(\kappa)$.

Protocol 21 **Min-Hash in the Private Hash Setting** $\mathcal{F}_{\mathsf{privH}}$

$\mathcal{F}_{\mathsf{privH}}$ works exactly the same as $\mathcal{F}_{\mathsf{minH}}$ except that it outputs only the final count c (with the prefix pre hidden to the participants).

Proof. WLOG, let $B = (x_1^B, \ldots, x_{n_B}^B)$ and $B_{+x^*} = (x_1^B, \ldots, x_{n_B}^B, x^*)$. Let $p = J(A, B)$, $s = \sigma(\delta, k, \min(n_A, n_B)) = O(\lg\lg\kappa)$. Recall the definition S_{x^*} in Section 5.4.1 and let $K_{x^*} = [k] \setminus S_{x^*}$. Note that for the iterations in K_{x^*}, the min-hash matches (denoted as $c_{K_{x^*}}$) for both (A, B) and (A, B_{+x^*}) will be identically distributed. This match count c_{x^*} will work as an additive noise. Since $h_1, \ldots, h_k$ are private, we have $c_{K_{x^*}} \sim \mathrm{B}(k - s, p)$. By applying Lemma 5.4 below, we conclude that $\mathcal{F}_{\mathsf{privH}}$ is differentially private. ■

Lemma 5.4. Consider a Binomial distribution $\mathrm{B}(n, p)$, where $n \in \Omega(\kappa)$ and $p \in (0, 1)$ is a constant independent of κ. Then, for any constant ϵ and $s = O(\lg\lg\kappa)$, there are $a, b \in [n]$ with $a < np < b$ such that

- For any $\ell \in [a, b]$, $e^{-\epsilon} \leq \frac{\Pr_{X\sim\mathrm{B}(n,p)}[X=\ell]}{\Pr_{X\sim\mathrm{B}(n,p)}[X+s=\ell]} \leq e^{\epsilon}$.
- For any $\ell \notin [a, b]$, $\Pr[\mathrm{B}(n, p) = \ell] = \mathsf{negl}(\kappa)$ and $\Pr[\mathrm{B}(n, p) + s = \ell] = \mathsf{negl}(\kappa)$.

The proof of the lemma is found in Section 7.3.1.

Remark. While $\mathcal{F}_{\mathsf{privH}}$ could be considered as a trusted curator model, a two-party protocol realizing it can be constructed without relying on a trusted curator. In particular, the computation of $(u_1^A, \ldots, u_k^A)$ (including all n hash evaluations) can be performed locally under a threshold FHE so that only the encryption of them may be sent to party B. Then, by computing the remaining steps under FHE and delivering the result using a threshold decryption, the protocol will securely realize $\mathcal{F}_{\mathsf{privH}}$ in the semi-honest setting. We note that the resulting protocol has sublinear communication in n since only the k inputs to the comparisons need to be communicated.

5.6 DDP of $\mathcal{F}_{\mathsf{minH}}$

In this section, we show that there are parameter regimes where the public min-hash protocol $\mathcal{F}_{\mathsf{minH}}$ can satisfy DDP without adding noise. In Figure 22, we first describe the family of distributions we consider in the context of our min-hash protocol. The distribution models a situation in which the adversary, having corrupted one of the two parties, has access to the view of the party and even the actual intersection. However, the adversary does not know the other party's input set (except from the intersection).

Below, we show that $\mathcal{F}_{\mathsf{minH}}$ achieves DDP under certain circumstances.

Theorem 5.5. For every constant $\epsilon > 0$, consider $\mathcal{F}_{\mathsf{minH}}$ in the random oracle model with $k = k(\epsilon, \kappa)$, where $k \in \Omega(\kappa)$. Let $R = B \setminus I$, each element of which has min-entropy at least κ. Let $n_A/k, n_B/k \in \Omega(\kappa)$, and $n_I/n_A \in (0, 1)$ is a

Protocol 22 Distribution Family Δ_{PH}

Parameterized with (n_A, n_B, n_I), a distribution $\mathcal{D}_{A,B}$ in this family samples (A, B) such that

- Letting $I = A \cap B$, it holds that $|A| = n_A$, $|B| = n_B$, and $|I| = n_I$

Output:

- The inputs to the parties P_1 and P_2 are A and B respectively.
- Give I to the adversary as the auxiliary information.

constant independent of κ. Then, $\mathcal{F}_{\mathsf{minH}}$ is computationally $(\epsilon, \delta, \Delta_{\mathsf{PH}})$-DDP against an adversary corrupting P_1 with $\delta \in \mathsf{negl}(\kappa)$. DDP against an adversary corrupting P_2 holds when the parameters are set symmetrically.

Theorem 5.6. For security parameter κ, every constant $\epsilon > 0$, and every constant $\gamma \in (0, 1)$, consider $\mathcal{F}_{\mathsf{minH}}$ in the random oracle model with $k = k(\epsilon, \kappa)$, where $k \in \Omega(\kappa \cdot \lg \lg \kappa)$. Let $R = B \setminus I$ be a set of size n_R, with $n_R/k^2 \in \Omega(\kappa)$. Let the universe $\mathcal{U}$ be of size $n_R \cdot \ell$, where $\ell = \Omega(n_R^3)$. Assume the secret set R is chosen chosen uniformly from all subsets of $\mathcal{U}$ of size n_R, conditioned on arbitrary leakage on R of length L, where $n_R \lg \ell - L \geq \frac{8n_R}{9} \lg \ell + 2n_R$. Let $|I| \in \Theta(n)$. Then the output of $\mathcal{F}_{\mathsf{minH}}$ achieves computational $(\epsilon, \delta, \Delta_{\mathsf{PH}})$-DDP with $\delta \in \mathsf{negl}(\kappa)$ against an adversary corrupting P_1. DDP against an adversary corrupting P_2 holds when the parameters are set symmetrically.

Remark. An easy way to realize $\mathcal{F}_{\mathsf{minH}}$ is to have each party locally hash their inputs using the k public hash functions and to locally compute the minimum for each iteration. The parties can then run a simple two-party computation to compute the number of times these minimums match. We note that this protocol has communication and computation that is sublinear in the input size as it only depends on the number of hash functions. By Theorems 5.5 and 5.6 this protocol achieves DDP when the conditions of either of the theorems are satisfied.

5.7 Proof of Theorem 5.5

We first give the intuition of the proof. We assume that each of the non-intersecting elements has high min-entropy. WLOG, consider an adversary corrupting P_1. The view of the adversary will be

$$\mathsf{view}_{P_1}^{\mathcal{F}_{\mathsf{minH}}}(A, B) := (A, c, h_1, \ldots, h_k).$$

As shown above, the sensitivity can be upper-bounded by a small value s.

Unlike $\mathcal{F}_{\mathsf{privH}}$, however, when we show the existence of sufficient noise from the remaining iterations, we need to take the additional leakage into consideration.

First, since the hash functions are public, iterations are no longer independent of each other as needed by the analysis in Section 5.5. We address this issue by employing the fact that each of the non-intersecting items has high min-entropy. In the random oracle model, as long as the adversary does not query hash function h on some point x, $h(x)$ is uniformly random to the adversary. Since the non-intersecting items have high min-entropy, the adversary is negligibly likely to query any of them to the hash functions, thus guaranteeing independence.

Good iterations and Poisson Binomial distribution. Now, to see how the remaining iterations still hide the sensitivity even with the public hash functions, let $R = B \setminus I$. For the remaining $k - s$ iterations, the high min-entropy of each element in R will jitter the final count. In particular, consider the jth hash function h_j in the protocol (among the $k - s$ remaining iterations) and let

$$v_j^A = \min h_j(A), \quad v_j^I = \min h_j(I), \quad v_j^R = \min h_j(R).$$

Suppose $v_j^A = v_j^I$. Then, if $v_j^R \geq v_j^I$, the min-hash u_j^A of A will be equal to the min-hash u_j^B of B (both of which are equal to v_j^I) and the final count c will be incremented due to this jth iteration. However, if $v_j^R < v_j^I$, then it will be $u_j^A \neq u_j^B$, and the final count will not be incremented. This way, the distribution of v_j^R will jitter the final count. The above discussion can be formalized into the following definition.

Definition 5.7 (θ-good iteration)**.** Let $n_R = n_B - n_I$, we define $\mathsf{good}_\theta(h_j, A, I, n_B)$ to be true if and only if the following holds:

$$\min h_j(A) = \min h_j(I), \text{ and } \min h_j(I) \in \left[1 - \left(\frac{1}{2} + \theta\right)^{1/n_R}, 1 - \left(\frac{1}{2} - \theta\right)^{1/n_R}\right].$$

The second condition of the definition requires that $\min h_j(I)$ is somewhere in the middle (parameterized by $\theta \in \Theta(1)$) so that the distribution of R (i.e., random v_j^R) may reduce the final count with a decent chance (and also keep the count with a decent chance). As long as n_I/n_A is a constant fraction, there are sufficiently many θ-good iterations, although we lose some iterations. In particular, if we let k_g be the number of good iterations, we have $k_g = \Theta(k)$.

With public hash functions and thereby $\min h_j(I)$ being leaked to the adversary, it turns out that the noise from the k_g iterations follows a Poisson Binomial distribution, which is a generalization of a Binomial distribution where each trial has a different success probability. However, using the techniques of [42], we can still show that this distribution works as a good noise to hide the private data.

5.7.1 Proof

WLOG, we consider two neighboring inputs (A, B) and (A, B_{+x^*}). DDP for the case in which x^* is added into A can be shown symmetrically. We prove the theorem by a hybrid argument. In particular, we define a slightly different ideal functionality $\mathcal{F}_{\mathsf{minH}}{}^{(1)}$ as follows:

- Let ${\mathcal{F}_{\mathsf{minH}}}^{(1)}$ be the same as $\mathcal{F}_{\mathsf{minH}}$ except that for each $x_i^B \in B \setminus A$, each element in $\{y_{i,j}^B\}_j$ is chosen uniformly at random from $[0, 1]$.

We set up the following hybrids. We will argue that for any $x^* \in \mathcal{U}$ and over $(A, B, I) \leftarrow \Delta_{\mathsf{PH}}$, it holds

$$\begin{aligned}(\mathsf{view}_{P_1}^{\mathcal{F}_{\mathsf{minH}}}(A, B), I) &\overset{c}{\approx} (\mathsf{view}_{P_1}^{{\mathcal{F}_{\mathsf{minH}}}^{(1)}}(A, B), I)\\ &\approx_{\epsilon,\delta} (\mathsf{view}_{P_1}^{{\mathcal{F}_{\mathsf{minH}}}^{(1)}}(A, B_{+x^*}), I) \overset{c}{\approx} (\mathsf{view}_{P_1}^{\mathcal{F}_{\mathsf{minH}}}(A, B_{+x^*}), I)\end{aligned}$$

for any constant $\epsilon > 0$ and for some $\delta \in \mathsf{negl}(\kappa)$, as long as each element in $B \setminus I$ has high min-entropy.

Recall that the min-entropy of each element x_i^B with $i \in B \setminus A$ is at least κ. Therefore, the probability that any adversary making at most polynomially many oracle queries queries any x_i^B is $\mathsf{negl}(\kappa)$. Conditioned on the adversary not querying any such x_i^B, any $y_{i,j}^B$ for $j \in [k]$ is chosen uniformly random from $\mathcal{U}$. The same argument shows $\mathsf{view}_{P_1}^{\mathcal{F}_{\mathsf{minH}}}(A, B_{+x^*}), I) \overset{c}{\approx} (\mathsf{view}_{P_1}^{{\mathcal{F}_{\mathsf{minH}}}^{(1)}}(A, B_{+x^*}), I)$. Therefore, we are left only to show $(\mathsf{view}_{P_1}^{{\mathcal{F}_{\mathsf{minH}}}^{(1)}}(A, B), I) \approx_{\epsilon,\delta} (\mathsf{view}_{P_1}^{{\mathcal{F}_{\mathsf{minH}}}^{(1)}}(A, B_{+x^*}), I)$.

DDP of ${\mathcal{F}_{\mathsf{minH}}}^{(1)}$. We show $(A, I, h_1, \ldots, h_k, c) \approx_{\epsilon,\delta} (A, I, h_1, \ldots, h_k, c_{+x^*})$, where c is the final count from ${\mathcal{F}_{\mathsf{minH}}}^{(1)}(A, B)$ and c_{+x^*} is the final count from ${\mathcal{F}_{\mathsf{minH}}}^{(1)}(A, B_{+x^*})$. We show how to leverage the uncertainties of $x_i^B \in R = B \setminus A$ so that good iterations work like the needed noise to guarantee DP.

Lemma 5.8. For any A, I, n_B and $n_R = n_B - n_I$, we have

$$p_\theta \overset{def}{=} \Pr_h[\mathsf{good}_\theta(h, A, I, n_B)] \geq \left(\left(\frac{1}{2} + \theta\right)^{\frac{n_A}{n_R}} - \left(\frac{1}{2} - \theta\right)^{\frac{n_A}{n_R}}\right) \cdot \frac{n_I}{n_A} \quad .$$

The proof is found in Section 7.3.2. This lemma shows that a random hash leads to a good iteration with probability p_θ, which is constant in our setting based on the assumption about n_A, n_I, n_R.

Recall that S_{x^*} was the random variable that represents the set of iterations j such that the min-hash u_j^B comes from x^* when P_2's input is B_{+x^*}. From Lemma 5.13, with overwhelming probability $|S_{x^*}| = O(\lg \lg \kappa)$.

Now, fix A, I, x^* and $h_1, \ldots, h_k$ and let G_θ be the set of iterations j in which a θ-good event takes place; i.e.,

$$G_\theta = \{j \in [k] : \mathsf{good}_\theta(h_j, A, I, n_B)\}\,.$$

Let $K_\theta = G_\theta \setminus S_{x^*}$. The following lemma shows that the θ-good events takes place $\Theta(\kappa)$-many times, with overwhelming probability.

Lemma 5.9. Suppose $k = \Theta(\kappa)$, $n_B = \Omega(\kappa^2)$, and $p_\theta \in \Theta(1)$. Let $s = |S_{x^*}|$. Then, we have

$$\Pr_{h_1,\ldots,h_k}\left[|K_\theta| > \frac{2}{3}(k - s)p_\theta\right] \geq 1 - \mathsf{negl}(\kappa).$$

The proof is found in Section 7.3.3.

Our goal. For a set W, define $c_W \overset{def}{=} \sum_{j \in W} \mathsf{Eq}(u_j^A, u_j^B)$. Let $\overline{K}_\theta := [k] \setminus K_\theta$. Note that the contributions to the final output can be divided into two parts:

- c_{K_θ}: The contribution from the iterations in K_θ, which contains all the θ-good iterations such that x^* does not hash to the minimum across B_{+x^*}.
- $c_{\overline{K_\theta}}$: The contribution from all the remaining iterations

Essentially, for any final count q, we are interested in comparing the two probabilities:

$$\Pr[c_{\overline{K_\theta}} + c_{K_\theta} = q] \text{ and } \Pr[c^{+x^*}_{\overline{K_\theta}} + c^{+x^*}_{K_\theta} = q].$$

Following our discussion on sensitivity in Section 5.4, the difference of $c_{\overline{K_\theta}}$ and $c^{+x^*}_{\overline{K_\theta}}$ is upper-bounded by $s = O(\lg \lg \kappa)$. Note that we have $c_{K_\theta} = c^{+x^*}_{K_\theta}$ because $j \in K_\theta$ implies $j \notin S_{x^*}$. Therefore, we only need to analyze the single distribution of c_{K_θ} as a noise and compare the following two probabilities:

$$\Pr[c_{K_\theta} = q] \text{ and } \Pr[c_{K_\theta} + s = q].$$

Distribution of c_{K_θ}. We have $c_{K_\theta} = \sum_{j \in K_\theta} c_j$, where $c_j = \mathsf{Eq}(u_j^A, u_j^B)$. Note that since we have $j \in K_\theta$, a θ-good event takes place in iteration j, i.e., $\min h_j(A) = \min h_j(I)$.

Let $\gamma_j = 1 - \min h_j(I)$. Note that the hash of each item R is randomly distributed in $\mathcal{F}_{\mathsf{minH}}{}^{(1)}$. Therefore, the probability that $c_j = 1$ is $(\gamma_j)^{n_R}$, in which case every hash of items in R must be at least $\min h_j(I)$.

Let $\eta_{-\theta} = 1/2 - \theta$ and $\eta_{+\theta} = 1/2 + \theta$. Since j is a good iteration, we have $(\gamma_j)^{n_R} \in [\eta_{-\theta}, \eta_{+\theta}]$. Therefore, letting $p_j = (\gamma_j)^{n_R}$, we have $c_j \sim \textsc{Ber}(p_j)$, where $\textsc{Ber}$ denotes the Bernoulli distribution. Since these Bernoulli distributions are independent from each other, can apply Lemma 5.10 below to conclude that $C_{K_\theta} \approx_{\epsilon,\delta} C_{K_\theta} + s$.

For $j \in [n]$, consider $c_j \sim \textsc{Ber}(p_j)$. With $p_J = \{p_j\}_{j=1}^n$, let $\mathsf{PB}(n, p_J)$ denote the distribution of $\sum_{j \in [n]} c_j$. This distribution is called a Additive Poisson Binomial distribution.

We conclude the proof by showing that the Additive Poisson Binomial distribution with appropriate parameters satisfies the following DP-like property.

Lemma 5.10. Consider an Additive Poisson Binomial distribution $\mathsf{PB}(n, p_J)$, where $n \in \Omega(\kappa)$ and for each p_j, it holds that $p_j \in [1/2 - \theta, 1/2 + \theta]$ where $\theta \in (0, 1/2)$ is a constant independent of κ. Then, for any constant ϵ and $s = \Theta(\lg \lg \kappa)$, there are $a, b \in [n]$ such that

- For any $\ell \in [a, b]$, $e^{-\epsilon} \leq \frac{\Pr[\mathsf{PB}(n,p_J)=\ell]}{\Pr[\mathsf{PB}(n,p_J)+s=\ell]} \leq e^{\epsilon}$.
- For any $\ell \notin [a, b]$, $\Pr[\mathsf{PB}(n, p_J) = \ell] = \mathsf{negl}(\kappa)$ and $\Pr[\mathsf{PB}(n, p_J) + s = \ell] = \mathsf{negl}(\kappa)$.

The proof is found in Section 7.3.4.

5.8 Highlights of Proof of Theorem 5.6

Here, we highlight only the important parts of the proof of Theorem 5.6. The full proof can be found in Section 7.3.5. We show that $\mathcal{F}_{\mathsf{minH}}$ satisfies DDP even when the size of the universe $\mathcal{U}$ of size $n_R \cdot \ell$ is polynomial in κ with $\ell = \Omega(n_R^3)$, and the secret set R is chosen from the uniform distribution on $\mathcal{U}$, conditioned on arbitrary leakage on R of length L, where $L \le n_R(\lg \ell - 3 \lg n_R - 2)$. WLOG, we assume that the adversary corrupts P_1.

We set $n'_R := n_R/3$; looking forward, it is the size of a subset $R' \subset R$, each of whose elements has high remaining min-entropy even after leakage (that we will define in the proof) is considered.

5.8.1 Min-hash Graph

Consider running the min-hash protocol $\mathcal{F}_{\mathsf{minH}}$ with k iterations such that k_g of them belong to G_θ. For this, we consider all the hash outputs in two different stages and define the following sets:

$$H_1 = \{h_j(A_{+x^*})\}_{j=1}^k, \; H_2 = \{h_j(\mathcal{U} \setminus A_{+x^*})\}_{j=1}^k.$$

Since we are in the random oracle model, each hash value is chosen uniformly at random. For our analysis, we construct the following bipartite graph $(\mathcal{X}, \mathcal{Y}, \mathcal{E})$, which we call the *min-hash graph*, based on the sets A, I and x^* along with the hash functions as follows:

$\mathbf{MinhashG}_{H_1}(A, I, x^*, H_2)$:

1. Set $\mathcal{X} = \mathcal{U} \setminus A_{+x^*}$. In other words, the graph considers *all potential elements that could be in* $R = B \setminus I$. A distribution of R is equivalent to a distribution of how to choose n_R nodes from $\mathcal{X}$. Note that H_1 determines G_θ (based on the hash values of A and I). We set $\mathcal{Y} = G_\theta$. In other words, $\mathcal{Y}$ corresponds to all the good iterations that could potentially positively contribute to the final count.

2. Let $p_j = \min h_j(I)$. Use H_2 to determine the set of edges:

$$\mathcal{E} = \{(i,j) : (i,j) \in \mathcal{X} \times \mathcal{Y} \text{ and } h_j(x_i) < p_j = \min h_j(I)\}.$$

   In other words, existence of an edge (i,j) means that if node i belongs to R, iteration j will *not* contribute to the final count.

3. Output the resulting bipartite graph $(\mathcal{X}, \mathcal{Y}, \mathcal{E})$.

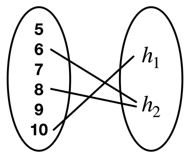


Figure 6: Min-hash graph

	1	2	3	4	5	6	7	8	9	10	11
h_1	0.83	0.25	0.77	0.85	0.93	0.35	0.86	0.92	0.49	0.21	0.5
h_2	0.62	0.83	0.27	0.59	0.63	0.26	0.4	0.26	0.72	0.36	0.6
h_3	0.68	0.11	0.67	0.29	0.82	0.3	0.62	0.23	0.67	0.35	0.7
h_4	0.02	0.43	0.22	0.58	0.69	0.67	0.93	0.56	0.11	0.42	0.8

Table 1: Example Hash Functions

Example. Let the universe be $\mathcal{U} = [11]$. Let $A = \{1, 2, 3, 4\}$, $I = \{2, 3\}$, $x^* = 11$. Let the threshold range for the θ-good iterations be $[0.2, 0.7]$. Assume that our protocol runs in 4 iterations using the hash functions defined in table 1.

Figure 6 shows the constructed min-hash graph. We have $\mathcal{X} = \{5, 6, \ldots, 10\}$ and $\mathcal{Y} = \{h_1, h_2\}$; h_3 has been ruled out since $p_3 = h_3(2) = 0.11 \notin [0.2, 0.7]$, and h_4 has been ruled out because $\min h_4(A) \neq \min h_4(I)$. Moreover, we have $p_1 = h_1(2) = 0.25$ and $p_2 = h_2(3) = 0.27$. Note that $(8, h_2) \in \mathcal{E}$, because $h_2(8) < p_2$.

5.8.2 Fixed Subsets (R', T) of Secret Items and Good Iterations

We fix H_1 and thereby the nodes $\mathcal{X}$ and $\mathcal{Y}$ of the min-hash graph. In this section, as the first step, we fix subsets $R' \subset \mathcal{X}$ and $T \subseteq \mathcal{Y}$ and analyze the noise over the choice of H_2. In other words, we are treating H_2 as private the adversary. Extending this, in the next section, we will consider the actual protocol setting where the hash functions are public and then analyze the noise over *a distribution of* R'.

Edge distribution in the min-hash graph. The probability (over the choice of H_2) that an edge (i, j) forms is exactly equal to p_j. Moreover, since we are in the random oracle model, the probability that (i, j) forms is independent of the probability that any other edge in the graph forms.

Noise distribution. We are interested in *the probability $E^{R'}_{T,r}$ over the choice of H_2 that the final count is reduced by exactly r due to the elements of R' over a bundle T of iterations.* In the random oracle model, the probability depends only on the size of the sets $n' = |R'|$ and $k_b = |T|$. Therefore, we will often use the notation $E^{n'}_{k_b,r} = E^{R'}_{T,r}$. We will sometimes even omit k_b and write $E^{n'}_{r}$. Observe that $E^{n'}_{r}$ is another way of representing an Additive Poisson Binomial distribution. That is, $E^{n'}_{k_b,r} = \Pr[\mathsf{PB}(k_b, p_J) = r]$. Therefore, based on Lemma 5.10, we have the following:

Corollary 5.11. Let $k_b \in \Omega(\kappa)$, and consider any H_1 that makes $|\mathcal{Y}| > k_b$ in the min-hash graph construction. For any $s = O(\lg\lg\kappa)$, any constant ϵ, there are $a, b \in [k_b]$ such that over the choice of H_2, we have

- For any $r \notin [a + s, b]$, then $E^{n'_R}_{k_b,r}$ and $E^{n'_R}_{k_b,r-s}$ are both negligible in κ.
- For any $r \in [a, b]$, then it holds $e^{-\epsilon/3} \leq E^{n'_R}_{k_b,r} / E^{n'_R}_{k_b,r-s} \leq e^{\epsilon/3}$.

The above indicates that the distribution over r is amenable for use as a noise distribution in a differential privacy context.

5.8.3 Noise Over the Choice of R' with Public Hash Functions

Our main technical challenge is to show that the properties needed for differential privacy hold *even when the hash functions are public.*

For this, we first fix H_1 and H_2. Then, we consider the derived min-hash graph $G = (\mathcal{X}, \mathcal{Y}, \mathcal{E})$. Let $\mathcal{D}$ be the distribution of R'. For any set T of iterations of size k_b and any integer r, let $I_{R',T,r}$ be the indicator random variable that is set to 1 if set R' contributes $-r$ to the total count in the min-hash protocol. We define a random variable $D_{T,r}$ that is *the probability that* R' *contributes to the noise reduction* r *over iterations in* T:

$$D_{T,r}(\mathcal{D}) := \Pr_{R' \sim \mathcal{D}}[I_{R',T,r}] = \sum_{R'} \Pr_{R' \sim \mathcal{D}}[R'] \cdot I_{R',T,r}.$$

Conditions for the hash functions. Ideally, we would like to show the following:

> For any fixed H_1 and H_2 and over distribution $\mathcal{D}$, it holds that $D_{T,r}$ and $D_{T,r-1}$ (and ultimately $D_{T,r-s}$) are close, except with the tail case of r whose probability weight is negligible.

The universal quantifier for H_1 and H_2 in the above can be *slightly relaxed so that the condition holds with all but small probability over the choice of the hash functions*, which can be captured by showing that $D_{T,r}$ is close to its mean $E^{n'_R}_{\hat{k},r}$ (and then applying Corollary 5.11).

Geometric collision property. This is essentially to show that $D_{T,r}$ is strongly concentrated around its mean. We could try to apply Chernoff bound to show the concentration property, but we cannot do so because $I_{R'_i,T,r}$ and $I_{R'_j,T,r}$ are not necessarily independent if $R'_i \cap R'_j \neq \emptyset$. Therefore, we instead use Chebyshev for bounding the tail, which requires $D_{T,r}$ to have small variance. Thus, our next goal is to upperbound $\mathsf{Var}[D_{T,r}]$. To do so, we introduce a property of distributions $\mathcal{D}$ over sets R' which we call the "Geometric Collision Property". In a nutshell, this property states that the probability that two sets R'_1, R'_2 drawn independently from $\mathcal{D}$ have intersection of size z is at most $(\frac{1}{n^{0.5}})^z$ for all $z \in [n']$. We show that $\mathsf{Var}[D_{T,r}]$ can be bounded for any distribution over sets R' that has this property.

Definition 5.12 (Geometric Collision Property)**.** Let $\mathcal{D}$ be a distribution over sets R' of size n'_R. We say that $\mathcal{D}$ has the *Geometric Collision Property* if for all $z \in [n'_R]$

$$\Pr_{R'_i, R'_j \sim \mathcal{D}} \left[|R'_i \cap R'_j| = z \right] \leq \left(\frac{1}{\sqrt{n_R}} \right)^z .$$

Based on this property, we can show the following lemma.

Lemma 5.13. Let $k_b \in \Omega(\kappa)$, and consider any H_1 that makes $|\mathcal{Y}| > k_b$ in the min-hash graph construction. Let $\mathcal{D}$ be a distribution over sets of size n'_R with geometric collision property. For any set T of size $k_b \in \Omega(\kappa)$, there exist $a, b \in [k_b]$, such that with probability $1 - O(\frac{k_b \cdot \lg^3(\kappa)}{\sqrt{n_R}})$ over choice of H_2, the following holds:

- For all $r \notin [a+s, b]$, $D_{T,r}$ is negligible, where $s = O(\lg \lg \kappa)$.
- For all $r \in [a, b]$, $e^{-\epsilon/3} E^{n'_R}_{k_b,r} \leq D_{T,r} \leq e^{\epsilon/3} E^{n'_R}_{k_b,r}$.

The proof is found in Section 7.3.6.

Multiple bundles of iterations towards DDP with negligible δ. We are not quite done yet. Using the above lemma, we are only able to reduce the failure probability only to $\sim 1/\sqrt{n}$, whereas we would like the failure probability to be negligible. In order to do that, we split the "good" iterations into u bundles, where u is a small superconstant number $u = \lg \lg \kappa$, and argue that with overwhelming probability at least one bundle serves as a good noise. Note that hash outputs are independent in each bundle and so the probability that all u bundles fail should be $(\frac{1}{\sqrt{n}})^u$, which is negligible. For this, we set the parameter $k_b = k_g/u$, where k_g is the number of good iterations.

5.8.4 Geometric Collision Property In the Face of Leakage

We conclude the proof by showing that R' indeed has the geometric collision property. It is not hard to see that the uniform distribution over all sets R' of size n' from a universe of size $n' \cdot \ell$ (where $\ell \in \Omega(n^3)$) satisfies the "Geometric Collision Property". It would seem, therefore, that we could take this as our secret distribution and the analysis would be complete. Unfortunately, even for the case in which the distribution is sets of size n' chosen uniformly at random from the universe, the analysis is not straightforward. The difficulty stems from the fact that the "noise" in the protocol is tied to the input itself. Therefore, if information about the input is leaked in any other part of the protocol, then the noise distribution changes and may no longer satisfy the required properties. Specifically in our case, learning the number of matches across the two parties' sets with respect to some of the hash functions leaks information about the secret set of the honest party (since the secret set affects those counts).

Strong chain-rule for min-entropy. We first observe that our initial min-entropy in the distribution over secret sets $\mathcal{R}$ is high (approximately $\frac{8n}{9} \lg \ell + 2n$) and that the entire information leaked about R from the counts of the iterations that are not θ-good is small. We can lower-bound the remaining min-entropy in $\mathcal{D}$, therefore, using the weak chain rule for min-entropy [58, Lemma 2.2].

If we want to use the weak chain rule to lower bound the remaining min-entropy with all but $2^{-\kappa}$ probability, however, we need to take a hit of κ in the min-entropy. Recall that each individual element in R can be viewed as being chosen from a set of size ℓ and thus has min-entropy of at most $\lg(\ell) \ll \kappa$. Thus, after applying the weak chain rule and losing more than κ bits of min-entropy, we can have certain elements that have only constant min-entropy, thus implying that collisions are likely in those positions. So the weak chain rule, while leaking only a small number of bits overall, can ruin the geometric collision property. Even worse, the min-entropy definition doesn't rule out the case in which *all* elements of R (i.e. the marginal distributions over each element in R) have only constant min-entropy, while the total min-entropy in R remains high!

This phenomenon has been previously observed and studied in the literature [164]. One way to deal with such a counter-intuitive situation is to actually leak a small amount of *additional* information, known as "spoiled" bits. This will lower the total min-entropy in R, but will ensure that a large fraction of blocks in R still have high min entropy of at least $1.5\lg(n)$. We extend the techniques of [164] to produce spoiling leakage so that the min-entropy in R still stays high in our protocol. We discuss more details about the strong chain rule in the next section.

5.8.5 Strong Chain Rule

Our strong chain rule considers min-entropy where leakage functions $\ell_1(\cdot), \dots, \ell_n(\cdot)$ are additionally considered. We describe our theorem in a general way, and we hope that it may find future applications in leakage-resilient cryptography.

Sequence of random variables. Recall that R is the set of secret items in the min-hash protocol. Here, we treat R as a sequence of block-by-block random variables $R = (R_1, \dots, R_n)$, associated with (potentially randomized) leakage functions $\ell_1(\cdot), \dots, \ell_n(\cdot)$ with randomness $\rho_1, \dots, \rho_n$. You can think of the blocks as coming in a streaming fashion in order of $R_1, R_2, \dots, R_n$.

Leakage functions. Loosely speaking, the properties we require of the leakage functions are that the i-th leakage ℓ_i can be computed given (R_i, ρ_i), and all the outputs of $(\ell_{i+1}, \ell_{i+2}, \dots, \ell_n)$. We also require that the total number of valid sequences of leakages from $\ell_1(\cdot), \dots, \ell_n(\cdot)$ should be sufficiently small (see Property 1 in Theorem 5.14 below).

Spoiling functions. Our theorem below states the existence of a spoiling function $f(\cdot)$ with certain properties, as well as properties of the random variables $(R_1, \dots, R_n)$ and $(\rho_1, \dots, \rho_n)$ conditioned on the output of the spoiling function $f(R)$.

The properties of $(R_1, \dots, R_n)$ and $(\rho_1, \dots, \rho_n)$ are roughly the following: (1) There exist disjoint sets V, W such that $V \cup W = [n]$ that are determined by $f(R)$. (2) Blocks $\{R_i\}_{i \in V}$ have high min-entropy conditioned on $f(R)$. (3) Blocks $\{R_i\}_{i \in W}$ have small support size (low max-entropy) conditioned on $f(R)$. (4) For $i \in V$, the random strings ρ_i are uniform random and independent conditioned on $f(R)$. (See Properties (5)-(8) in Theorem 5.14).

The properties of $f(\cdot)$ are roughly the following: (1) The failure probability (outputting $\perp$) is small. (2) As long as the total number of valid sequences of leakages from $\ell_1(\cdot), \dots, \ell_n(\cdot)$ is sufficiently small, the image size of f is small. This property ensures that we do not lose too much of the total min-entropy of R by releasing $f(R)$ (3) The leakages $\{\ell_i(\cdot)\}_{i \in W}$ can be computed given $f(R)$. (See Properties (2)-(4) in Theorem 5.14)

Our theorem. The main difference between our spoiling lemma and prior ones is that our min and max entropy guarantees on $R = (R_1, \dots, R_n) \mid f(R)$ hold even with respect to additional leakage $\{\ell_i\}_{i \in W}$ which is included in the spoiled bits $f(R)$.

Theorem 5.14 (Block structures with few bits spoiled and leakage)**.** Let $\mathcal{U} = U_1 \times \cdots \times U_n$ be a fixed universe and $R = (R_1, \dots, R_n)$ be a sequence of (possibly correlated) random variables where each R_i is over U_i (and all are disjoint) and $|U_i| = \ell$ for all i. Let $\rho_1, \dots, \rho_n$ be a sequence of uniformly random strings over $\{0,1\}^m$ and let $\ell_1(\cdot), \dots, \ell_n(\cdot)$ be leakage functions. Then, for any $\epsilon \in (0,1)$, any $\delta > 0$ and any $c \in [2^\delta, \ell/2^\delta]$, there exists a spoiling leakage function $f(R)$ that satisfies the following properties.

1. A sequence $\beta_1, \dots, \beta_n$ is valid if for all $i \in V$, $\beta_i = \perp$ and for all $i \in W$, $\beta_i = \ell_i(R_i, \rho_i, \beta_{>i})$, where $\beta_{>i} = (\beta_{i+1}, \dots, \beta_n)$. We require that the number of valid sequences $\beta_1, \dots, \beta_n$ is at most B.

2. It holds that $\Pr_R[f(R) = \perp] \leq \epsilon n$.

3. $|Im(f)| \leq B \cdot (2(\lg(\ell) + \lg(1/\epsilon))/\delta)^n$.

4. Conditioned on any $y \in Im(f) \setminus \{\perp\}$, for all $i \in W$, the leakage $\ell_i(R_i, \rho_i, \beta_{>i})$ can be computed from y. Here, $\beta_j = \perp$ if $j \in V$ and $\beta_j = \ell_j(R_j, \rho_j, \beta_{>j})$ otherwise.

5. Let $Im(f)$ be the set of images of f. Every $y \in Im(f) \setminus \{\perp\}$ specifies two disjoint sets V and W such that $V \cup W = [n]$.

6. Conditioned on any $y \in Im(f) \setminus \{\perp\}$, for every $i \in V$, every element in distribution $R_i \mid R_{<i}$ has low probability weight, i.e.,

$$\forall y \in Im(f) \setminus \{\perp\}, \forall r \text{ s.t. } f(r) = y, \forall i \in V: \quad \Pr\left[R_i = r_i \;\middle|\; R_{<i} = r_{<i},\ y\right] \leq \frac{2^\delta}{c}.$$

7. Conditioned on any $y \in Im(f) \setminus \{\perp\}$, for every $i \in W$, it holds that $R_i \mid R_{<i}$ has small support size, i.e.,

$$\forall y \in Im(f) \setminus \{\perp\}, \forall r \text{ s.t. } f(r) = y, \forall i \in W:$$
$$|\{r_i : \Pr[R_i = r_i | R_{<i} = r_{<i}, y]] \geq 0\}| \leq 2^\delta \cdot c.$$

8. $\{\rho_i\}_{i \in V}$ are distributed independently and uniformly at random conditioned on $f(R)$.

The proof is found in Section 7.3.7. Typically, one would like to set c as large as possible, while ensuring that the size of V remains above some threshold. The achievable tradeoffs between c and $|V|$ are determined by the min-entropy of R before the spoiling bits $f(R)$ are released. For our applications, we require $c = n^{1.5}$ and $|V| \geq n/3$. In Section 7.3.8, We show that our min-entropy assumption on R implies that this parameter setting is achievable.

5.9 Empirical Evaluation

5.9.1 Comparison With Prior Work

We compare our noisy min-hash (NMH) protocol π_{NMH} with the current state-of-the-art approach, called sketch-flip-merge (SFM) [101] and the generalized randomized response mechanism (GRR) [10]. In particular, we evaluate the trade-off between communication cost and cardinality estimation accuracy, while achieving (almost) the same level of privacy guarantee as follows:

- For a given privacy parameter ϵ (with δ fixed to 2^{-40}), we choose the right amount of noise for our protocol and vary the number of hash functions k to measure the communication cost and estimation accuracy trade-off.
- We then compare these accuracy results with the state-of-the-art protocols using the same communication and privacy parameter.

  We use the relative root mean squared error (RRMSE) of the union size as our accuracy metric. This choice is primarily to ensure a fair comparison between our protocol and the SFM protocol. Further details on this matter are provided in the discussion of the SFM protocol below.

  In Figure 7, we demonstrate the comparison of NMH, SFM, and GRR.

Our protocol. We calculate the communication of our protocol π_{NMH} with $\mathcal{F}_{\mathsf{psi\text{-}ca}}$ instantiated with the PSI-CA protocol described in Figure 23. It is a variant of the protocol in [53], where H_2 is applied to $\{a'_i\}_{i\in[v]}$ and $\{b'_j\}_{j\in[w]}$ in order to reduce the communication. The original protocol is secure under the DDH assumption in the random oracle model. Essentially the same security proof found in the original paper can be applied to show the security of this variant, when H_2 is also modeled as a random oracle.

Protocol 23 **Private Set Intersection Cardinality**

Let G be a multiplicative group of order q. Let $H_1 : \{0,1\}^* \to G$ and $H_2 : \{0,1\}^* \to \{0,1\}^\lambda$ be hash functions.
Input: P_1 has $C = \{c_1, \ldots, c_v\}$ and P_2 has $S = \{s_1, \ldots, s_w\}$.

1. P_1 samples a random exponent $R_c \leftarrow \mathbb{Z}_q$. For $i \in [v]$, P_1 computes $a_i = H_1(c_i)^{R_c}$. P_1 sends $(a_1, \ldots, a_v)$.
2. P_2 samples $R_s \leftarrow \mathbb{Z}_q$ and computes $(a'_1, a'_2, \ldots, a'_v) = \mathsf{shuffle}(a_1^{R_s}, \ldots, a_v^{R_s})$. P_2 also computes $(b_1, b_2, \ldots, b_w) = \mathsf{shuffle}(H_1(s_1)^{R_s}, \ldots, H_1(s_w)^{R_s})$. P_2 sends $(H_2(a'_1), \ldots, H_2(a'_w))$ and $(b_1, \ldots, b_w)$ to P_1.
3. P_1 computes $(b'_1, \ldots, b'_w) = (b_1^{R_c}, \ldots, b_w^{R_c})$. P_1 outputs the following value:

$$| \{H_2(a'_1), \ldots, H_2(a'_v)\} \cap \{H_2(b'_1), \ldots, H_2(b'_w)\}|.$$

We briefly sketch the security proof here while referring the full proof to the original paper [53]. We first show the simulator for the corrupted P_1. Let t be the protocol output (i.e., set intersection cardinality). The simulator chooses random t indices $(i_1, \ldots, i_t)$ (resp., $(j_1, \ldots, j_t)$) from $[v]$ (resp., $[w]$). In order to prepare $(b_1, \ldots, b_w)$, the simulator replaces $H_1(s_{j_k})^{R_s}$ (for $k \in [t]$) with $H_1(c_{i_k})^{R_s}$, and the remaining values $H_1(s_h)^{R_s}$ are simulated with random numbers. Since H_1

is a random oracle (i.e., for an input x, we have $H_1(x) = g^r$ for a random r), this simulation is indistinguishable under the DDH assumption. When P_2 is corrupted, the first message $\{a_i = H_1(c_i)^{R_c} : i \in [v]\}$ is simulated by random values. The simulation is also indistinguishable under the DDH assumption. The above PSI-CA protocol exchanges $v + w$ elliptic curve points and w hashes, resulting in $(v + w) \cdot 256 + w \cdot 80$ bits. In protocol π_{NMH}, the parties will run this PSI-CA protocol by setting $v = w = k + 2\ell_B$.

Sketch-Flip-Merge (SFM) [101]. While our main focus is on comparing the accuracy of Jaccard Index estimation, in the absence of available code, we had to rely on their analysis of the relative root mean squared error (RRMSE) of cardinality estimation instead of Jaccard Index estimation. This poses challenges in evaluating the accuracy of the Jaccard Index for SFM. In particular, although the Jaccard Index can be estimated by calculating the ratio of estimated intersection size over the estimated union size, its RRMSE cannot be directly calculated from RRMSEs for the intersection and union sizes. This is because the two estimates have dependency, and we can only conjecture that the derived estimate through the division operation will probably have a worse RRMSE.

In the end, giving a slight advantage to SFM, we decided to focus on the accuracy of cardinality estimation of the size of the union only. In our case, the union size was estimated based on the Jaccard Index from the min-hash protocol and n_A and n_B. Following the approach of SFM [101], we perform $m = 1000$ estimates to measure the accuracy in the form of relative root mean squared error (RRMSE); that is, letting $\hat{n}_{U,1}, \ldots, \hat{n}_{U,m}$ be the union size estimates, and n_U be the real union size, we define $\mathrm{RRMSE}(\hat{n}_{U,1}, \ldots, \hat{n}_{U,m}; n_U)$ to be $\frac{1}{n_U}\sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{n}_{U,i} - n_U)^2}$. To match the communication complexity, we set the sketch of SFM to be a $(B \times P)$-bit matrix such that $B \cdot P = 592w$ and $P = 24$.

Generalized Randomized Response (GRR) [10]. We also compare our protocol with the generalized randomized response MinHash protocol in [10]. Following their guidance in experiments, we select the range of their hash function to be a single bit and let their protocol use $592w$ hash functions to match our communication cost.

Since their actual protocol would take too long to run for large n and k, in order to facilitate the large number of hash functions, we wrote code simulating the error based on their privacy and utility analysis. As with the other protocols, we perform 1000 estimates to lower the variance of the errors. To align with our other comparison with SFM, we report the relative root mean squared error of the union size.

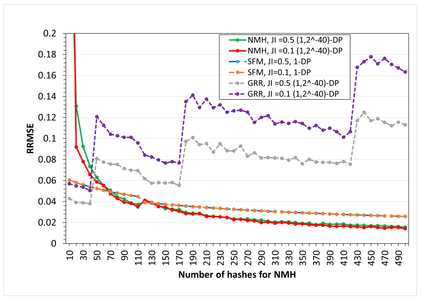


Figure 7: Accuracy: NMH, SFM, GRR.

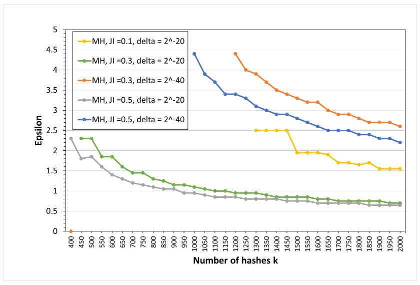


Figure 8: DDP based on k, and JI

Comparison Results. In Figure 7, we demonstrate the comparison of NMH, SFM, and GRR. We set $n = 10^6$. The result shows that our error is consistently smaller than both SFM and GRR for a reasonable range of communication costs, which corresponds to the usage of $k \in [100, 500]$ hashes for our noisy min-hash protocol. Specifically, as we increase the number of hash functions, both our protocol and SFM achieve increased accuracy, due to larger sketch sizes that better represent the input sets. On the other hand, while GRR performs well with smaller communication, adding more communication becomes counter productive. This is because each additional bit in their protocol corresponds to an extra hash function output, which increases the noise needed to achieve the same privacy guarantee. Finally, both GRR and our protocol exhibit spikes in the accuracy trends, corresponding to crossover points where a significant amount of additional noise is required to keep δ from getting above 2^{-40}.

In concluding remarks, it is noteworthy that both SFM and GRR protocol discloses the entire noisy sketch, revealing collective information about the party's input set. We note that differential privacy does not prohibit revealing collective information about the inputs; rather, it mandates that individual contributions should not be discernible in the output. In contrast, our min-hash protocol employs secure two-party computation and discloses no information about the input set, except for the final $(\lg k)$-bit output. When deciding which scheme to use, depending on the specific use case, this observation may need to be taken into account.

5.9.2 DDP of Noiseless Protocol

We empirically evaluate the DDP guarantee for the noiseless min-hash protocol in the public hash setting, in which each element in the secret set has high entropy. As before we set $n_A = n_B = 10^6$. Figure 8 shows how the privacy parameter ϵ changes with the number k of iterations. Although we demonstrate the results for $JI \leq 0.5$, the results for $JI > 0.5$ are similar. We omit the data points where ϵ is greater than 5, which happens when k is small, and focus on the more meaningful ϵ range. We observe the following:

- Roughly speaking, when the number k of iterations of the min-hash protocol is reasonably large (at least 500), the noiseless min-hash protocol provides a decent level of DDP with privacy parameter $\epsilon \in [0.5, 5]$.

- Higher values of k correspond to improved privacy parameters. Note that as k grows, more iterations will be θ-good. Since hashes of non-intersecting items work as noise in θ-good iterations, more θ-good iterations will essentially amount to adding more noise, therefore offering better privacy guarantee.
- When k is the same, the best privacy parameter is achieved when JI is around 0.5. This is due to the likelihood that a hash function is θ-good being maximized when striking a balance between the two conditions stipulated in Definition 5.7: (i) the hash of an intersecting item should be the minimum hash value, and (ii) the minimum hash value is neither too large nor too small.

6 FACTS

6.1 Introduction

The proliferation of fake and misleading information online has had significant impact on political discourse [106] and has resulted in violence [158]. Large services like Facebook and YouTube have begun to remove or label content that they know to be fraudulent or misleading [73, 188], through a combination of a manual process of reviewing posts/videos and automated machine learning techniques.

However, on end-to-end encrypted messaging services (EEMS), like Signal, WhatsApp, Telegram, etc., where so-called "fake news" is also shared, such review is impossible. At no point do the providers see the plain-text, unencrypted contents of messages transmitted through their systems and thus cannot identify and remove offending material. Such platforms must instead rely on their users to identify and report malicious content. Even then, identifying and removing *users* who repeatedly post misleading and dangerous content may still be difficult because some platforms, like Signal, also hide the path the message took, so identifying and addressing the original source of the misinformation may not be possible.

Tyagi et al. [177] introduced a first approach for overcoming this challenge and allow EEMS to effectively *traceback* an offending message to find the originator based on a user complaint. The traceback procedure also assures that all other messages remain private and that innocent parties cannot be blamed for originating the offending messages.

While innovative, there are two notable shortcomings of Tyagi et al.'s traceback scheme. First, it requires extensive "housekeeping" on the part of the platform that scales as the number of messages in the system. Second, a single, possibly malicious, complaint can trigger a traceback and thus reveal the message contents as well as the history of prior recipients, which is counter to the goals of EEMS to maintain the privacy of users communicating through this system. One malicious user (e.g., a government agent) can reveal the source of a piece of information (e.g., a leak) that they have received, violating the privacy of the sender (e.g., the leaker) by issuing a single complaint to the EEMS. While it may be possible to apply manual review to these complaints, the scale of possible complaints could make this impractical. Several follow-on papers [110, 151] show how to achieve *source-tracking* for EEMS to identify the source of a message without relying on traceback of all intermediate recipients. However, these systems still allow a single complainer to trigger the source-tracking.

In this paper, we aim to resolve this conflict between privacy and ability to identify misinformation in EEMSs by first observing that "fake news" messages are, by definition, viral and are thus received, and likely complained about, by a large number of users. Private messages, such as leaks, on the other hand, are likely to be targeted and are thus only received by a small number of users; indeed, any message received by only a few users is inherently less impactful overall and more likely deserving of privacy protections. This leads to a more nuanced approach for identifying fake news: apply a threshold approach to complaint management, whereby only viral fake news would overcome the threshold and trigger an audit.

Counting the number of complaints in a private manner is a non-trivial problem if the privacy of the EEMS' clients is to be maintained prior to the threshold being reached, even given available cryptographic solutions. For example, a homomorphic encryption solution (e.g., [120]) would enable the checking and updating of counts for each message, but the access patterns of clients checking and updating counters could reveal how many complaints a message receives even if the threshold is not reached. Oblivious RAM (ORAM) (e.g. [90, 170]) could be used to protect the access patterns, but have high computational overheads and usually assume clients may share secrets and are not malicious. Pricate Information Retrieval (PIR) does not assume clients are trusted, but has different scalability challenges and does not address the problem of obliviously updating without revealing which message is being complained about.

We propose a different approach we call a Fuzzy Anonymous Complaint Tally System (FACTS). FACTS maintains an (approximate) counter of complaints for each message, while also ensuring that, until a threshold is exceeded, the status of these counters is kept private from the server and all users who have not received the message. FACTS builds on top of any end-to-end encrypted messaging platform, incurring only small overhead for message origination and forwarding. In particular, FACTS maintains the communication pattern of the underlying messaging system, requiring no new communication or secrets between users even for issuing complaints.

To avoid the high overheads of existing solutions, FACTS uses a novel oblivious data structure we call a *collaborative counting Bloom filter* (CCBF). This data structure allows us to obliviously increment and query approximate counters on millions of messages while only requiring 12MB of storage. Moreover, incrementing a counter only requires flipping *one bit* on the server and only uses the minimal communication of $\log |T|$ bits to address a single bit in the server-stored bit vector T. While the resulting counters are only approximate, we show experimentally and analytically that we are able to enforce the threshold on complaints with good accuracy, namely, below 10% error in theory, and below 3% in most realistic deployment scenarios.

The contributions of this paper are as follows:

- We develop a collaborative counting Bloom filter, a new oblivious data structure for counting occurrences of a large number of distinct items.
- We use this data structure to instantiate a provably-secure system, FACTS, for privacy-preserving source identification of fake news in EEMSs.
- Finally, we perform experiments to show the accuracy and overhead of FACTS in realistic deployment scenarios.

6.1.1 Setting and Goals

FACTS is built on top of an end-to-end encrypted messaging system (EEMS). For this work, we focus on the setting of server-based EEMSs with a server S that enables (authenticated) encrypted communication between the system users. Examples of such EEMSs include Signal and WhatsApp, among many others.

To make sure that FACTS is compatible with existing encrypted messaging systems, we make the following performance requirements:

1. **Messaging costs:** Originating and forwarding messages should incur little computational overhead for both users and the server over the standard procedure in the encrypted messaging system,
2. **Server storage:** The storage overhead of the server should be small (i.e., a single table not exceeding a few MBs),
3. **User costs and requirements:** Issuing complaints requires a small amount of communication and computation from the complaining user, and no cost to other users. Moreover, complaints can not require direct communication between users or require the users to have any apriori shared secrets that are not known to the server.
4. **Complaint throughput:** Issuing complaints may be slower than standard forwarding of messages, but the system must be able to handle millions of complaints per day.

To ensure privacy of messages and complaints, FACTS requires that complaints remain hidden from the server (and colluding clients) until a threshold of complaints is reached. Additionally, FACTS ensures integrity of the complaint process ensuring correctness of complaint counts and the identity of the revealed originator once the threshold is reached. Specifically, FACTS satisfies the following security guarantees:

1. **Message privacy:** All messages remain end-to-end encrypted and private from the server and non-receiving clients until a threshold of complaints is reached and an audit is issued. Moreover, even after the audit, only information about the audited message is revealed.

2. **Originator integrity:** Once a threshold of complaints is reached on a message, FACTS will only identify information about the true originator of the message. In particular, no innocent party can be framed as the originator.

3. **Complaint privacy:** The server and any colluding clients who have not received a message x should have no information about the number of complaints on x. In particular, the server should not be able to tell what message is being complained about.

4. **Complaint integrity:** A set of malicious clients should not be able to alter the number of complaints on any message x. Specifically, they cannot block or delay complaints, and cannot (significantly) increase the number of complaints on a message x except through the legitimate complaint process.

6.1.2 Building FACTS

Recall that our goal is to enable privacy-preserving counters to tally complaints on each message m. This suggests an immediate solution where the server stores an encrypted counter for each message, and clients interact with the server to increment the counter and check the threshold. While implementing such counters is certainly possible using homomorphic encryption [85] or standard secure computation techniques [21,88,186] , the problem is that the access pattern of clients' updates to counters leaks information to the server by revealing the complaint histogram. This suggests a further modification to store the counters inside an oblivious RAM (ORAM) [90] to hide such access patterns from the client. However, in our setting this would require a multi-client ORAM [38, 102, 131] which incurs significant performance penalties including at least $O(\log n)$ communication overhead when there are n distinct messages. Moreover, this would require direct communication between clients to maintain their ORAM state, and additionally, no security against malicious clients.

In FACTS, we take a different approach. Instead of relying on encryption to hide the counters from the server, we hide the counters in plain sight by mixing together the counters for all the messages in a way oblivious to the server. To make this possible, we relax the functionality of FACTS to only enforce approximate, rather than exact, thresholds. That is, the threshold will be triggered on a message x after $(1 \pm \epsilon)t$ complaints for a small error ϵ. Making this relaxation allows us to use a sketch-based approach for counting the complaints.

To achieve this functionality obliviously, we develop a collaborative counting Bloom filter (CCBD). This data structure consists (roughly) of a collection of Bloom filters, one for each message, where the Bloom filters corresponding to different messages are mixed together to hide them from the server. Specifically, the server stores a table of s bits. A random subset of v bits (V_x) is assigned to each message x at origination; these bits will be used for tracking complaints about this message (for intuition, one can think of these bits as forming a Bloom filter for storing the set of complaints about the message). We stress that the server has no information about which bits correspond to which messages.

To complain about a message x, a user who has received x can find the corresponding bit locations, and will (attempt to) flip one of the bits from 0 to 1. However, allowing users to flip any bit they choose, would allow malicious users to significantly accelerate complaints for a message they wish to disclose. To prevent this behavior, we restrict each client to only be able to flip (i.e., complain on) a small (of size u) set of locations U_C. Thus, to complain about a message m, a client first identifies the set V_x of bits corresponding to x. Then, she checks how many of these bits have already been set to 1, and if this exceeds a specified threshold, notifies the server to trigger an audit. If the threshold for x is not yet exceeded, the client sees whether any of the 0 bits in V_x are in her set U_C, and if there are any such bits, she flips one of them (chosen at random) from 0 to 1. Otherwise, the user still flips a random bit in their set U_x, so the server cannot discern anything about the message being complained on. We prove in Section 6.4 that the actual number of complaints necessary to trigger an audit can be calculated with high precision allowing us to (approximately) enforce the desired threshold.

6.1.3 Limitations of FACTS

In order to present FACTS, it is also important to recognize what our system does *not* do.

First, unlike some prior work, e.g. [84], [125], FACTS does not attempt to automatically detect misinformation. Instead, it relies on users reporting it when they see it. This reliance on users has inherent benefits and limitations. While our system is not subject to the kinds of machine-generated false positives that can arise from, e.g., hash collisions [31], our model is inherently vulnerable to any sufficiently large group of dishonest users, who could trigger an audit on a benign message. This is why we suggest the possibility of a manual human review process on message contents before the service provider would take any action on an audited message; see Section 6.8.

Second, due to the approximate nature of FACTS, it works most effectively for relatively large thresholds, say in the hundreds and above. For our application to fake news detection, this is reasonable as such messages are likely to garner a large number of complaints, and indeed this was our main motivation for this paper. We leave as interesting possible future work to implement a system supporting smaller thresholds, even as small as 2, efficiently.

One additional functionality limitation is that, as is true with any application using Bloom filters, the CCBF data structure can fill up once too many complaints have been registered. To deal with this issue it is necessary to periodically reset the counters and refresh the CCBF data structure. We refer to each such refresh period as an *epoch*, and in the remainder of the paper only present algorithms for a single epoch.

Finally, on the security side, an important limitation is that FACTS reveals meta-data on who issues complaints (but not what message they complain on). It is important to consider what is revealed by this meta-data. By observing the timing of messages and complaints, the server can make some inferences about what messages users are sending and complaining about. For example, suppose that the server sees that A sends a message to B, and then B issues a complaint. Then, it may be reasonable for the server to assume that A has sent the message which B complained about, even though this is not directly leaked by our system. Nonetheless, our definition guarantees that the server cannot be certain that this is indeed the case. We note that the messaging meta-data is already a byproduct of the underlying EEMS platform. FACTS only adds complaint meta-data to this leakage; see Section 6.8 for some further discussion.

6.1.4 Paper Layout

The remainder of the paper is organized as follows. In Section 6.2, we introduce some of the notation we use throughout the paper. Then, in Section 6.3 we describe the syntax and functionality of FACTS. In Section 6.4 we present and analyze our main building block, the CCBF data structure. Then, in Section 6.5, we show how to use a CCBF to instantiate FACTS. We demonstrate the accuracy and performance through experimental evaluation in Section 6.6 and then prove the security of FACTS in Section 6.7. Finally, we describe some variants of FACTS and directions for future work in Section 6.8 and present related work in Section 6.9.

6.2 Preliminaries

We use $[n]$ to denote the set $1, \ldots, n$. We write $x \leftarrow X$ to indicate that the value x is sampled uniformly at random from the set X. We use κ to denote a statistical security parameter and κ to denote a computational security parameter. We also assume the existence of a hash function $H : \{0,1\}^* \rightarrow \{0,1\}^*$ which is modeled as a random oracle. We let $\mathsf{poly}(\cdot)$ denote a polynomial function and $\mathsf{negl}(\cdot)$ denote a negligible function.

6.3 Fuzzy Anonymous Complaint Tally System (FACTS)

In this section, we present the syntax for FACTS and describe how FACTS is used. We show how to instantiate FACTS in Section 6.5.

Assumptions: We assume that each user A has a unique identifier ID_A, and that the server can authenticate these IDs. (We will abuse notation to use A to represent the user and also the id ID_A). We also assume that the server has an identifier ID_S (we will denote this by S) that can be authenticated by all users.

Additionally, we assume that the underlying end-to-end encrypted messaging system (EEMS) offers methods send(A, B, x) and receive(A, B, x) for sending and verifying a message x sent from user A to user B. Moreover, we assume that this communication is encrypted and authenticated. In particular, receive verifies that the received message was sent by A and was not modified in transit. Importantly, we do not assume that this platform is anonymous, instead assuming that the full messaging history i.e., who sent a message to whom and the size of that message, is available to the server.

Syntax: FACTS is a tuple of protocols $\mathsf{FACTS} = (\mathsf{Setup}, \mathsf{SendMsg}, \mathsf{RcvMsg}, \mathsf{Complain}, \mathsf{Audit})$. The first is used to set up FACTS, the next two are used to send and verify messages, while the last two methods are used to issue complaints and audit received messages.

- $\mathsf{Setup}(c)$: This takes as input an upper bound on the total number of users and initiates the FACTS scheme for c users.
- $\mathsf{SendMsg}(A, B, \mathsf{tag}_x, x)$: This method is used by a user A to send a message x to another user B. This may be a new message *originated* by A (indicated by $\mathsf{tag}_x = \perp$) or a forward of a previously received message.
- $\mathsf{RcvMsg}(A, B, \mathsf{tag}_x, x)$: This algorithm is run by B upon receiving a message (tag_x, x) from A.

  This algorithm checks whether tag_x is indeed a valid tag generated by A on message x. If this is the case, then B accepts the message, otherwise he rejects the message.
- $\mathsf{Complain}(C, \mathsf{tag}_x, x)$: This protocol is run by a user C to complain about a received message (tag_x, x).
- $\mathsf{Audit}(C, \mathsf{tag}_x, x)$: This protocol issues an audit of a message x revealing (tag_x, x) to S. This will be called by C when the number of complaints on m exceeds a pre-defined threshold (with high probability).

Usage: The following workflow demonstrates the standard usage of FACTS. To originate a new message x, a user A runs the $\mathsf{SendMsg}$ protocol with the server S to create metadata tag_x. $\mathsf{SendMsg}$ then sends this metadata and the message (tag_x, x) to the receiving user B using the messaging platforms send method. Upon receiving a message (tag_x, x), B first locally runs $\mathsf{RcvMsg}(A, B, \mathsf{tag}_x, x)$ to verify that the received message and tag are valid, if this fails he ignores the message. To forward a received message (tag_x, x), a user A runs $\mathsf{SendMsg}$ with the server S to produce metadata tag'_x; A then discards this metadata, and the original message (tag_x, x) is sent instead using the messaging platform's send method.[11]

If a user B receives a message (tag_x, x) that it considers "fake", he can use the $\mathsf{Complain}$ protocol to issue a new complaint on this message. After issuing a complaint, B checks whether the threshold of complaints on x has been reached.

[11]We note that since the underlying messaging scheme is encrypted, the actual ciphertext sent will not be the same as the ciphertext received.

If so, he calls Audit to trigger an audit on the message (tag_x, x), revealing x and the originator of x to the server S.

We note that users may join and leave during the execution of FACTS as long as the total number of identifiable users does not exceed c.

6.4 Collaborative counting Bloom filter

Our system records complaints in a special data structure which we call a *collaborative counting Bloom filter*, or CCBF. This data structure shares some of the same basic functionality as a counting Bloom filter [75, 139] or count-min sketch [50], which is to insert elements and compute the (approximate) frequency of a given element.

Our CCBF differs from a usual count-min sketch in that each update operation is accompanied by a *user id*, and each user can only perform a single update for a given element. This can be thought of as a strict generalization of the normal count-min sketch operations, where the latter may be simulated by our CCBF by choosing a unique user id for each update.

The actual data structure for the CCBF is also far simpler than the 2D array of integers used for a count-min sketch; instead, we store only a single length-s bit vector T. As a result, our CCBF will have the following desirable properties:

- The bit-length of T scales linearly with the total number of insertions.
- Each witness operation (insertion) changes exactly one bit in the underlying bit vector from 0 to 1.
- The CCBF is *item-oblivious*, meaning that after observing an interactive update protocol, the adversary learns which user id made the update, but not which item was updated.

The downside to our CCBF is a far lower accuracy of the count operation in general compared to count-min sketches. However, we will show that, for careful parameter choices, the count operation is highly accurate within a certain range, which is precisely what is needed for the current application.

6.4.1 CCBF Construction

The CCBF consists of a single size-s bit vector T and two operations:

- $\mathsf{Increment}(x, C)$: Increases the count by 1 for item x according to user id C.
- $\mathsf{TestCount}(x, t)$: Returns true if the number of increments performed so far for item x is *probably* greater than or equal to t.

Note that TestCount is probabilistic, in the sense that it may return false when the actual count is greater than t, or true when the actual count is less than t. Our construction guarantees the correctness probability is always at least $\frac{1}{2}$, and our tail bounds below show the correctness probability quickly goes towards 1 when the actual count is much smaller or larger than t.

The performance and accuracy of the CCBF is governed by three integer parameters s, u, and v, with $u, v \leq s$, which must be set at construction time. The first, s, is the fixed size of the table T. Each user i is randomly assigned a static set of exactly u locations in the T; i.e., a uniformly random subset of $\{0, 1, 2, \ldots, s-1\}$, which we call the *user set*. Similarly, each possible item x is assigned a random set of exactly v bit vector locations, which we call the *item set*.

The two CCBF operations can be implemented by a single server and any number of clients. The protocols are simple and straightforward, save for the calculation of the *tipping point* τ which we present in the next subsection.

In these protocols, the size-s bit vector T is considered *public* or *world-readable*; it is known by all parties at all times. In reality, the server who actually stores T may send it to the client periodically, or whenever a client initiates a Increment or TestCount protocol. However, the bit vector T is only writable by the server.

The Increment(x, C) protocol, outlined in Algorithm 24, involves the User attempting to set a single bit from 0 to 1 within the item set for x. However, the user is only allowed to write locations within their own user set. So, if there are no 0 bits in the intersection of these two index sets, the user instead changes any other arbitrary 0 bit in its own user set in order to maintain item obliviousness.

Algorithm 24 Increment(x, C)

```
1. User and server separately compute the list of $u$ user locations for user $C$,
   $U_C \subseteq \{0, \ldots, s-1\}$.
2. User computes list of $v$ item locations for item $x$, $V_x \subseteq \{0, \ldots, s-1\}$
3. User checks each location in $U_C$ in the table $T$ to compute a list $S_C =$
   $\{i \in U_C \mid T[i] = 0\}$ of settable locations for user $C$
4. If $S_C = \emptyset$, then the user cannot proceed and calls abort.
5. Else if $S_C \cap V_x \neq \emptyset$, user picks a uniformly random index $i \leftarrow S_C \cap V_x$ and
   sends index $i$ to server.
6. Else user picks a random index $i \leftarrow S_C$ and sends index $i$ to server.
7. Server checks that received index $i$ is in the user set $U_C$ and that $T[i] = 0$,
   then sets $T[i]$ to 1.
```

Since the bit vector T is considered world-readable, the only communication here is the single index i from client to server over an authenticated channel. In reality, to avoid race conditions, the server will actually send the table entry values $T[i]$ for all $i \in U_C$ to the user first and lock the state of the global bit vector T until receiving the single index response back from the user.

The TestCount(x,t) protocol is not interactive as it only requires reading the entries of T. The precise computation of the *tipping point* τ is detailed in the next section. Note that this computation depends only on the *total* number of bits set in the bit vector T as well as the parameters s,u,v; therefore the computation of τ is independent of the item x and could for example be performed once by the server and saved without violating item obliviousness.

This protocol is detailed in Algorithm 25.

Algorithm 25 TestCount(x,t)

1. Use parameters s,u,v and current value of m total number of bits set in T, to compute the tipping point τ.
2. Compute list of v item locations for item x, $V_x \subseteq \{0,\dots,s-1\}$
3. Check how many bits of T are set for indices in V_x. Return **true** if and only if this count is greater than or equal to τ.

6.4.2 Calculating the tipping point

The key to correctness of the TestCount protocol is a calculation of the *tipping point* τ, which is the expected number of 1 bits within any item set, if that item has been incremented t times. We now derive an algorithm to compute this expected value exactly, in $O(tv)$ time and $O(v)$ space.

Let s be the total size of the table T and $m \le s$ be the total number of calls to Increment so far. That is, m equals the number of 1 bits in T. Recall that $u,v \le s$ are the number of table entries per user and per item, respectively.

We first derive the probability that two subsets of the s slots have given-size intersection. Next we derive a recursive formula for τ using these intersection probabilities. The nearest integer to τ can then be efficiently computed using a simple dynamic programming strategy.

Intersection probabilities

For the remainder, we use Knuth's notation $n^{\underline{k}}$ to denote the *falling factorial*, defined by

$$n^{\underline{k}} = \frac{n!}{(n-k)!} = n \cdot (n-1) \cdot (n-2) \cdots (n-k+1).$$

Lemma 6.1. Let k,a,b,s be non-negative integers with $k \le b \le a \le s$, and suppose S and T are two subsets of a size-s set with $|S| = a$ and $|T| = b$, each chosen independently and uniformly over all subsets with those sizes. Then

$$\Pr(|S \cap T| = k) = \frac{a^{\underline{k}} \cdot b^{\underline{k}} \cdot (s-a)^{\underline{b-k}}}{s^{\underline{b}} \cdot k!}. \tag{3}$$

Proof. The number of ways to choose S and T with a size-k intersection, divided by the total number of ways to choose two size-a and size-b sets, equals

$$\frac{\binom{s}{k} \cdot \binom{s-k}{a-k} \cdot \binom{s-a}{b-k}}{\binom{s}{a} \cdot \binom{s}{b}}.$$

This simplifies to (3). ■

Because the numerator and denominator are each products of $b+k$ single-precision integers, the value of (3) can be computed in $O(b)$ time to full accuracy in machine floating-point precision.

Furthermore, equation (3) has the convenient property that, after altering any value a, b, or k by ± 1, we can update the probability with only $O(1)$ additional computation. So, for example, one can compute the probabilities for every $k \leq b$ in the same total time $O(b)$.

Recurrence for number of unfilled message slots

Fix an arbitrary item x, and let $w \leq v$ denote the number of 0 bits of T within x's item set. Let $k \leq m$ denote the number of Increment operations performed on item x performed so far.

First, for convenience define p_w to be the probability that an arbitrary user is able to write to one of the w remaining unfilled slots for the message. From Lemma 6.1, we have

$$p_w = 1 - \frac{(s-u)^{\underline{w}}}{s^{\underline{w}}}, \tag{4}$$

which can be computed in $O(w)$ time. In fact, we pre-compute *all* possible values of p_w with $0 \leq w \leq v$ in $O(v)$ total time.

Now consider the random variable for the number of 0 bits within x's item set after k Increment's on x, if the item set originally had w 0 bits. Define $R_{w,k}$ to be the expected value of this random variable, which can be calculated recursively as follows.

If $w = 0$, then the slots are all filled, and if $k = 0$ then there are no more Increment's, so the number of unfilled slots remains at w. Otherwise, the first Increment will fill an additional slot with probability p_w, leaving $w - 1$ remaining unfilled slots, and otherwise will leave w remaining unfilled slots. This implies the following recurrence relation:

$$R_{w,k} = \begin{cases} 0, & w = 0 \\ w, & k = 0 \\ p_w R_{w-1,k-1} + (1-p_w) R_{w,k-1}, & w, k \geq 1 \end{cases}$$

All values of $R_{w,t}$ with $0 \leq w \leq v$ can be computed in $O(tv)$ time and $O(v)$ space, using a straightforward dynamic programming strategy.

Computing the tipping point

We now show how to compute the tipping point value τ, which is the expected number of filled item slots after t Increments on that item, by summing the $R_{w,t}$

values over all possible values of w based on the number of *other* calls to Increment m.

To this end, define q_w to be the probability that $w \le v$ slots for a given item are unfilled after m total calls to Increment for other items. Because other calls to Increment are for other unrelated items, each one goes to a uniformly-random unfilled slot over the entire size-s table T. Therefore q_w is the same as the probability of a size-m set and a size-v set having intersection size exactly $v - w$. From Lemma 6.1, this is

$$q_w = \frac{m^{\underline{v-w}} \cdot v^{\underline{v-w}} \cdot (s-m)^{\underline{w}}}{s^{\underline{v}} \cdot (v-w)!}.$$

We can pre-compute all values of q_w for $0 \le w \le v$ in total time $O(v)$.

After pre-computing the values of p_w, $R_{w,t}$, and q_w, we can finally express the tipping point τ as a linear combination

$$\tau = v - \sum_{w=0}^{v} q_w R_{w,t}, \tag{5}$$

rounded to the nearest integer.

In total, the computation requires $O(tv)$ time and $O(v)$ space.

6.4.3 Tail Bounds

Next, we prove lower and upper bounds on the probability of filling a single additional item slot during an Increment operation, Lemmas 6.2 and 6.3 respectively. The proofs, which are intricate but not especially surprising, can be found in Section 7.4.1.

In order to make our scheme practically realizable, we state and prove explicit rather than asymptotic results, with all constants specified. These constants in themselves are not particularly meaningful; rather, they represent the tightest values which worked with our proof techniques and the parameter ranges we deemed reasonable for the application in mind.

Lemma 6.2. Let x be an item such that at most τ of x's item slots are filled. If the CCBF parameters s, u, v satisfy $v \ge 7.042652\tau$ and $u \ge 0.5184846\frac{s}{\tau}$, then the probability that a call to Increment(x, C) fills in one more of x's item slots is at least 0.956414.

Lemma 6.3. Let x be any item. If the CCBF parameters s, u, v satisfy $371 \le v \le 0.00386s$ and $u \le 3.65151\frac{s}{v}$, then the probability that a call to Increment(x, C) fills in one more of x's item slots is at most 0.974876.

Now we use the probability upper bound to prove an upper bound on the tipping point τ.

Lemma 6.4. Let s, u, v be CCBF parameters that satisfy the conditions of Lemma 6.3, and suppose m, t are integers such that $s \ge 96m$ and $v \le 7.409t$. Then the tipping point τ, for threshold t and with m total set bits in the table T, is at most $1.0520553t$.

We can now state our main theorems on the accuracy of the CCBF data structure. Consider a call to the predicate function TestCount(x, t), which attempts to determine whether the number of prior Increment calls with the same item x is at least t. Our exact computation of the tipping point $r(t)$ shows that this function always returns the correct answer with at least 50% probability. But of course, so would a random coin flip!

Let k be the *actual* number of calls to Increment(x, C) that have occurred. Then two kinds of errors can occur: a *false positive* if TestCount(x, t) returns true but $k < t$, and a *false negative* if TestCount(x, t) returns false when $k \geq t$. Intuitively, both errors occur with higher likelihood when the true count k is close to t. Our main theorem captures and quantifies this intuition, saying that, ignoring low-order terms, TestCount is accurate to within a 10% margin of error with high probability.

Theorem 6.5. Let n be an upper bound on the total number of calls to Increment, and t be a desired threshold for TestCount. Suppose the parameters s, u, v for a CCBF data structure satisfy the conditions of Lemma 6.2, and furthermore that $v \leq 8t$. If the actual number of calls to Increment(x, C) is at most $t - 2.1\sqrt{\lambda t}$, then the probability TestCount(x, t) gives a false positive is at most $2^{-\lambda}$.

Theorem 6.6. Let n be an upper bound on the total number of calls to Increment, and t be a desired threshold for TestCount. Suppose the parameters s, u, v for a CCBF data structure satisfy the conditions of Lemmas 6.2 and 6.4. If the actual number of calls to Increment(x, C) is at least

$$1.1t + .4\lambda + .7\sqrt{\lambda t}, \tag{6}$$

then the probability TestCount(x, t) gives a false negative is at most $2^{-\lambda}$.

We can easily summarize the various conditions on the parameters as follows:

Corollary 6.7. Let n be a limit on the total number of calls to Increment, and t be a desired threshold satisfying $50 \leq t \leq \frac{n}{20}$. Then by setting the parameters of a CCBF according to $s = 96n$, $v = 7.409t$, and $u = 47.31\frac{n}{t}$, any call to TestCount(x, t) will satisfy the high accuracy assurances of Theorems 6.5 and 6.6.

6.5 Instantiating FACTS

We are now ready to present our construction of FACTS. This construction is based on the collaborative counting Bloom filter (CCBF) data structure presented in Section 6.4 to obliviously count the number of complaints on each message. It uses an underlying EEMS for sending end-to-end encrypted messages between users.

Setup: The setup procedure for FACTS first sets up the underlying end-to-end encrypted messaging system (EEMS). For simplicity, we assume that there is a fixed number c of users using the system. Setup generates all necessary keys for the server S and all c users and distributes the keys. We note that if the messaging system is already setup, FACTS can simply leverage this for communication. Additionally, the server initializes an empty CCBF data structure.

Sending and receiving messages: We now describe how FACTS originates, forwards, and verifies messages. We start our description with an auxiliary protocol $\mathsf{Originate}(A, x)$ between a user A and the server S to originate a new message x. This protocol is used to create an origination tag tag_x containing information about the message and originator. This tag binds the originator's identity A to the message x to enable recovery upon an audit, while keeping A private from receiving users, and keeping the message x private from the server S.

Roughly, this protocol works by having S produce a signature on (a hash of) the message together with the originator's identity. Due to the use of the hash, S produces this signature without learning anything about the message, while the fact that S includes the originator's identity in this signature prevents a malicious originator from including the wrong identity in the message. Moreover, since the tag is bound to the message, this prevents a replay attack where an adversary reuses tags across messages to change the identity of the originator.

Algorithm 26 $\mathsf{Originate}(A, x)$

```
1. To originate a message $x$, the originator $A$ chooses a random salt $r \leftarrow \{0,1\}^\kappa$, computes a salted hash $h = H(r||x)$, and sends $h$ to $S$.
2. $S$ computes an encryption of the sender's identity, $e \leftarrow \mathsf{Enc}_{PK_S}(A)$, and produces signature $\sigma = \mathsf{Sig}_{SK_S}(h||e)$. $S$ sends the tuple $(e, \sigma)$ to $A$.
3. $A$ outputs $\mathsf{tag}_m = (r, e, \sigma)$.
```

Next, we describe the $\mathsf{SendMsg}$ protocol which makes use of the $\mathsf{Originate}$ protocol to send a message x between clients A and B while preserving (encrypted) information about the originator of x. x can either be a newly originated message or a forward of a previously received message. In either case, $\mathsf{SendMsg}$ runs the $\mathsf{Originate}$ protocol to produce a new tag tag'_x on the message x. In the case of a new message, tag'_x is sent along with the message, while in the case of a forward, it is discarded and the message is forwarded along with its original tag instead.

RcvMsg is a non-interactive algorithm that allows a receiving user to verify the tag, tag_x, affiliated with a message x. Specifically, the receiver B verifies the server's signature included in tag_x to make sure that the tag indeed corresponds to x and that the originator id has not been modified. Importantly, B can perform this verification without learning the identity of the originator since the tag contains an encryption of this identity (this ciphertext is what is verified by B).

Complaints and Audit: We now describe how FACTS allows users to complain about received messages and to trigger an audit once enough complaints are

Algorithm 27 $\mathsf{SendMsg}(A, B, \mathsf{tag}_x, x)$

```
1. If $\mathsf{tag}_x = \perp$, then $x$ is a new message $A$ wants to originate. $A$ runs
   $\mathsf{tag}_x \leftarrow \mathsf{Originate}(A, x)$.
2. If $\mathsf{tag}_x \neq \perp$ $x$ is a message that $A$ wants to forward. $A$ runs $\mathsf{tag}'_x \leftarrow$
   $\mathsf{Originate}(A, x)$ and discards the output.
3. $A$ sends $(\mathsf{tag}_x, x)$ to $B$ using the E2E messaging platform's send protocol.
```

Algorithm 28 $\mathsf{RcvMsg}(A, B, \mathsf{tag}_x, x)$

```
1. Parse $\mathsf{tag}_x$ as $\mathsf{tag}_x = (r, e, \sigma)$
2. Compute $h = H(r||x)$
3. Run $\mathsf{Ver}_{PK_S}(\sigma, (h||e))$ to check that $\sigma$ is a valid signature by the server on
   $(h||e)$. If not, then discard the received message.
```

registered on a message. For these methods we make extensive use of a CCBF data structure for (approximately) counting complaints and detecting when a threshold of complaints has been reached.

The $\mathsf{Complain}$ protocol is used by a receiving user to issue a complaint on a received message (tag_x, x). We assume that prior to issuing a complaint the user verifies that tag_x is valid using the RcvMsg protocol, and thus will only consider the case of valid tags. To issue a complaint on (tag_x, x), the user C calls CCBF.$\mathsf{Increment}(\mathsf{tag}_x, C)$. As described in Section 6.4, this runs a protocol with the server in which the user (eventually) sends the location of a bit to flip to 1 to increment the CCBF count for the message x. To prevent malicious adversaries from flooding FACTS with complaints, we enforce a limit of L complaints per user per epoch. Note that since the server knows the identities of complaining users, he can easily enforce this restriction.

Two important observations are in order here. First, we use tag_x rather than the message x as the item to increment in the CCBF. The reason for this is that the tag is unpredictable to an adversary who has not received the message x through FACTS (even if $\mathcal{A}$ knows x). Second, we note that the CCBF.$\mathsf{Increment}$ procedure is inherently sequential. It requires that the CCBF table T be locked for the duration of the $\mathsf{Increment}$ call to prevent race condition and to maintain obliviousness (see Section 6.4 for discussion). This means that only one user can run this procedure at a time. Thus, we focus on making this procedure as cheap as possible to minimize the impact of this bottleneck. In case multiple clients call $\mathsf{Complain}$ at overlapping times, the server can queue these complaints and process them one at a time.

The Audit protocol checks whether a threshold of complaints has been reached for a given message x and, if so, triggers an audit of this message. This protocol works by using the CCBF.$\mathsf{TestCount}$ protocol to check whether the threshold t

Algorithm 29 $\mathsf{Complain}(C, \mathsf{tag}_x, x)$

```
1. Parse $\mathsf{tag}_x$ as $\mathsf{tag}_x = (r, e, \sigma)$
2. Call CCBF.$\mathsf{Increment}(C, \mathsf{tag}_x)$
```

of complaints has been reached on this message. If this returns True, then the user simply sends (tag_x, x) to the server who first checks the validity of the tag, and then if it's valid, decrypts the corresponding part of the tag to recover the identity of the message originator.

An important observation is that the CCBF.TestCount operation is read-only and thus does not need to block. Thus, unlike the Complain command, many clients can execute the Audit command in parallel.

Algorithm 30 $\mathsf{Audit}(C, \mathsf{tag}_x, x)$

```
1. Parse $\mathsf{tag}_x$ as $\mathsf{tag}_x = (r, e, \sigma)$
2. Call CCBF.$\mathsf{TestCount}(\mathsf{tag}_x, t)$.
3. If TestCount returns True, $x$ sends $(\mathsf{tag}_x, x)$ to $S$
4. $S$ verifies that the tag is valid by checking the $\sigma$ is a valid signature on
   $h||e$ where $h = H(r||x)$.
5. If so, $S$ recovers the identity $(A)$ of the originator by computing $A =$
   $\mathsf{Dec}_{SK_S}(e)$.
```

We note that Audit allows the server to learn the message x and the originator A. We do not specify what the server does upon learning this information, as that is specific to a particular use of FACTS. One possible option is for the server to review x to see if it is truly a malicious message, and if so, block the user A from sending further messages. However, this decision is orthogonal to the FACTS scheme and we do not prescribe a particular action here.

6.6 Experimental Evaluations

In this section, we empirically evaluate the accuracy and performance of FACTS. We perform two sets of experiments. The first, measures the error in terms of number of complaints above or below the threshold as a function of the total number of complaints. The second, measures the performance overhead for messaging and complaint as a function of the threshold.

6.6.1 Experimental parameters

For our experiments, we set the maximum number of complaints per epoch $n = 1,000,000$. If we consider an epoch of one day, this results in approximately

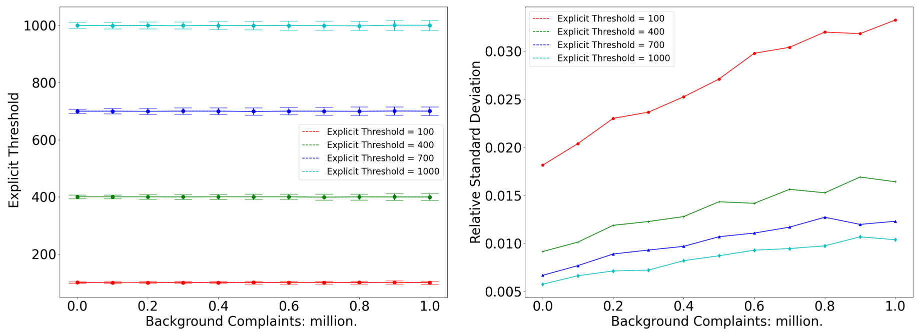


Figure 9: The (left) Mean and (right) Relative Standard Deviation of Experimental Explicit Threshold vs. The Number of Background Complaints.

11.6 complaints per second. To understand accuracy and efficiency of FACTS, we measure them for a range of thresholds $100 \le t \le 1000$. With these fixed, we set the remaining parameters according to Corollary 6.7. In particular, we set the server's storage $s = 96n = 12MB$. The user set size u varies from (approximately) 47,000 to 470,000 bits, while the message set size v goes from (approximately) 740 to 7400.

6.6.2 Accuracy and stability

To measure the accuracy of FACTS, we observe the actual number of complaints necessary to cause an audit on a single message as a function of the background noise (i.e., total complaints on other messages). We calculate both the mean and the standard deviation of this value to capture the accuracy and stability of the complaint mechanism. To get a statistically meaningful estimate of these, our experiments run 1000 iterations of each parameter configuration.

The results of our experiments are presented in Figure 9. The left side of this figure shows the mean number of complaints to trigger an audit for a given threshold t. As can be seen from the error bars, the absolute errors in number of complaints is quite small, with a maximum deviation of about 10 complaints at a threshold of 1000. Not surprisingly, we see that this error increases as the background noise increases, but the mean number of complaints remains remarkably steady at the desired value. The right side of Figure 9 shows the relative standard deviation of the number of complaints as a function of background noise. From this graph we can see that the relative error is only a few percent, with a maximum relative error of about 3.5%. Not surprisingly, the threshold 100 measurement incurs the highest relative error because the noise is a much higher ratio when compared to the threshold. These experiments suggest that FACTS achieves good accuracy for a wide variety of threshold and background noise.

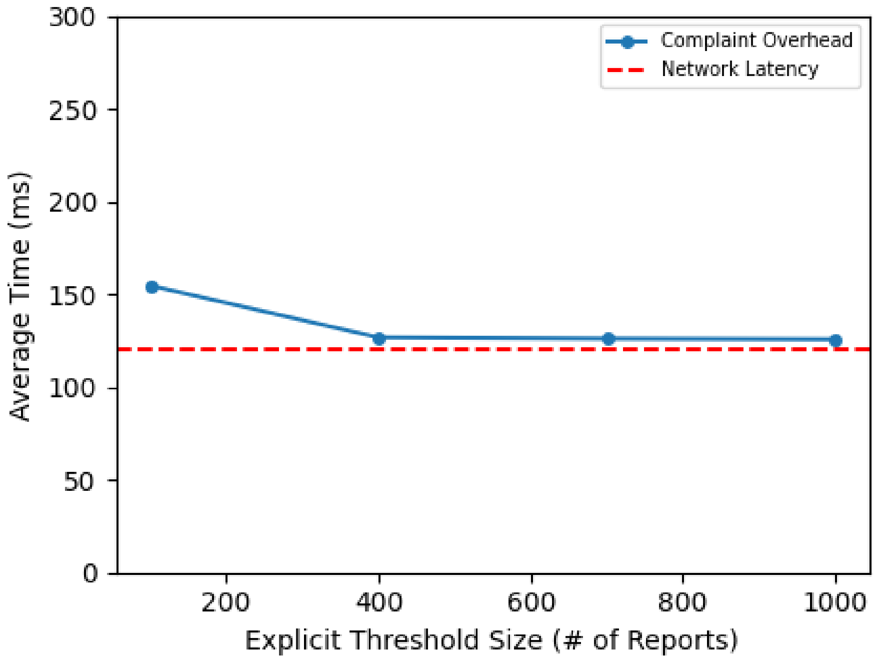


Figure 10: Average complaint time as a function of the threshold with $n = 1,000,000$ complaints per epoch. Complaint time is measured as the average of 100 samples and has a variance of less than 5ms. Network latency shows the minimum latency required to transmit 3 sequential messages over the network, a lower bound on complaint time.

6.6.3 Performance overhead

Our next set of experiments measures the performance overhead of FACTS as a function of the threshold to start an audit. Specifically, we measure the overhead of sending a message using FACTS, and the cost of issuing a complaint. We note that for the message sending cost, we do not measure the cost of the EEMS communication, instead only measuring the added overhead due to FACTS.

For these experiments, we implemented both the client and server using the Rust programming language. We used SHA-3 for a hash function, and for encryption and signatures we used Rust's ring library [166] implementation of OpenSSL's ChaCha20-Poly1305 protocol and Ed25519 respectively. To instantiate the CCBF, we used a simple library bitvec [150] that allows memory to be bit addressed, rather than byte addressed, which gains us a quick, compact way to store the CCBF data structure.

To simulate network overhead, we implemented a simple web server and client, which communicated over a (simulated) 8 Mbps network with a latency of 80ms, using TLS 1.3. Since we are only measuring the overheads of FACTS over the underlying EEMS, our measurements did not include the time to send the message over the EEMS, nor the time to establish the TLS connection. All experiments were run on a 4.7Ghz Intel Core i7 with 16GB of RAM, with a sample size of 100 for each metric. As in the accuracy experiments, we set $n = 1,000,000$ and threshold varying from 100 to 1000, with the remaining parameters determined by Corollary 6.7.

For our measurement of message origination we looked at the cost of originating and sending a message of size 2MB. Creating and sending such a message with the encrypted hash and identity took 98ms, which indicates that the major bottleneck in this process is the 80ms network latency. We see then that when a user wishes to forward a message, they will still call $\mathsf{Originate}(A, x)$, but then forward the original message whereas in an EEMS this would just require a forward. Thus, the overhead of FACTS on a forward is slightly less than 100ms.

Figure 10 shows our measurements of the time to issue a complaint as a function of the audit threshold. The time for this is dominated by the time to retrieve the user set (i.e., the bits that the user can write) from the server. Since the size of this set $u = O(n/t)$, this time grows inversely with the threshold t. Thus, as the threshold increases, the total complaint time decreases very quickly, going down to essentially just the network latency when $t \geq 400$.

These experiments show that both the (added) cost of sending messages and the cost of complaints (for sufficiently large t) are dominated by the networking costs. Thus, as long as the latency of the network is reasonable, FACTS can scale to millions of complaints per day.

6.7 Security of FACTS

In this section we analyze the security of FACTS. We provide security definitions capturing the privacy and integrity guarantees provided by FACTS and prove that our protocols described in Section 6.5 achieve these definitions.

6.7.1 Adversary Model

We consider two different types of adversaries against FACTS. The first is an honest-but-curious server S. Such a server may also collude with some of the users. However, all such users, as well as the server, will follow the protocol. This adversary class models what the FACTS server learns in running the system, so we want to limit what the server learns. However, we have to assume that the server acts honestly, as a malicious server can fully break the integrity and availability of FACTS. For example, since the server produces the signatures binding originators to messages, a malicious adversary with knowledge of this key could arbitrarily assign originators by forging this signature.

We also consider a second type of adversary controlling a group of malicious users who do not collude with the server. Such users may want to violate the confidentiality of FACTS by learning extra information about messages or complaints, beyond what they learn through the messages they validly receive. Or, they may want to break the integrity of the complaint and audit mechanism of FACTS to blame innocent parties for audited messages, or to delay or speed-up the auditing of targeted messages. This models an external adversary, say a malicious company or government, who may want to distribute fake information without being audited or may want to block certain information or users from the system.

6.7.2 Privacy

We begin by looking at the privacy guarantees provided by FACTS.

Privacy vs. Server: We first give a definition for privacy against a semi-honest server who may also collude with some semi-honest users. In this setting we aim to argue that unless a message is audited or is received by an adversarial user, the server learns no information about the message or the complaints on the message. In particular, the server should not be able to tell whether any message is a new message or a forward and how many, if any, complaints this message may have. In fact, the only thing that the server learns is the *metadata* of who is sending messages to whom and who is issuing complaints, but not anything more.

Specifically, we propose a real-or-random style definition to capture privacy against the server. This definition captures the fact that the view of the server (and colluding users) until a message is audited or received by a colluding user just consist of random values, and thus is independent of the messages and complaints.

Concretely, we define the following game between an adversary $\mathcal{A}$ controlling the server (and possibly some colluding users) and a challenger.

$\underline{\mathsf{Game}^{\mathsf{server\text{-}privacy}}_{\mathsf{EEMS}}(\mathcal{A})\text{:}}$

1. The challenger runs $\mathsf{Setup}(c)$ to set up the EEMS with c users. He hands all keys corresponding to corrupted parties to $\mathcal{A}$

2. $\mathcal{A}$ chooses a sequence of messages $((\text{send}, A_0, B_0, \mathsf{tag}_{x_0}, x_0), \ldots, (\text{send}, A_\ell, B_\ell, \mathsf{tag}_{x_\ell}, x_\ell))$[12], and a sequence of complaints $((\text{complain}, C_0, \mathsf{tag}_{x_0^c}, x_0^c), \ldots, (\text{complain}, C_{\ell'}, \mathsf{tag}_{x_{\ell'}^c})$ and interleaves them arbitrarily. We require that none of the sending users (A_i), receiving users (B_i), or complainers (C_i) are controlled by $\mathcal{A}$.

3. The challenger chooses $b \leftarrow \{0, 1\}$ and does the following:

   (a) If $b = 0$, Run the $\mathsf{SendMsg}$ and $\mathsf{Complain}$ protocols with inputs supplied by $\mathcal{A}$, giving $\mathcal{A}$ the resulting server view.

   (b) If $b = 1$,

   - for each $\mathsf{SendMsg}$ command, choose $r \leftarrow \{0, 1\}^\kappa$ and send this to S. Choose $x' \leftarrow \{0, 1\}^{|x|+|\mathsf{tag}_x|}$ and send x' from A_i to B_i using EEMS.send.
   - The challenger maintains a set USED $\subseteq [s]$[13]. For each $\mathsf{Complain}$ command, the challenger chooses $ind \leftarrow [s] \setminus \text{USED}$, sends ind from u_i to S, and adds ind to USED.

4. $\mathcal{A}$ outputs a bit b'

5. We say that $\mathcal{A}$ has advantage

$$\mathsf{Adv}_{\mathsf{EEMS}}^{\mathsf{server\text{-}privacy}}(\mathcal{A}) = |\Pr[b = b'] - 1/2|.$$

Definition 6.8 (Privacy vs. Server)**.** A FACTS scheme is *private against a semi-honest server* if the adversary has a negligible advantage in the game above $\mathsf{Adv}_{\mathsf{EEMS}}^{\mathsf{server\text{-}privacy}}(\mathcal{A}) \leq \mathsf{negl}(\kappa)$

Theorem 6.9. FACTS is private against a semi-honest server

Proof sketch. First, consider the server's view on a $\mathsf{SendMsg}$ command. This view consists of a message $h = H(r||m)$ for $r \leftarrow \{0, 1\}^\kappa$ and the leakage from EEMS.send, i.e., the identities A and B, as well as $|(\mathsf{tag}_x, x)|$. Since the challenger uses the same sender, receiver, and message length, the only thing left to prove is that h is indistinguishable from random. Since r is chosen uniformly at random, and H is a random oracle, $H(r||m)$ is uniformly random to $\mathcal{A}$ unless $\mathcal{A}$ queries $H(r||m)$. However, since $\mathcal{A}$ makes at most $\mathsf{poly}(\kappa)$ queries to H, the probability that he makes this query is at most $\mathsf{poly}(\kappa)/2^\kappa \leq \mathsf{negl}(\kappa)$.

Next, we consider the $\mathsf{Complain}$ commands. The server's view on a complaint consists of the complainer's ID C and an index in the CCBF to flip to 1. In a real execution of $\mathsf{Complain}$, this index is chosen at random from the set $S_C \cap V_x$ where $S_C = \{i \in U_C \mid T[i] = 0\}$ and V_x is the list of item locations for x.[14] However, since U_C and V_x are chosen at random, we can equivalently sample a

[12]We note that since $S \in \mathcal{A}$, $\mathcal{A}$ can produce valid-looking tags for each of these messages by producing the necessary signatures.

[13]Recall that s is the size of the CCBF bit vector T

[14]Technically, the item used in the CCBF is the tag tag_x, but we use x here for ease of notation.

random 0-index in the bit vector T and then choose U_C and V_x conditioned on them containing this location. Hence the location sent to the server is uniformly random unless $\mathcal{A}$ makes the corresponding H query, which only happens with $\mathsf{negl}(\kappa)$ probability. ■

The above theorem states that, beyond the meta-data of who sent a message to whom and who has sent complaints and when, FACTS reveals no information about messages and complaints to a semi-honest server until an audit occurs (or a malicious user receives a message). Moreover, the view of the server is completely random when conditioned on the meta-data. Now, suppose that a message x is audited (or is received by an adversary-controlled user). When this happens, the adversary learns the tag and message (tag_x, x). This enables $\mathcal{A}$ to learn the identity of the originator (by decrypting it from tag_x) and to learn the entire history of this message, i.e., the transmission and complaint history of x. However, since the server's view of all other messages is indistinguishable from independent random strings (modulo the meta-data), the adversary does not learn anything more about these messages as a result of an audit on x.

Privacy vs. Users We now proceed to analyze security of our protocol against (possibly malicious) users that are not colluding with the server. This models the case of a third party adversary that tries to learn information about the messages and complaints in FACTS. Here, we no longer assume that a message x is never received by a malicious user and thus we cannot use a real-or-random style definition as before. Instead, we argue that a user cannot distinguish a new message from a forwarded message unless another corrupted user has previously seen that message. This also shows that a malicious user cannot learn the identity of the message originator. Since users do not receive any communication on complaints, we only consider message privacy here.

Concretely, we define the following game between an adversary $\mathcal{A}$ controlling a set of users, and a challenger.

$\mathsf{Game}^{\mathsf{user\text{-}privacy}}_{\mathsf{EEMS}}(\mathcal{A})$:

1. The challenger runs $\mathsf{Setup}(c)$ to set up the EEMS with c users and gives all key material for the corrupted users to $\mathcal{A}$. Let $B \in \mathcal{A}$ be a user controlled by the adversary.
2. $\mathcal{A}$ chooses messages x, x' s.t. $|x| = |x'|$ and honest users $O, A \notin \mathcal{A}$
3. The challenger chooses $b \mid \{0,1\}$ and does the following:
   (a) If $b = 0$, the challenger runs $\mathsf{SendMsg}(O, A, \perp, x')$ and $\mathsf{SendMsg}(A, B, \perp, x)$ with $\mathcal{A}$ receiving the view of B.
   (b) If $b = 1$, the challenger runs $\mathsf{SendMsg}(O, A, \perp, x)$ and $\mathsf{SendMsg}(A, B, \mathsf{tag}_x, x)$ (where tag_x is the tag received by A from O).
4. $\mathcal{A}$ outputs a bit b'

5. We say that $\mathcal{A}$ has advantage

$$\mathsf{Adv}^{\mathsf{user\text{-}privacy}}_{\mathsf{EEMS}}(\mathcal{A}) = |\Pr[b = b'] - 1/2|.$$

Definition 6.10 (User privacy)**.** A FACTS scheme achieves *privacy against malicious users* if the adversary has a negligible advantage in the game above $\mathsf{Adv}^{\mathsf{user\text{-}privacy}}_{\mathsf{EEMS}}(\mathcal{A}) \leq \mathsf{negl}(\kappa)$

Theorem 6.11. FACTS achieves privacy against malicious users.

Proof sketch. The view of B on an execution of $\mathsf{SendMsg}(\cdot, B, \mathsf{tag}_x, x)$ consists of the received message and tag (tag_x, x) where $\mathsf{tag}_x = (r, e, \sigma)$. Since e is a semantically secure encryption of the identity of the originator, $\mathcal{A}$ cannot distinguish between the case when $e = \mathsf{Enc}(A)$ (when $b = 0$) and the case when $e = \mathsf{Enc}(O)$ (when $b = 1$) except with advantage negligible in κ. Additionally, since tag_x is generated identically both when $b = 0$ and $b = 1$ except for this change in e, this means that tag_x does not help $\mathcal{A}$ distinguish between these two cases. ∎

6.7.3 Integrity

We now turn to the integrity guarantees provided by FACTS. We aim for a few different notions of integrity to show that malicious users cannot interfere with the complaint and audit process. First, no adversary controlling a subset of the users should be able to frame an honest user as the originator of an audited message he did not originate. Second, an adversary controlling a subset of the users should not be able to significantly delay the audit of a malicious message. In particular, such an adversary should not be able to prevent a malicious message sent by one of his users from being audited. Finally, an adversary controlling a small set of users should not be able to significantly speed up the auditing of a targeted message. In particular, such an adversary should not be able to cause an audit without complaints from some honest users.

We begin by defining the following game between a challenger and an adversary $\mathcal{A}$ controlling a subset of the users to capture the inability of an adversary to forge a valid tag that it has not seen before.

$\underline{\mathsf{Game}^{\mathsf{unforgeability}}_{\mathsf{EEMS}}(\mathcal{A})\text{:}}$

1. The challenger runs $\mathsf{Setup}(c)$ to set up the EEMS with c users and gives all key material for the corrupted users to $\mathcal{A}$.

2. $\mathcal{A}$ requests $\mathsf{SendMsg}$ operations on messages of its choice both from honest and corrupted users. ($\mathcal{A}$ is given the view of corrupted users in all these executions consisting of (tag_x, x).)

3. $\mathcal{A}$ outputs a tag, message pair (tag_y, y)

4. We say that $\mathcal{A}$ WINS if tag_y is a valid tag for message y with originator $O \notin \mathcal{A}$, and there has not been a prior command $\mathsf{SendMsg}(O, \cdot, \perp, y)$.

Definition 6.12 (No framing)**.** We say that a FACTS scheme disallows *framing* if for any PPT $\mathcal{A}$, $\mathcal{A}$ WINS in the above game with probability at most $\mathsf{negl}(\kappa)$.

Theorem 6.13. The FACTS scheme is unforgeable.

Proof sketch. A valid tag tag_y with originator O consists of $\mathsf{tag}_y = (r, e, \sigma)$ where r is a random seed s.t. $H(r||y) = h$, $e = \mathsf{Enc}_{PK_S}(O)$, and $\sigma = \mathsf{Sig}_{SK_S}(h||e)$. Thus, to frame O, $\mathcal{A}$ needs to produce a valid signature on $h||\mathsf{Enc}(O)$. $\mathcal{A}$ can observe tags from polynomially many messages originated by $\emptyset$, but except with probability negligible in κ none of them will have the same value h. Thus, by the unforgeability of Sig, $\mathcal{A}$ cannot produce the necessary signature except with probability negligible in κ. ■

Next, we give a definition that captures the ability of an adversary controlling a subset of the users to delay the audit of a particular message. Our goal is to show that the adversary cannot protect a malicious message from being audited.

Specifically, we define the following game,

$\underline{\mathsf{Game}^{\mathsf{no-delay}}_{\mathsf{EEMS}}(\mathcal{A})\text{:}}$

1. The challenger runs Setup to set up the EEMS with c users and gives all key material for the corrupted users to $\mathcal{A}$.
2. $\mathcal{A}$ issues a single $\mathsf{SendMsg}(A, B, x)$ command with $A \in \mathcal{A}$ to produce tag_x
3. $\mathcal{A}$ outputs a list of $\mathsf{Complain}$ commands with at most n total complaints, of which at least ℓ are complaints on tag_x.
4. The challenger runs the specified complaint commands, and then runs $\mathsf{Audit}(A, \mathsf{tag}_x, x)$
5. We say that $\mathcal{A}$ WINS if this audit is not successful (i.e., the audit threshold is not reached).

Definition 6.14 (No delay)**.** We say that a FACTS scheme is *ℓ-audit delay resilient* for integer $\ell < n$ if for any PPT $\mathcal{A}$, $\mathcal{A}$ WINS in the above game with probability at most $\mathsf{negl}(\kappa)$.

Theorem 6.15. The FACTS scheme is ℓ-audit delay resilient for any $\ell \geq 1.1t + .4\kappa + .7\sqrt{\kappa t}$.

Proof sketch. This follows immediately from Theorem 6.6 ■

Next, we define the following game to capture the ability of a small number of malicious users to cause the audit of some message. Importantly, this definition also captures the case where malicious users try to audit an honest message (on which there are no complaints by honest users). Specifically, the following game is between an adversary $\mathcal{A}$ corrupting at most ℓ users and a challenger

$\underline{\mathsf{Game}^{\mathsf{no-speedup}}_{\mathsf{EEMS}}(\mathcal{A})\text{:}}$

1. The challenger runs Setup to set up the EEMS with c users and gives all key material for the ℓ corrupted users to $\mathcal{A}$.
2. The challenger runs a single $\mathsf{SendMsg}(A, B, x)$ command for $A \notin \mathcal{A}$ and $B \in \mathcal{A}$.
3. $\mathcal{A}$ may issue at most L $\mathsf{Complain}$ commands per each user he controls.[15]
4. The challenger runs the specified $\mathsf{Complain}$ commands, and then runs $\mathsf{Audit}(\cdot, \mathsf{tag}_x, x)$.
5. We say that $\mathcal{A}$ WINS if this audit is successful.

Definition 6.16 (No speed up)**.** We say that a FACTS scheme is *ℓ-party audit speed-up resilient* if for any PPT $\mathcal{A}$ controlling at most ℓ users, $\mathcal{A}$ WINS in the above game with probability at most $\mathsf{negl}(\kappa)$.

Theorem 6.17. The FACTS scheme is ℓ-party audit speed-up resilient for $\ell \leq (t - 2.1\sqrt{\lambda t})/L$.

Proof sketch. This follows immediately from Theorem 6.5 because each user $\in \mathcal{A}$ makes at most L complaints. ■

6.8 Alternative FACTS

In this section we describe several optimizations or enhancements to the basic FACTS protocol.

Revealing even less during audits. Recall that the FACTS system we presented reveals two things to the server (or an auditor) after the threshold of complaints has been reached: the user id of the message's originator, and the contents of the message itself. Indeed, one of our motivations was to avoid revealing the entire path or tree of message forwarding as in prior work [177].

However, in some environments, even this may be too much to release to a service provider that could be, for instance, compromised or influenced by an oppressive regime. An advantage of FACTS is that the system for tallying complaints actually does not *require* this information in order to function properly. Here we briefly sketch simple modifications to the scheme to achieve this additional hiding, with a note of caution that we have not analyzed the formal security under these variants.

Hiding the originator's identity entails omitting the encrypted user id from the origination protocol. To do this, the server's signature σ of the message hash and sender identity should be replaced with a *blind signature* of the message hash only. In this way, a later audit which reveals the (unblinded) signature will not reveal anything about the originator's identity. The disadvantage of course would be that there is no way for the system to identify and penalize users who regularly submit fake news to the platform.

[15]Recall that FACTS enforces a limit of L complaints per user per epoch.

To hide the message contents, these would simply be omitted from what is sent to the auditor once the threshold is reached. In this case, the notion of "audit" may be understood to be simply confirming that *some* message (with the given hash) has passed the threshold of complaints, and publishing the hash to all users as *potentially* fake news that has received a large number of complaints. The client software could easily be configured to flag such messages as they are received afterwards, without ever revealing to the server any contents or recipients of such message. Here the disadvantage is obviously that no third-party auditing or fact-checking is possible, raising the possibility of false positives in which messages are flagged.

Throttling complaints. The FACTS system and underlying CCBF data structure assume a global limit n on the number of complaints per epoch, but do not require any per-user limit besides the natural limit of u, the size of the user set.

However, there is some potential for abuse by users who issue many complaints in a single epoch: they may attempt to "attack" another known message by issuing multiple complaints that set bits in that message's user set; they may collude with others and attempt to go over the total per-epoch limit of n complaints; or they may simply attempt a denial-of-service attack to prevent other complaints from being issued.

An simple solution to these problems is to apply a limit $\ll u$ on the maximum number of complaints per user per epoch. This is easy for the server to apply, since users are authenticated during the $\mathsf{Complain}$ protocol. More nuanced limits based on a user's reputation or longevity on the platform could also be applied.

Users with a small "quota" of allowed complaints per epoch could even be encouraged to participate initially in the complaint process by forwarding questionable content to a trusted reputable user on the system, who would then presumably apply their own judgment and possible issue a complaint in turn. This idea is aligned with many existing content moderation settings on (unencrypted) social media platforms.

Handling epoch rollover. As described, the FACTS system resets all counters at the end of a single epoch. However, this may mean that if a "fake news" message is first detected towards the end of an epoch, the complaints for this message may get split between the current and next epochs and thus fail to trigger an audit in either epoch.

A potential solution to this problem is to always run two epochs concurrently, where each epoch lasts for time t, and the epochs start times are $t/2$ apart. Users complain in both of the epochs, and an audit occurs if the number of complaints in either epoch exceed the threshold. This way, regardless when a "fake news" message is first detected, there will be an epoch with at least $t/2$ time left to accumulate complaints. Since we assume that fake news messages are ones that are received and complained on by many users, and that users are likely to complain shortly after first receiving a message, this provides enough time for a threshold of complaints to be reached.

Regional complaint servers. The most significant performance bottleneck in

FACTS is the necessary global lock on the table T while a single user is waiting to download their user set U_C and reply with their complaint index. Even though the communication size is quite small for practical settings, the inherent latency across global communications networks may impose a challenge.

For example, if many complaining users have a round-trip latency of more than 200ms, then the global complaint rate among all users cannot be higher than 5 complaints per second, or some 432,000 complaints per day, regardless of any parameter settings or chosen epoch length.

One possible solution for a large-scale platform facing this issue would be to allow multiple local complaint servers, each with their own CCBF table T, to independently operate and accumulate complaints per messages. This makes sense, as most targeted misinformation content is local to a given country or region, and it would still be possible for each regional server to share audited message information with others in order to prevent spread of viral false content between regions.

Third-party audits. While many messaging and social media platforms currently employ their own "in-house" teams for content moderation, there have been some attempts at separating the role of the server from that of auditor.

From a protocol standpoint, we can imagine a separate Server and Auditor: the former is semi-honest, handles the encrypted messaging system and maintains the public CCBF table T. The Auditor is fully honest and non-colluding, but computationally limited; intuitively, the third-party Auditor should only be involved once a messaged has passed the desired threshold of complaints.

The FACTS system supports this option easily with the need for any additional cryptographic setup during origination. Because the CCBF table T is globally shared among all users as well as the Auditor, any complaining user who computes TestCount on their own to see that the probabilistic threshold has been surpassed, can then forward their complaint (i.e., the opened message) directly to the Auditor. Being fully honest, the Auditor may hold a copy of the decryption key from origination and use this to determine what kind of action may be necessary (such as suspending the originating user's account, flagging the message, etc.).

While it doesn't appear idea imposes any additional interesting challenges from a cryptographic standpoint, it could be useful for some kinds of messaging platforms.

Hiding message metadata. Our FACTS system is certainly no more private than the underlying EEMS which is being used to actually pass messages between users. In our analysis, we explicitly assumed that the EEMS leaks metadata on the sender and recipient of each message, but not the contents.

However, some existing EEMS attempt to also obscure this metadata in transmitting messages, so that the server does not learn both sender and recipient of any message. This can trivially be accomplished by foregoing a central server and doing peer-to-peer communication (note that FACTS may still be useful as a central complaint repository); or using more sophisticated cryptography to hide metadata [35, 51, 175].

Of particular interest for us is the recently deployed *sealed sender* mechanism on the popular Signal platform [130]. The goal in this case is to obscure the sender, but not the recipient, from the server handling the actual message transmission. We note that this concept plays particularly well with FACTS, as the additional leakage in our protocol of the identity of each complaining user, can be presumably correlated via timings with the receipt of some message, but this is exactly what is revealed under sealed sender already! Both systems thus work to still hide message sender and originator identities (at least until an audit is performed).

However, note that recent work [132] has shown that some timing attacks are still possible under sealed sender, and the same attacks would apply just as well to FACTS. But the solutions proposed in [132] might also be deployed alongside FACTS to prevent such leakage; we leave the investigation of this question for future work.

6.9 Related Work

Message Franking: The most common approach today for reporting malicious messages in encrypted messaging systems is *message franking* [57, 95, 176]. Message franking allows a recipient to prove the identity of the sender of a malicious message. However, message franking is focused on identifying the last sender of a message, whereas we are interested in identifying the originator. Moreover, message franking does not provide any threshold-type guarantees to prevent unmasking of senders given only one (or a few) complaints.

Oblivious RAM (ORAM): Oblivious Random-Access Memory (ORAM) [87, 90, 143] allows a client to obliviously access encrypted memory stored on a server without leaking the access pattern to the server. The standard ORAM definition assumes a single user with full control over the database. While some important progress has been made on multi-client ORAM protocols [38, 102, 131], these solutions are still not scalable to millions of malicious users as would be needed for our application.

Oblivious Counters and Oblivious Data-Structures: Like CCBF, oblivious counters [86, 120] build counters that can be stored and incremented without revealing the value of the counter. However, these techniques focus on exact counting, and do not provide efficient ways for storing large numbers of counters, as needed for our applications. More generally, oblivious data-structures, e.g. [122, 161, 180] construct higher-level data structures such as heaps, trees, etc. to enable oblivious operations over encrypted data. However, these largely focus on higher-level applications and do not provide the compression achieved by CCBF.

Privacy-Preserving Sketching: CCBF can be viewed as a small data structure (a *sketch*) for storing the counts of complaints on a large set of messages. There has indeed been a lot of recent interest (e.g., [8, 45, 68, 79, 112, 136, 178]) in private sketching algorithms for cardinality estimation, frequency measurement, and other approximations. However, these works generally focus on a multi-party setting, with multiple parties running secure computation to evaluate the statistic in question. Since our goal was to restrict ourselves to user-server communication only, such techniques do not seem applicable to our setting.

7 Omitted Proofs

7.1 Secure Search

7.1.1 Proof of Lemma 3.4

Lemma 7.1 (4.1)**.** Consider a Bloom filter with false positive rate $\frac{1}{m}$, where m is an arbitrary positive integer. Suppose at most m $\mathsf{BF.Check}$ operations are performed in the BF. Then, for any $\delta > 0$, we have:

$$\Pr[\#\text{ false positives} \geq 1+\delta] \leq \frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}.$$

Proof. Let α_i be the ith item that is checked through $\mathsf{BF.Check}$. That is, we consider a sequence of

$$\mathsf{BF.Check}(\alpha_1), \ldots, \mathsf{BF.Check}(\alpha_m),$$

where α_i is an arbitrary item. Since we wish to upper bound the false positives (i.e., we don't care about true positives), it suffices to consider the case that for every i, $\alpha_i \notin \mathsf{BF}$ (i.e, α_i has not been inserted in the BF) as this maximizes the number of possible false positives.

Let $X_1, \ldots, X_m$ be independent Bernoulli random variables with $\Pr[X_i = 1] = 1/m$. Since the BF false positive rate is assumed to be $1/m$, we have for all i,

$$\Pr[\mathsf{BF.Check}(\alpha_i) = 1] = \Pr[\text{query } i \text{ is a false positive}] \leq 1/m.$$

Thus, we can bound the number of false positives by $\sum_{i=1}^{m} X_i$.

Now, let $\mu := \mathbf{Exp}[\sum X_i] = m \cdot \frac{1}{m} = 1$. By applying the Chernoff bound with $\mu = 1$, we have:

$$\Pr\left[\sum_{i=1}^{m} X_i \geq 1+\delta\right] \leq \frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}.$$

■

7.1.2 Proof of Theorem 3.8

Theorem 7.2 (6.3)**.** Given an FHE scheme, and a (n, s, c, f_p)-CODE scheme over domain D in the random oracle model, the construction in Algorithm 8 yields a (ℓ, f_p)-secure search scheme for records in domain D in the random oracle model, where $\ell = \frac{c(s)\cdot \ell_c}{s}$, ℓ_c is the length of an FHE ciphertext with plaintext space D, and s is the number of matching records.

Proof. We begin by proving that the adversary cannot distinguish between two different queries. The adversary chooses a database x and two queries q^0, q^1, with the promise that $s = \sum_{i=1}^{n} q^0(x_i) = \sum_{i=1}^{n} q^1(x_i)$.

The entire view of the adversary during the experiment can be reconstructed efficiently given (1) the encrypted database $[\![x]\!]$, (2) the encrypted query $[\![q]\!]$, (3) the decrypted value of s.

We note that the CODE scheme may return either more than s values to the client (in case of a false positive) or less than s values (in case decoding fails), but both of these occur with probability at most $\mathsf{negl}(\kappa)$ and thus we can ignore them in the following.

Since the value of s is the same for q_0 and q_1, the only thing that changes in the view of the adversary when switching from $b = 0$ to $b = 1$ is the encrypted query $\tilde{q_b}$. Therefore, the adversary guesses b with negligible advantage by the IND-CPA security of the FHE scheme.

The proof that the adversary cannot distinguish between the same query applied to two different databases follows nearly identically. ■

7.2 Secure Sampling

7.2.1 Definitions

We assume that readers are familiar with security notions of standard cryptographic primitives [119] and formal definitions of a protocol securely realizing an ideal functionality (cf. [71]).

Ideal functionality $\mathcal{F}_{2PC}$
The ideal functionality works as follows:

$\mathcal{F}_{2PC}$: Ideal functionality for evaluating two-party circuits.

The functionality has the following parameter:

- Two party binary circuits $C_1(\cdot,\cdot)$ and $C_2(\cdot,\cdot)$.

The functionality proceeds as follows:

1. Receive inputs x_1 and x_2 from P_1 and P_2 respectively.
2. Send $C_1(x_1, x_2)$ to P_1 and $C_2(x_1, x_2)$ to P_2.

It is well know that Yao's protocol securely realizes F_{2PC} in the semi-honest security setting with a constant round and $O(|C_1| + |C_2|)$ communication [128].

Differential privacy
We say that two vectors $\mathbf{d} = (d_1, d_2, \ldots)$ and $\mathbf{d}' = (d'_1, d'_2, \ldots)$ are *neighboring* if they have the same length, and there exists only one index i s.t. $d_i \neq d'_i$.

Definition 7.3 (Differential privacy with a trusted curator [61, 64])**.** A mechanism $\mathcal{M}$ satisfies (ϵ, δ)-differential privacy if for all neighboring data sets $\mathbf{d}$ and $\mathbf{d}'$, and all sets $S \subseteq Range(\mathcal{M})$

$$\Pr[\mathcal{M}(\mathbf{d}) \in S] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}(\mathbf{d}') \in S] + \delta$$

Differential privacy in the two-party setting. Our presentation here follows the similar definitions given in prior work [18, 45, 162]. For a two-party protocol Π and an input $(\mathbf{d}_1, \mathbf{d}_2)$, we let $\Pi(\mathbf{d}_1, \mathbf{d}_2)$ denote the execution of Π on this input. For an adversary $\mathcal{A}$ (corrupting either P_1 or P_2), we define Let $\mathsf{view}_A^{\Pi}(\mathbf{d}_1, \mathbf{d}_2)$ be the view of the protocol to A (consisting of input, the random tape, the protocol transcript, and the output).

Definition 7.4. Let $\epsilon > 0$ and $0 \le \delta < 1$. A (randomized) protocol Π preserves *computational two-party (ϵ, δ)-Differential Privacy*, if for any PPT distinguisher $\mathcal{D}$, for any PPT adversary $\mathcal{A}$, and for all neighboring inputs $\mathbf{d} := \mathbf{d}_1 \| \mathbf{d}_2$ and $\mathbf{d}' := \mathbf{d}'_1 \| \mathbf{d}'_2$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that,

$$\Pr[\mathcal{D}(\mathsf{view}_A^{\Pi}(\mathbf{d}_1, \mathbf{d}_2), 1^{\kappa}) = 1] \le e^{\epsilon} \cdot \Pr[\mathcal{D}(\mathsf{view}_A^{\Pi}(\mathbf{d}'_1, \mathbf{d}'_2), 1^{\kappa}) = 1] + \delta + \mathsf{negl}(\kappa)$$

Securely Realizing $\mathcal{F}_{\mathsf{biasCoin}}$ with Semi-honest Security

We can securely realize $\mathcal{F}_{\mathsf{biasCoin}}$, by executing $\mathcal{F}_{2PC}$ for the following circuit C_{coinflip}. Since we just execute $\mathcal{F}_{2PC}$ with a circuit, security of the protocol is immediate.

$C_{\mathsf{coinflip}}(\|\mathbf{w}_1\|_1, \{r_{1,j}\}_{j=1}^{\kappa}, b_1, \|\mathbf{w}_2\|_1, \{r_{2,j}\}_{j=1}^{\kappa}, b_2)$ $\quad \triangleright$ r_j, b_j are random bits.

1. P_1's input is $(\|\mathbf{w}_b\|_1, \{r_{1,j}\}, b_1)$ and P_2's input is $(\|\mathbf{w}_2\|_1, \{r_{2,j}\}, b_2)$. We require $\|\mathbf{w}_1\|_1, \|\mathbf{w}_2\|_1, \{r_{1,j}\}, \{r_{2,j}\} \in \{0,1\}^{\kappa}$, and $b_1, b_2 \in \{0,1\}$.
2. Let $s_1 = \|\mathbf{w}_1\|_1$ and $s_2 = \|\mathbf{w}_2\|_1$. Let $s = s_1 + s_2$. Compute $\mathsf{mask} = 0^{\kappa-h}1^h$ such that $s \& \mathsf{mask} = s$ and $s | \mathsf{mask} = \mathsf{mask}$ where $\&$ (resp., $|$) denotes bitwise AND (resp., bitwise OR) operation. Note that there is a single h satisfying the above conditions, i.e., the effective bit-length h of s with $2^{h-1} \le s < 2^h$. This computation can be done by checking all possible candidates of h one by one in $O(\kappa)$ steps.
3. For $j = 1, ..., \lambda$, let $r_j = (r_{1,j} \oplus r_{2,j}) \& \mathsf{mask}$. Note that it holds $r_j < 2^h$.
4. Find the first j^* such that $r_{j^*} \le s$. If there is no such j^* output error.
5. Compute $b = b_1 \oplus b_2$.
6. If $r_{j^*} \le s_1$, output b to both P_1 and P_2. Otherwise, output b to P_1 and $b \oplus 1$ to P_2.

Note that $\Pr[r_j > s] = 1 - s/2^h < 1/2$. Therefore, with λ repetitions, we have a good j^* with probability $1 - 2^{-\lambda}$. Finally, we have $\Pr[r_{j^*} \le s_1 | r_{j^*} \le s] = \frac{s_1}{s} = p$.

7.2.2 Proofs

Proof of Theorem 4.1

We describe the simulator Sim in the $\{\mathcal{F}_{\mathsf{osample(L_1)}}, \mathcal{F}_{2PC}\}$-hybrid model for the case that Party 1 is corrupted. The simulator and proof of security are analogous in the case that Party 2 is corrupted.

Sim receives as input $\mathbf{w}_1$, the output i^*, and $||\mathbf{w}_1 + \mathbf{w}_2||_2$. Sim samples r^* from a geometric distribution with success probability $p = ||\mathbf{w}_1 + \mathbf{w}_2||_2^2$.

Sim invokes Party 1 on input $\mathbf{w}_1$. For $i \in [r^* - 1]$, Party 1 sends its input to the first three invocations of $\mathcal{F}_{\mathsf{osample}(\mathsf{L}_1)}$ and Sim returns to it three random values in $\mathbb{Z}_n$. Party 1 sends its input to the second three invocations of $\mathcal{F}_{\mathsf{osample}(\mathsf{L}_1)}$ and Sim returns to it three random values in $\mathbb{Z}_n$. Party 1 sends its input to the $\mathcal{F}_{2PC}$ functionality and Sim returns to it $\perp$. For $i = r^*$, Party 1 sends its input to the first three invocation of $\mathcal{F}_{\mathsf{osample}(\mathsf{L}_1)}$ and Sim returns to it three random values in $\mathbb{Z}_n$. Party 1 sends its input to the second three invocations of $\mathcal{F}_{\mathsf{osample}(\mathsf{L}_1)}$ and Sim returns to it three random values in $\mathbb{Z}_n$. Party 1 sends its input to the $\mathcal{F}_{2PC}$ functionality and Sim returns to it i^*.

It is clear that the view of Party 1 is identical in the ideal and real world, assuming that Sim samples the first succeeding round, r^*, from the correct distribution. In the following, we argue that this is indeed the case.

As was shown in the correctness analysis, if the protocol has not already halted before round r, then the probability of halting (and outputting some valid index) in round r is:

$$||\mathbf{w}_1||^2 + 2\langle \mathbf{w}_1, \mathbf{w}_2 \rangle + ||\mathbf{w}_2||^2 = ||\mathbf{w}_1|\mathbf{w}_2||_2^2.$$

Since r^* is defined as the round in which the protocol halts, the distribution on r^* is exactly the distribution on the number of Bernoulli trials (with probability $p = ||\mathbf{w}_1|\mathbf{w}_2||_2^2$) needed to get one success. Sampling the number of rounds is therefore equivalent to sampling the random variable corresponding to the number of rounds from a geometric distribution with success probability $p = ||\mathbf{w}_1|\mathbf{w}_2||_2^2$, which is exactly what Sim does.

Proof of Theorem 4.2

We first give the simulation of the sender. The simulator proceeds as follows:

- Send $\mathbf{w}$ to $\mathcal{F}_{\mathsf{osample}(\mathsf{L}_1)}$ and receive π as the output. Place π on the random tape of the sender.
- Simulate the output of F_{2PC} by sending a random r_1 to the sender.
- Simulate the key generation protocol honestly.
- Send a random encryption for $[\![r_2]\!]$ in Step 4.

Due the semantic security of the underlying FHE scheme, the simulation is indistinguishable. Next, we give the simulation of the receiver.

- The simulator receives output $i \oplus \pi$ from $\mathcal{F}_{\mathsf{osample}(\mathsf{L}_1)}$.
- The simulator works as functionality F_{2PC} sending a random r_2 as the output to the receiver.

- The simulator runs the key generation protocol honestly, and stores the threshold decryption key of the sender.
- In step 6, the simulator computes $c := [\![i \oplus \pi]\!]$ and sends it to the receiver.
- The simulator runs the threshold decryption protocol honestly.

The simulation is perfect.

Proof of Theorem 4.10

We describe the simulator Sim in the $\{\mathcal{F}_{L_1}^{ss}, \mathcal{F}_{2PC}\}$-hybrid model for the case that Party 1 is corrupted. The simulator and proof of security are analogous in the case that Party 2 is corrupted.

Sim receives as input $\mathbf{w}_1$ and the output i^*. Sim invokes Party 1 on input $\mathbf{w}_1$. For $j \in [B]$, the simulator works as follows:

- Upon Party 1 sending its input to $\mathcal{F}_{L_1}$, Sim returns a uniformly random share r
- In place of the encryption of w_{2,i_j} from Party 2, Sim sends Party 1 an FHE ciphertext encrypting 0.
- Upon Party 1 sending its input to $\mathcal{F}_{2PC}$, Sim returns to Party 1 an FHE ciphertext encrypting 0.

The only differences in the view of Party 1 in the ideal and hybrid worlds, are that (1) In the hybrid world it gets a secret share of i_j, whereas in the ideal world it gets a uniformly random value; (2) In the hybrid world it gets an encryption of w_{2,i_j} from Party 2, whereas in the ideal world it gets an encryption of 0; (3) In the hybrid world it gets encryptions of i_j or 0 from the ideal functionality $\mathcal{F}_{2PC}$, whereas in the ideal world it always gets encryptions of 0.

Receiving uniformly random values instead of correct secret shares does not affect the view of Party 1, since the additive secret sharing used has perfect secrecy. Further, switching from encryptions of w_{2,i_j} and i_j to encryptions of 0 is indistinguishable due to the semantic security of the threshold FHE scheme. Thus, the view of Party 1 is computationally indistinguishable in the hybrid world and the ideal world.

This concludes the proof of security of the L_2 sampling protocol.

Proof of Lemma 4.15

We want to show that

$$\begin{aligned}
\Pr_{D_{L_p}(\mathbf{w}_1,\mathbf{w}_2)}[i] &= \frac{(w_{1,i}+w_{2,i})^p}{||\mathbf{w}_1+\mathbf{w}_2||_p^p} \\
&= \frac{(w_{1,i}+w_{2,i})^p}{c\cdot(||\mathbf{w}_1||_p^p+||\mathbf{w}_2||_p^p)} \\
&\leq \frac{2^{p-1}(w_{1,i}^p+w_{2,i}^p)}{c\cdot(||\mathbf{w}_1||_p^p+||\mathbf{w}_2||_p^p)} \\
&= 2^{p-1}/c\cdot \Pr_{D_{\mathsf{ignore},p}(\mathbf{w}_1,\mathbf{w}_2)}[i].
\end{aligned}$$

The inequality holds due to Jensen's inequality with convex function $f(x) = x^p$:

$$\begin{aligned} 1/2^p \cdot (w_{1,i} + w_{2,i})^p &= f(1/2 \cdot w_{i,1} + 1/2 \cdot w_{i,2}) \\ &\leq 1/2 \cdot f(w_{i,1}) + 1/2 \cdot f(w_{2,i}) \\ &= 1/2(w_{i,1}^p + w_{i,2}^p). \end{aligned}$$

This completes the proof of Lemma 4.15.

Proof of Lemma 4.16

Note that Π_{L_p} simply performs rejection sampling in a distributed setting where sampling from $D_{\mathsf{ignore},p}(\mathbf{w}_1, \mathbf{w}_2)$ and computing the probabilities is done in a distributed manner. It is therefore well-known that as long as for all $i \in [n]$,

$$\Pr_{D_{L_p}(\mathbf{w}_1,\mathbf{w}_2)}[i] \leq 2^p/c \cdot \Pr_{D_{\mathsf{ignore},p}(\mathbf{w}_1,\mathbf{w}_2)}[i], \tag{7}$$

then Π_{L_2} samples from the exact correct distribution, and the number of samples required from $D_{\mathsf{ignore},p}(\mathbf{w}_1, \mathbf{w}_2)$ in protocol Π_{L_2} follows a geometric distribution with probability $c/2^p$. Thus, if condition (7) is met, the protocol samples exactly correctly and completes in an expected $2^p/c \leq 2^p \in \tilde{O}(1)$ number of rounds (since $c \geq 1$ and $p \in O(1)$). Further, it can be immediately noted that condition (7) is met due to Lemma 4.15. Finally, each round has $\tilde{O}(1)$ communication, since Π_{ignore} has communication $\tilde{O}(1)$ (by Lemma 4.12) and since, in addition to that, only a constant number of length $\tilde{O}(1)$ values are exchanged in each round. Combining the above, we have that Π_{L_p} has expected communication $\tilde{O}(1)$ and worst case (with all but negligible probability) communication $\tilde{O}(1)$.

Proof of Lemma 4.23

We will show that the joint distribution over the i-th entries of $\tilde{\mathbf{w}}_1 := \mathbf{M}\mathbf{w}_1 = (\tilde{w}_{1,1}, \ldots, \tilde{w}_{1,k})$, $\tilde{\mathbf{w}}_2 = \mathbf{M}\mathbf{w}_2 = (\tilde{w}_{2,1}, \ldots, \tilde{w}_{2,k})$ can be sampled perfectly, given only $\langle \mathbf{w}_1, \mathbf{w}_1 \rangle$, $\langle \mathbf{w}_2, \mathbf{w}_2 \rangle$, and $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$.

Due to independence of each of the coordinates of $\tilde{\mathbf{w}}_1, \tilde{\mathbf{w}}_2$, this immediately implies that the entire $\mathsf{approxIP}(\mathbf{w}_1, \mathbf{w}_2)$ can be simulated perfectly given only $\langle \mathbf{w}_1, \mathbf{w}_1 \rangle$, $\langle \mathbf{w}_2, \mathbf{w}_2 \rangle$, and $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$.

We begin by noting that

$$\begin{aligned} \tilde{w}_{1,i} &= w_{1,1}M_{i,1} + w_{1,2}M_{i,2} + \cdots + w_{1,n}M_{i,n} \\ \tilde{w}_{2,i} &= w_{2,1}M_{i,1} + w_{2,2}M_{i,2} + \cdot + w_{2,n}M_{i,n} \end{aligned}$$

In the following, we show how to jointly sample $(\tilde{w}_{1,i}, \tilde{w}_{2,i})$.

Step 1: We begin by sampling from the marginal distribution over the first element of the tuple $\tilde{w}_{1,i}$. Note that $\tilde{w}_{1,i}$ is distributed exactly as a Gaussian random variable with mean 0 and variance $\langle \mathbf{w}_1, \mathbf{w}_1 \rangle$. Thus, we can perfectly sample from the marginal distribution over $\tilde{w}_{1,i}$ given only $\langle \mathbf{w}_1, \mathbf{w}_1 \rangle$. Let z be the resulting sample.

Step 2: We would now like to sample from the conditional distribution $\tilde{w}_{2,i}$, conditioned on $\tilde{w}_{1,i} = z$.

First, the conditional distribution of $M_{i,1}, \ldots, M_{i,n}$ conditioned on $w_{1,1}M_{i,1} + w_{1,2}M_{i,2} + \cdots + w_{1,n}M_{i,n} = z$ is defined by the multivariate Gaussian distribution with the following mean μ and covariance matrix $\mathbf{\Sigma}$ (see Corollary 7 in [55]):

$$\mu = \frac{z}{\langle \mathbf{w}_1, \mathbf{w}_1 \rangle} \cdot \mathbf{w}_1, \qquad \mathbf{\Sigma} = \mathbf{I} - \frac{\mathbf{w}_1 \mathbf{w}_1^T}{\langle \mathbf{w}_1, \mathbf{w}_1 \rangle}$$

Now, $\tilde{w}_{2,i}$ is a linear combination of the variables $M_{i,1}, \ldots, M_{i,n}$ with coefficients $w_{2,1}, \ldots, w_{2,n}$. Therefore, $\tilde{w}_{2,i}$ is distributed as a univariate Gaussian with mean μ' and variance σ' as follows (see [129] for example).

$$\mu' = \langle \mathbf{w}_2, \mu \rangle = \frac{z\langle \mathbf{w}_1, \mathbf{w}_2 \rangle}{\langle \mathbf{w}_1, \mathbf{w}_1 \rangle}$$

and

$$\sigma' = \mathbf{w}_2^T \mathbf{\Sigma} \mathbf{w}_2 = \mathbf{w}_2^T \mathbf{w}_2 - \frac{\mathbf{w}_2^T \mathbf{w}_1 \mathbf{w}_1^T \mathbf{w}_2}{\langle \mathbf{w}_1, \mathbf{w}_1 \rangle} = \langle \mathbf{w}_2, \mathbf{w}_2 \rangle - \frac{(\langle \mathbf{w}_1, \mathbf{w}_2 \rangle)^2}{\langle \mathbf{w}_1, \mathbf{w}_1 \rangle}$$

Note that the mean and variance depend only on $\langle \mathbf{w}_1, \mathbf{w}_1 \rangle$, $\langle \mathbf{w}_2, \mathbf{w}_2 \rangle$, and $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$, so we can sample from this distribution given only those values. Let y be the resulting sample.

Step 3: Output (z, y)

7.2.3 Realizing $\mathcal{F}_{\mathsf{osample(L_1)}}$ with OT with Less Precision

We implement oblivious sampling using a 1-out-of-m OT scheme. In particular, the receiver, as an OT receiver, chooses a random index from $[m]$, and the sender, as an OT sender, prepares an m-dimensional input vector that encodes the L_1 distribution of $\mathbf{w}$ in a way that we will describe soon.

Assumption about the level of precision of the input. With this approach, each element from the prepared OT input vector will be chosen uniformly with probability $\frac{1}{m}$. Therefore, the size m affects the level of precision of the sampling. In particular, we set $\mu := 1/m$ as a precision unit, and we assume the following:

For each $i \in [n]$, it holds that $\frac{w_i}{\|\mathbf{w}\|_1}$ is a multiple of μ.

If the input vector $\mathbf{w}$ is not consistent with the above requirement, one can round it by using the following function $\mathsf{rounding}_\mu(\mathbf{w})$:

$\mathsf{rounding}_\mu(\mathbf{w})$

1. Let $\mathbf{w} = (w_1, \ldots, w_n)$. For $i = 1, \ldots, n$, compute $w_i' = \mathsf{trunc}_\mu(w_i)$. Here, for any real number x with $x \in [0, 1]$, denote $\mathsf{trunc}_\mu(x) = \tilde{x} \cdot \mu$ where $\tilde{x}$ is an integer that minimizes $\Delta := x - \tilde{x} \cdot \upsilon$ subject to $\Delta \geq 0$. Typically, we have $\mu = 2^{-q}$ for a certain positive integer q, and $\mathsf{trunc}_\mu(x)$ is simply truncating the lower order bits in the binary representation of x.

   Let $\mathbf{w}' = (w_1', \ldots, w_n')$.

2. Repeat the following until L_1 norm of $\mathbf{w}'$ becomes 1.

   Find $j = \arg\max_{i\in[n]}(w_i - w_i')$, and increase w_j' by μ.

3. Output $\mathbf{w}'$.

The above algorithm makes sure that for all i, it holds $|w_i' - w_i| < \mu$, which means $\mathbf{w}'$ is a good approximation of $\mathbf{w}$, with each element having an additive error of at most μ. To see why, note that in step 1, some w_i's will get truncated leading to small difference, i.e., $w_i - w_i' < \mu$. In step 2, since the truncated weights are added back to the elements in decreasing order of difference $(w_i - w_i')$, only some of the truncated w_i's will be updated to $w_i' + \mu$ (which will still be close to w_i) until the L_1 norm of $\mathbf{w}'$ becomes 1.

In a situation where the low precision is acceptable, this OT-based solutions could be more efficient. However, if one needs a higher level of precision, we recommend using the FHE-based solution described in the next subsection. We also observe that by using an OT protocol with $O(\log m)$ communication [108, Theorem 2.2], we expect we could support fairly large values of m. The bottleneck on larger m is likely to be storage, and the computation time needed for the OT.

Protocol 31 Oblivious sampling protocol realizing $\mathcal{F}_{\mathsf{osample(L_1)}}$ based on 1-out-of-m OT

Inputs: The sender has input $\mathbf{w}$. We require every $w_i/\|\mathbf{w}\|_1$ is a multiple of $\mu := \frac{1}{m}$.

1. The sender computes the following:

   (a) Given $\mathbf{w}$, the sender prepares an m-dimensional input vector as follows:

   (b) For $i = 1, \dots, n$, do:

      Let $k_i = \frac{w_i}{\|\mathbf{w}\|_1} \cdot m$. Insert k_i copies of the index i into the m-dimensional vector $\mathbf{v}$; that is, there should be k_i slots (out of m) whose value is i in the m-dimensional vector $\mathbf{v}$.

      Note that for each i, the fraction of the slots containing the index i in $\mathbf{v}$ is $\frac{k_i}{m} = \frac{w_i}{\|\mathbf{w}\|_1}$.

   (c) The sender chooses a random pad π uniformly at random.

   (d) Let $\mathbf{v} = (v_1, \dots, v_m)$. The sender shuffles $\mathbf{v}$ and blinds it by updating $v_i := v_i \oplus \pi$ with a randomly chosen π.

2. Execute an OT protocol where the sender is the OT sender with input $\mathbf{v}$ and the receiver is the OT receiver with a randomly chosen number from $[m]$. Let u be the output to the OT receiver.

3. Output π to the sender and output u to the receiver.

Oblivious sampling protocol. With the assumption about the level of precision of the input vector $\mathbf{w}$, we can implement oblivious sampling. See Protocol 31.

Due to security of the OT protocol, the OT sender won't know the OT receiver's choice. However, since the OT receiver does know the sampled index, which leaks information about the data array, we hide this information from the OT receiver by having the OT sender *shuffle* the input vector.

At the end of the OT protocol, the sender and receiver will hold the sampled index i in a secret shared form; that is, the sender will hold π and the receiver $\pi \oplus i$. Note that the sender re-uses the same π across all inputs, in order to fix its share, independently of the receiver index. Security holds even with the re-use of this value because the receiver learns only a single element.

Security. We will prove the following theorem.

Theorem 7.5. Protocol 31 securely realizes $\mathcal{F}_{\mathsf{osample(L_1)}}$ in the semi-honest security model.

Proof. We first give the simulation of the sender. This is trivial in the OT-hybrid model, as the sender receives no messages in this protocol. The simulator submits the sender's input to $\mathcal{F}_{\mathsf{osample(L_1)}}$, and recieves π as output. It places π on the sender's random tape, accepts the sender's input to the OT functionality, and terminates.

Next, we give the simulation of the receiver. The simulator submits input to $\mathcal{F}_{\mathsf{osample(L_1)}}$, and receives output u. Upon receiving OT choice from the adversary, it feeds u to the adversary as the OT output. The simulation is perfect, since the view of the adversary contains nothing more than u. ∎

7.2.4 L_1 Sampling Protocol with Secret Shared Output

We give a more formal description of the functionality $\mathcal{F}_{L_1}^{ss}$.

$\mathcal{F}_{L_1}$: Ideal functionality for two-party L_1 sampling

The functionality has the following parameter:

- $n \in \mathbb{N}$. The dimension of the input weight vectors $\mathbf{w}_1$ and $\mathbf{w}_2$.

The functionality proceeds as follows:

1. Receive inputs $\mathbf{w}_1$ and $\mathbf{w}_2$ from P_1 and P_2 respectively.
2. Sample $i \in [n]$ with probability $\frac{w_{1,i}+w_{2,i}}{\|\mathbf{w}_1+\mathbf{w}_2\|_1}$
3. Choose a random number π consisting of $\lceil \log n \rceil$ bits.
4. Send π to P_1 and $i \oplus \pi$ to P_2.

We describe a protocol securely realizing $\mathcal{F}_{L_1}^{ss}$ in the $(\mathcal{F}_{\mathsf{osample(L_1)}}, \mathcal{F}_{\mathsf{biasCoin}})$-hybrid.

Security of the protocol can be shown similarly to Theorem 4.1.

Protocol 32 Protocol securely realizing $\mathcal{F}^{ss}_{L_1}$ in the $(\mathcal{F}_{\mathsf{osample(L_1)}}, \mathcal{F}_{\mathsf{biasCoin}})$-hybrid.

Inputs: Party P_b has input $\mathbf{w}_b$.

1. Execute $\mathcal{F}_{\mathsf{osample(L_1)}}$ with P_1 as a sender with input $\mathbf{w}_1$ and P_2 as a receiver. Let $\langle i_1 \rangle$ be the secret share of the output index.
2. Execute $\mathcal{F}_{\mathsf{osample(L_1)}}$ with P_2 as a sender with input $\mathbf{w}_2$ and P_1 as a receiver. Let $\langle i_2 \rangle$ be the secret share of the output index.
3. Execute $\mathcal{F}_{\mathsf{biasCoin}}$ where P_1 has input $\|\mathbf{w}_1\|_1$ and P_2 has input $\|\mathbf{w}_2\|_1$. Let $\langle b \rangle$ be the secret share of the output bit. In addition, P_1 chooses random bits π.
4. Execute $\mathcal{F}_{2PC}$ for the following circuit:
   (a) Input: $\langle i_1 \rangle, \langle i_2 \rangle, \langle b \rangle, \pi$.
   (b) Compute $i = i_1 \cdot (1-b) + i_2 \cdot b$.
   (c) Output π to P_1 and $\pi \oplus i$ to P_2.

7.3 Min-Hash

7.3.1 Proof of Lemma 5.4

Let $c = p/(1-p)$. We need to find $a < np$ and $b > np$ satisfying the following condition.

$$\frac{\Pr_{\mathrm{B}(n,p)}[a]}{\Pr_{\mathrm{B}(n,p)+s}[a]} = \frac{\binom{n}{a} p^a (1-p)^{n-a}}{\binom{n}{a-s} p^{a-s}(1-p)^{n-a+s}} < \left(\frac{n-a}{a-s}\right)^s \cdot c^s \le e^{\epsilon}.$$

$$\frac{\Pr_{\mathrm{B}(n,p)}[b]}{\Pr_{\mathrm{B}(n,p)+s}[b]} = \frac{\binom{n}{b} p^b (1-p)^{n-b}}{\binom{n}{b-s} p^{b-s}(1-p)^{n-b+s}} > \left(\frac{n-b}{b}\right)^s \cdot c^s \ge e^{-\epsilon}.$$

Case 1: $s \ge \epsilon$. We set $a = \frac{np + s(1-p)\cdot e^{\epsilon/s}}{e^{\epsilon/s}\cdot(1-p)+p}$ and $b = \frac{e^{\epsilon/s}\cdot np}{(1-p)+e^{\epsilon/s}\cdot p}$. Note these values satisfy the above inequalities.

To show the second requirement of the tail bound, it suffices to show that $\Pr_{\mathrm{B}(n,p)}[X \le a+s] = \mathsf{negl}(\kappa)$; the other case holds similarly.

Let $\mu := np \in \Theta(\kappa)$ and let $d = 1 - (a+s)/\mu$. By applying the Chernoff bound, we have

$$\Pr_{X \leftarrow \mathrm{B}(n,p)}[X \le a+s] = \Pr[X \le (1-d)\mu] \le \exp(-d^2\mu/2).$$

We will show that we have $d = \Omega(\frac{1}{\lg\lg\kappa})$, which implies that with $\mu \in \Theta(\kappa)$, the above probability is negligible in κ. Let $t = s(1-p)\cdot e^{\epsilon/s}$ and $u = e^{\epsilon/s}\cdot(1-p)+p$.

Then, we have $a = \frac{\mu+t}{u}$. Note that we have $\epsilon/s \cdot (1-p) + 1 \leq u \leq e$. We have the following:

$$d = \frac{\mu - a - s}{\mu} = \left(1 - \frac{1}{u}\right) - \frac{t/u + s}{\mu} \geq \frac{\epsilon/s \cdot (1-p)}{e} - \frac{t/u+s}{\mu} = \Omega\left(\frac{1}{\lg\lg\kappa}\right) - \tilde{O}(1/\kappa).$$

Case 2: $s \leq \epsilon$**.** Let $a = \frac{c}{c+e} \cdot (n + e\epsilon) < np$ and $b = \frac{c}{c+e^{-1}} \cdot n > np$. Observe that $\frac{n-a}{b-s} \cdot c = e$ and $\frac{n-b}{b} \cdot c = e^{-1}$. Therefore, given $s \leq \epsilon$, the above inequalities hold. The tail bounds specified as the second condition of the lemma can be shown using the Chernoff bound since $np - a \in \Theta(np) = \Theta(\kappa)$ and so does $b - np$.

7.3.2 Proof of Lemma 5.8

Let $t = \frac{1}{n_R}$, Let $\mathsf{low} = 1 - \left(\frac{1}{2}+\theta\right)^t$ and $\mathsf{high} = 1 - \left(\frac{1}{2}-\theta\right)^t$. Then, we have:

$$\begin{aligned}
&\Pr_h[\mathsf{good}_\theta(h, A, I, n_B)] \\
&= \Pr_h\left[(\min h(I) \geq \mathsf{low}) \wedge (\min h(I) = \min h(A))\right] \\
&\quad - \Pr_h\left[(\min h(I) \geq \mathsf{high}) \wedge (\min h(I) = \min h(A))\right] \\
&= \Pr_h\left[\min h(A) \geq \mathsf{low}\right] \cdot \Pr\left[\min h(I) = \min h(A) \mid \min h(A) \geq \mathsf{low}\right] \\
&\quad - \Pr_h\left[\min h(A) \geq \mathsf{high}\right] \cdot \Pr\left[\min h(I) = \min h(A) \mid \min h(A) \geq \mathsf{high}\right] \\
&\geq \left(\frac{1}{2}+\theta\right)^{t \cdot n_A} \cdot \frac{n_I}{n_A} - \left(\frac{1}{2}-\theta\right)^{t \cdot n_A} \cdot \frac{n_I}{n_A}
\end{aligned}$$

7.3.3 Proof of Lemma 5.9

We can lowerbound $|K_\theta|$ with $\mathrm{B}(k-s, p_\theta)$. Recall $s \in O(\lg\lg\kappa)$. Let $\mu = (k-s)p_\theta = \Omega(\kappa)$. Applying the Chernoff Bound, we have $\Pr\left[|K_\theta| \leq (1-1/3)\mu\right] \leq \exp(-\mu/18) \leq \mathsf{negl}(\kappa)$.

7.3.4 Proof of Lemma 5.10

Hockey stick divergence. We first review hockey stick divergence [13,159,190]. The hockey-stick divergence between two probability measures P, Q over Z is defined as:

$$\mathsf{D}^{\mathsf{hs}}_\alpha(X, Y) = \sup_{S \subseteq Z}(X(S) - \alpha Y(S)) = \sum_{z \in Z}[(X(z) - \alpha Y(z)]_+,$$

where $\alpha \geq 1$ and $[x]_+ = \max\{x, 0\}$. We observe that the following holds directly from the definition of the hockey stick divergence.

Corollary 7.6. For any probability measures X, Y over Z and for any ϵ, δ, it holds

$$X \approx_{\epsilon,\delta} Y \text{ if and only if } \mathsf{D}^{\mathsf{hs}}_{e^\epsilon}(X, Y) \leq \delta \text{ and } \mathsf{D}^{\mathsf{hs}}_{e^\epsilon}(Y, X) \leq \delta.$$

Proof of Lemma 6. Let $\eta_{-\theta} = 1/2 - \theta$ and $\eta_{+\theta} = 1/2 + \theta$. For brevity, we let C denote $\mathsf{PB}(n, p_J)$. For any distribution $\mathcal{D}$, let $P_{\mathcal{D}}$ denote the probability measure with respect to $\mathcal{D}$. We first show that for any $\epsilon > 0$, it holds that $\mathsf{D}^{\mathsf{hs}}_{e^\epsilon}(P_C, P_{C+1})$ is at most

$$\max\left(\mathsf{D}^{\mathsf{hs}}_{e^\epsilon}\left(P_{\mathrm{B}(\lceil\frac{n}{2}\rceil,\eta_{+\theta})}, P_{\mathrm{B}(\lceil\frac{n}{2}\rceil,\eta_{+\theta})+1}\right), \mathsf{D}^{\mathsf{hs}}_{e^\epsilon}\left(P_{\mathrm{B}(\lceil\frac{n}{2}\rceil,\eta_{-\theta})}, P_{\mathrm{B}(\lceil\frac{n}{2}\rceil,\eta_{-\theta})+1}\right)\right).$$

We start with an upper bound of the hockey-stick divergence is reached at extreme points. We rely on the results in [42]. Although they use the Renyi divergence, their results are general enough to be applied to any f-divergence.

Lemma 7.7. ([42, Lemma 3.5])

$$\mathsf{D}^{\mathsf{hs}}_{e^\epsilon}(P_C, P_{C+1}) \le \max_{j\in[n]} \mathsf{D}^{\mathsf{hs}}_{e^\epsilon}\left(P_{\mathrm{B}(j,\eta_{-\theta})+\mathrm{B}(n-j,\eta_{+\theta})}, P_{\mathrm{B}(j,\eta_{-\theta})+\mathrm{B}(n-j,\eta_{+\theta})+1}\right). \quad (8)$$

Next, we apply data processing inequality to simplify (8) from the above lemma.

Lemma 7.8. ([42, Lemma 3.6]) (8) is upper bounded by

$$\max\left(\mathsf{D}^{\mathsf{hs}}_{e^\epsilon}\left(P_{\mathrm{B}(\lceil\frac{n}{2}\rceil,\eta_{+\theta})}, P_{\mathrm{B}(\lceil\frac{n}{2}\rceil,\eta_{+\theta})+1}\right), \mathsf{D}^{\mathsf{hs}}_{e^\epsilon}\left(P_{\mathrm{B}(\lceil\frac{n}{2}\rceil,\eta_{-\theta})}, P_{\mathrm{B}(\lceil\frac{n}{2}\rceil,\eta_{-\theta})+1}\right)\right). \quad (9)$$

We extend the above to upper bound the hockey-stick divergence between probability measures differed by an integer amount greater than 1, i.e., P_C and P_{C+s} for $s > 1$.

Corollary 7.9. For any $\epsilon > 0$, it holds that $\mathsf{D}^{\mathsf{hs}}_{e^\epsilon}(P_C, P_{C+s})$ is at most

$$\max\left(\mathsf{D}^{\mathsf{hs}}_{e^\epsilon}\left(P_{\mathrm{B}(\lceil\frac{n}{2}\rceil,\eta_{+\theta})}, P_{\mathrm{B}(\lceil\frac{n}{2}\rceil,\eta_{+\theta})+s}\right), \mathsf{D}^{\mathsf{hs}}_{e^\epsilon}\left(P_{\mathrm{B}(\lceil\frac{n}{2}\rceil,\eta_{-\theta})}, P_{\mathrm{B}(\lceil\frac{n}{2}\rceil,\eta_{-\theta})+s}\right)\right).$$

Finally, to give a bound on the divergence, we can apply Lemma 5.4 to argue that the binomial distribution hides the small sensitivity. Specifically, as $\lceil\frac{n}{2}\rceil \in \Theta(\kappa)$ and $s = \lg\lg\kappa$, we can claim (ϵ, δ)-DDP with $\delta = \mathsf{negl}(\kappa)$.

Similarly, it holds that $\mathsf{D}^{\mathsf{hs}}_{e^\epsilon}(P_{C+s}, P_C) \le \mathsf{negl}(\kappa)$. ■

7.3.5 Proof of Theorem 5.6

On the definition of a θ-good iteration. We keep the same definition of a θ-good iteration, except we set the exponent to $1/n'_R$, instead of $1/n_R$, and we also require $\theta \le 1/10$. In particular,

- $\min h(A) = \min h(I)$ and $\min h(I) \in \left[1 - \left(\frac{1}{2} + \theta\right)^t, 1 - \left(\frac{1}{2} - \theta\right)^t\right]$ with $\boxed{t = \dfrac{1}{n'_R}}$.

Bundle of good iterations K_θ. The total number of iterations in the min-hash protocol π_{NMH} is $k = \Omega(\kappa \cdot \lg\lg\kappa)$. We require that $n_R/k^2 = \Omega(\kappa)$.

Using Lemma 5.8, with all but negligible probability, at least $\Omega(\kappa \cdot \lg\lg\kappa)$ iterations are θ-good. Recall that G_θ denotes the set of θ-good iterations, and $K_\theta = G_\theta \setminus S_{x^*}$. We set $k_g = |K_\theta|$. We further divide these k_g iterations into $u = \lg\lg\kappa$ bundles, each of which is of size $k_b = \Omega(\kappa)$. Those bundles are denoted by $K_{\theta,1}, \ldots, K_{\theta,u}$. We also let $K_{bad} := \overline{K_\theta}$.

Random variables for the protocol output. Let out^+_{bad} be the protocol's match count for sets A, B_{+x^*} w.r.t. the hash functions in K_{bad}:

$$out^+_{bad} := \left|\{j \in K_{bad} : \min h_j(A) = \min h_j(B_{+x^*})\}\right|.$$

Likewise, let out_{bad} be the number of matches for sets A and B (instead of B_{+x^*}) in iterations in K_{bad}. Similarly, for $i \in [u]$, we let out^+_i and out_i denote the output for the i-th bundle, with or without x^* respectively. Note that $out^+_i = out_i$, since we ruled out S_{x^*} from K_θ. Note that the final output of the min-hash protocol for input B_{+x^*} is equal to $out^+_{bad} + \sum_{i=1}^{u} out_i$; the final output for input B is $out_{bad} + \sum_{i=1}^{u} out_i$. Let

$$\mathbf{out} = out^+_{bad} || out_{bad} || out_1 || \cdots || out_u.$$

We also consider the output with the ith bundle missing; that is, for $i \in [u]$ let

$$\mathbf{out}_{-i} = out^+_{bad} || out_{bad} || out_1 || \cdots || out_{i-1} || out_{i+1} || \cdots || out_u.$$

Upper-bounding leakage from the output. Since $|K_{bad}|$ and $|K_{\theta,i}|$ are at most $k \in poly(\kappa)$, we can safely assume that the total number of bits in **out** is

$$2\lg|K_{bad}| + \sum_{i=1}^{u} \lg|K_{\theta,i}| \le (2 + \lg\lg\kappa)\lg|poly(\kappa)| \le \kappa.$$

Distribution of R and its min-entropy. The original distribution on the secret set R is the uniform distribution over all sets of size n_R with each element is chosen from a universe $\mathcal{U}$. The universe $\mathcal{U}$ has size $\ell \cdot n_R$ with $\ell \ge 4(n_R)^3$.

Now choose, uniformly at random, a partition $\{U_1, \ldots, U_{n_R}\}$ of $\mathcal{U}$ where each $|U_j| = \ell$ such that the element in the jth slot of R belongs to U_j. These universes $\{U_1, \ldots, U_{n_R}\}$ are leaked in the analysis.

Let $\mathcal{D}$ denote the original distribution over the set R, but conditioned on the leaked information $\{U_1, \ldots, U_{n_R}\}$. The distribution $\mathcal{D}$ is equivalent to a distribution over streams of n_R elements, where the element in the i-th slot is chosen uniformly at random from U_i. Therefore, $\mathcal{D}$ has min-entropy $n_R \lg \ell$.

We additionally consider arbitrary leakage $f(R)$ of length L such that

$$n_R \lg \ell - L \ge \frac{8n_R}{9}\lg\ell + 2n_R.$$

Available iterations in a bundle. For a fixed set $Z \subseteq R$, in a min-hash graph, we say that a set of iterations in the ith bundle $K_{\theta,i}$ is available with respect to Z if there are no edges from Z to that set. In other words, no elements in Z contribute to the final count reduction for any of those iterations. In this sense, those iterations are is still available for the count reduction by the other elements than those in Z. More formally, consider a graph $G \leftarrow \mathbf{MinhashG}_{H_1}(A, I, x^*, H_2)$ and letting $G = (\mathcal{X}, \mathcal{Y}, \mathcal{E})$, we define

$$\mathsf{Avail}_G(K_{\theta,i}, Z) := \{j \in K_{\theta,i} : \forall z \in Z : (z, j) \notin \mathcal{E})\}.$$

Existence of a good bundle. We now describe an experiment to check if the ith bundle of iterations is good in the sense that given the fixed hash, the distribution $\mathcal{D}$ (after the leakage) satisfies the DP-like property conditions specified in Lemma 5.13. Roughly speaking, Lemma 5.13 shows that a bundle will be good with a high probability.

Process $\mathbf{IsAGoodBundle}(i, \mathbf{out}_{-i}, \mathcal{D}, A, I, x^*, H_1, H_2)$:

1. Consider $G \leftarrow \mathbf{MinhashG}_{H_1}(A, I, x^*, H_2)$.
2. Let $\mathcal{D}_{1,i} := \mathcal{D} \mid \mathbf{out}_{-i}$. In other words, $\mathcal{D}_{1,i}$ is the distribution $\mathcal{D}$ on R, but conditioned on the output vector $\mathbf{out}_{-i}$. If $\mathcal{D}_{1,i}$ has min-entropy less than $n_R \lg(\ell) - L - 2\kappa$ then output $\mathsf{FAIL}_{1,i}$ and terminate.
3. Check if if there is a leakage function $f_G(R)$ which leaks $V = \{j_1, \ldots, j_{n'_R}\}$ and $T = \mathsf{Avail}_G(K_{\theta,i}, R \setminus R')$ such that there exists a distribution with the Geometric Collision Property over sets $R' = \{x_j \in R : j \in V\}$. If there is no such distribution, output $\mathsf{FAIL}_{2,i}$ and terminate. Let $\mathcal{D}_{2,i} := \mathcal{D}_{1,i}|f_G(R)$.
4. If it holds $|T| \leq \frac{1}{10}|K_{\theta,i}|$, output $\mathsf{FAIL}_{3,i}$ and terminate. Let $k_v = |T|$.
5. Compute $D_{T,r}(\mathcal{D}_{2,i})$ and check if $D_{T,r}$ satisfies the conditions given in Lemma 5.13. Output $\mathsf{FAIL}_{4,i}$ and terminate, if the above check fails.
6. Output SUCCESS.

Failure probability $\mathsf{FAIL}_{1,i}$. We claim that $\mathsf{FAIL}_{1,i}$ takes place with a negligible probability. By applying [58, Lemma 2.2], the average min-entropy of $\mathcal{D}|\mathbf{out}_{-i}$ is at least $n_R \lg \ell - L - \kappa$, which implies that the min-entropy of $\mathcal{D}|\mathbf{out}_{-i}$ is at least $n_R \lg \ell - L - 2\kappa \geq \frac{8n_R}{9} \lg \ell + n_R$ with probability $1 - 2^{-\kappa}$ (assuming that $n_R \geq 2\kappa$).

Failure probability $\mathsf{FAIL}_{2,i}$.

Lemma 7.10. The experiment outptus $\mathsf{FAIL}_{2,i}$ with a negligible probability.

We give the proof later in Section 7.3.8.

Failure probability $\mathsf{FAIL}_{3,i}$ We show that $\mathsf{FAIL}_{3,i}$ occurs with negligible probability. Let $n = n_R$ and $n' = n'_R$ for brevity of notation. Recall that

$n' = n/3$. Let X_j be an indicator variable that represents whether there is an edge from $(n-n')$ nodes to iteration j. Therefore, we have

$$\Pr_{H_2}[|T| = r] = \Pr_{H_2}\left[\sum_{j=1}^{k_b} X_j = k_b - r\right].$$

Recall that $p_j \leq 1-(\eta_{-\theta})^{1/n'}$ and $\Pr[X_j = 1] = 1-(1-p_j)^{n-n'} \leq 1-(\eta_{-\theta})^{\frac{n-n'}{n'}} = 1-(\eta_{-\theta})^2 \leq 1-(2/5)^2$. Therefore, we have

$$m := \mathbf{E}\left[\sum_{j=1}^{k_b} X_j\right] \leq k_b \cdot (1-(\eta_{-\theta})^2) \leq 0.84k_b$$

Using the Chernoff bound and due to $k_b \in \Omega(\kappa)$, we have

$$\Pr_{H_2}\left[|T| \leq \frac{k_b}{10}\right] = \Pr_{H_2}\left[\sum_{j=1}^{k_b} X_j \geq \frac{9}{10}k_b\right] \leq \exp\left(-\frac{(0.9k_b - m_0)^2}{2m_0}\right) = \exp(-\Omega(\kappa)).$$

Failure probability $\mathsf{FAIL}_{4,i}$. By Lemma 5.13, for all $i \in [u]$, conditioned on $\mathsf{FAIL}_{1,i}$, $\mathsf{FAIL}_{2,i}$, $\mathsf{FAIL}_{3,i}$ not occurring, let

$$p_4 := \Pr_{H_1,H_2}[\textbf{IsAGoodBundle}(i, \mathbf{out}_{-i}, \mathcal{D}, A, I, x^*, H_1, H_2) = \mathsf{FAIL}_{4,i}].$$

Then, we have $p_4 \in O(k_v \lg^3(\kappa)/(n_R)^{0.5})$.

Existence of a good bundle out of u bundles. Observe that conditioned on $\mathsf{FAIL}_{1,i}$, $\mathsf{FAIL}_{2,i}$, $\mathsf{FAIL}_{3,i}$ not occurring, the process outputs $\mathsf{FAIL}_{4,i}$ independently of $(\mathbf{out}_{-i}, R')$, since the hash values in H_2 for any iteration are chosen independently of those for the other iterations. Using the above, since $k_v/\sqrt{n_R} = O(1/\sqrt{\kappa})$, we have the following:

> *The experiment* **IsAGoodBundle** *outputs SUCCESS for at least one bundle with probability* $1 - p_4^u = 1 - \mathsf{negl}(\kappa)$.

Noise distribution. We define a noise distribution Φ and give an analysis of the hockey stick divergence of $\Phi(r)$ and $\Phi(r - \lg\lg(\kappa))$.

Definition 7.11 (Noise distribution Φ)**.** We define $\Phi(r)$ as follows:

- Choose H_1 and H_2 randomly.
- Let $i^* \in [u]$ be the index to the bundle that **IsAGoodBundle** outputs SUCCESS.
- For $r \in [0, k_v]$, output $D_{T,r}(\mathcal{D}_{2,i^*})$, where $T = \mathsf{Avail}_G(K_{\theta,i^*}, R \setminus R')$.
- For $r \notin [0, k_v]$, $\Phi(r) := 0$

Lemma 7.12. The hockey stick divergences $\mathsf{D}^{\mathsf{hs}}_{e^\epsilon}(\Phi(r), \Phi(r - \lg\lg(\kappa)))$ and $\mathsf{D}^{\mathsf{hs}}_{e^\epsilon}(\Phi(r - \lg\lg(\kappa)), \Phi(r))$ are both negligible in κ.

Proof. For brevity, for any r, denote $D_r := D_{T,r}(\mathcal{D}_{2,i^*})$. Conditioned on **IsAGoodBundle** outputting SUCCESS with input $\mathbf{out}_{-i^*}$, we have a and b such that for $r \in [a + \lg\lg\kappa, b]$,

$$e^{-\epsilon} \leq \frac{e^{-\epsilon/3} E^{n'}_{k_v,r}}{e^{\epsilon/3} E^{n'}_{k_v,r-\lg\lg(\kappa)}} \leq \frac{D_r}{D_{r-\lg\lg(\kappa)}} \leq \frac{e^{\epsilon/3} E^{n'}_{k_v,r}}{e^{-\epsilon/3} E^{n'}_{k_v,r-\lg\lg(\kappa)}} \leq e^{\epsilon}.$$

The first and last inequalities are from Corollary 5.11. The second and third inequalities are from the condition that the process outputs SUCCESS. The hockey stick divergence $\mathsf{D}^{\mathsf{hs}}_{e^\epsilon}(\Phi(r), \Phi(r - \lg\lg(\kappa)))$ is therefore at most

$$\sum_{r \notin [a+\lg\lg\kappa, b]} D_r \leq k_v \cdot \mathsf{negl}(\kappa) = \mathsf{negl}(\kappa). \qquad \blacksquare$$

Similarly, $\mathsf{D}^{\mathsf{hs}}_{e^\epsilon}(\Phi(r - \lg\lg(\kappa)), \Phi(r))$ is also $\mathsf{negl}(\kappa)$.

Putting it all together. Let c be the final count produced by running protocol π_{NMH}. We consider the probabilities

$$\Pr_{H_1,H_2,\mathcal{D}}[c \mid B_{+x^*}] \quad \text{and} \quad \Pr_{H_1,H_2,\mathcal{D}}[c \mid B].$$

We consider only runs of the protocol that yield c and for which there exists some $i^* \in [u]$ such that the process **IsAGoodBundle** returns SUCCESS given $\mathbf{out}_{-i^*}$ as input. We just have argued that such an i^* exists with all but negligible probability.

Further, we consider only runs of the protocol for which $out^{+}_{bad} - out_{bad} \leq s = \lg\lg(\kappa)$. By Lemma 5.13, this also occurs with all but negligible probability, We will also leak $k_v = |\mathsf{Avail}(K_{\theta,i^*}, R \setminus R')|$.

Conditioned on the above events, by the definition of the distribution Φ, the value out_{i^*} contributes $(k_v - r)$ to the final count c with probability $p = \Phi(r)$. Recall that every iteration j in K_{θ,i^*} is good, which means $\min h_j(A) = \min h_j(I)$, potentially contributing to the output.

Therefore, assuming none of bad events occur (which happens with overwhelming probability), by applying Lemma 7.12, the probability that the ratio of probabilities of a certain output out for B_{+x^*} and B is not contained in $[e^{-\epsilon}, e^{\epsilon}]$ is $\mathsf{negl}(\kappa)$, and therefore we conclude that the protocol satisfies the DDP security.

7.3.6 Proof of Lemma 5.13

When considering the probability of $D_{T,r}(\mathcal{D})$ and $I_{R',T,r}$ over the choice of H_2, the identity of T doesn't matter except for its size $k_b = |T|$. Therefore, in this case, we will simply use $D_{k_b,r}(\mathcal{D})$ and $I_{R',k_b,r}$ Moreover, when it is clear from the context, we will sometimes omit k_b and $\mathcal{D}$ and say $E^{R'}_r = E^{n'_R}_r$ and $I_{R',r} = I_{R',k_b,r}$, and $D_r = D_{k_b,r}(\mathcal{D})$.

We first show the following lemma holds.

Lemma 7.13. Let $\mathcal{D}$ be a distribution over sets of size n'_R with geometric collision property. Fix H_1 and consider k_b, θ, a, b specified in Lemma 5.11 with the same requirements. Then, we have the following:

Case 1: If $r \notin [a+s, b]$, we have $\Pr_{H_2}[D_{k_b,r}(\mathcal{D}) \leq \mathsf{negl}(\kappa)] \geq 1 - \mathsf{negl}(\kappa)$.

Case 2: If $r \in [a, b]$, then we have

$$\Pr_{H_2}\left[e^{-\epsilon/3} E_{k_b,r}^{n'_R} \leq D_{k_b,r}(\mathcal{D}) \leq e^{\epsilon/3} E_{k_b,r}^{n'_R}\right] \geq 1 - (e^{\epsilon/3} - 1)^{-2} \cdot \frac{16 \lg^3(\kappa)}{\sqrt{n_R}}.$$

Then, Lemma 5.13 follows by taking a union bound over different cases of $r \in [k_b]$.

Proof of Lemma 7.13

We also define $\rho(R') := \Pr_{R' \sim \tilde{\mathcal{D}}}[R']$.

Proof for Case 1. We first consider Case (1). By applying the Case (1) of Corollary 5.11, we have $E_r^{n'_R} \in \mathsf{negl}(\kappa)$. Given $E_r^{n'_R} \in \mathsf{negl}(\kappa)$, we show

$$\Pr_{H_2}[D_r(\mathcal{D}) \leq \mathsf{negl}(\kappa)] \geq 1 - \mathsf{negl}(\kappa).$$

Recall that $D_r(\mathcal{D}) = \sum_{R'} \rho(R') \cdot I_{R',r}$. Assume toward the contradiction that the negation of the statement holds. This means there are polynomials p and q, and a collection Heavy of R's such that

$$\Pr_{H_2}\left[\sum_{R' \in \mathsf{Heavy}} \rho(R') \cdot I_{R',r} \geq 1/p(\kappa)\right] \geq 1/q(\kappa).$$

The above implies that $\sum_{R' \in \mathsf{Heavy}} \rho(R') \geq 1/p(\kappa)$. Now, since $\mathcal{D}$ and H_2 are independent, we have $\sum_{R' \in \mathsf{Heavy}} \rho(R') \Pr_{H_2}[I_{R',r}] \geq \frac{1}{p(\kappa) q(\kappa)}$. However, considering that $\Pr_{H_2}[I_{R',r}] = E_r^{n'_R}$, which is negligible, the above is a contradiction.

Proof for Case 2. We will bound $D_r = \sum_{R'} \rho(R') \cdot I_{R',r}$ using Chebyshev inequality. For this, we would like to bound the variance of D_r.

We start with showing the following lemma, which will allow us to ignore the tail when we bound the variance. Below, the value z will correspond to the size of the intersection of the two sets R'_i and R'_j.

Lemma 7.14. Fix H_1. Consider a graph $G \leftarrow \mathbf{MinhashG}_{H_1}(A, I, x^*, H_2)$. Consider any set T of iterations in G such that $|T| = k_b$. Let Z be a set of left nodes in G such that $|Z| \leq n'_R$. Let $z = |Z|$. Consider the probability (over the choice of H_2) that Z has more than $z \lg\lg\kappa$ outgoing edges in G. This probability is negligible in κ.

Proof. Let $p = 1 - (\eta_{-\theta})^{1/n'_R}$. We first show that $p \leq 1/n'_R$. Recall $\theta \leq 1/10$, which implies $e^{-1} \leq 1/2 - \theta = \eta_{-\theta}$. Therefore, we have $(1 - 1/n'_R)^{n'_R} \leq e^{-1} \leq \eta_{-\theta}$, so $1 - 1/n'_R \leq (\eta_{-\theta})^{1/n'_R}$. Therefore, we have $p = 1 - (\eta_{-\theta})^{1/n'_R} \leq 1/n'_R$.

Let $\mathsf{Edges}(Z, T)$ be the set of edges from Z to T. Over the choice of H_2, the probability that each pair in $Z \times T$ forms an edge is at most p. Therefore, we can simply use a Binomial distribution to bound the probability. In particular, with $t = \lg\lg\kappa$ we have

$$\Pr_{H_2}\left[|\mathsf{Edges}(Z,T)| \geq zt\right] \leq \Pr\left[\mathrm{B}(z\hat{k}, p) \geq zt\right] \leq \binom{z\hat{k}}{zt} \cdot p^{zt} \leq \binom{z\hat{k}}{zt} \cdot \left(\frac{1}{n'_R}\right)^{zt} \leq \left(\frac{e\hat{k}}{tn'_R}\right)^{zt}$$

Since n'_R is much larger than k_b, the above probability becomes negligible in κ. ∎

Now we prove the following lemma towards bounding the variance of D_r.

Lemma 7.15. Fix H_1. We set the parameters for k_b, a and b as stated in Lemma 5.13. Let R'_i, R'_j be sets of nodes on the left of size n'_R such that with $|R'_i \cap R'_j| = z$. Let $\zeta = z \lg\lg\kappa$. Then for all $a \leq r \leq b$, we have

$$\Pr_{H_2}[I_{R'_i,r} \wedge I_{R'_j,r}] = \mathop{\mathbb{E}}_{H_2}[I_{R'_i,r} \cdot I_{R'_j,r}] \leq \left(1 + \frac{\zeta \cdot (e^{\zeta\epsilon/3} + 1)}{{\eta_{-\theta}}^{zk_b/n'_R}}\right) \left(E_r^{n'_R}\right)^2$$

Proof. Fix R'_i, R'_j with $|R'_i \cap R'_j| = z$. Let $Z = R'_i \cap R'_j$ and $X = R'_i - Z$. Then, we have

$$\begin{aligned}
\Pr_{H_2}[I_{R'_i,r} \wedge I_{R'_j,r}] &= \sum_{m=0}^{r} \Pr[I_{X,m} \wedge I_{Z,r-m} \wedge I_{R'_j,r}] \\
&\leq \sum_{m=0}^{r-\zeta} \Pr[I_{Z,r-m}] + \sum_{m=r-\zeta+1}^{r} \Pr[I_{X,m} \wedge I_{R'_j,r}] \\
&= \sum_{m=\zeta}^{r} \Pr[I_{Z,m}] + \sum_{m=r-\zeta+1}^{r} \Pr[I_{X,m}] \cdot \Pr[I_{R'_j,r}] \\
&\leq \mathsf{negl}(\kappa) + \sum_{m=r-\zeta+1}^{r} \Pr[I_{X,m}] \cdot \Pr[I_{R'_j,r}] \\
&= \mathsf{negl}(\kappa) + E_r^{n'_R} \cdot \sum_{m=r-\zeta+1}^{r} \Pr[I_{X,m}].
\end{aligned}$$

The second inequality holds due to Lemma 7.14.

It is left to bound $\Pr[I_{X,m}]$ for $m \in (r - \zeta, r]$. We observe that $\Pr_{H_2}[I_{X,m}] = \Pr[I_{R'_i,m} | \; I_{Z,0}]$. In other words, the event that X contributes to noise pattern m is equivalent to the event that R'_i contributes to m conditioned on the intersection having no contribution. Therefore, we have

$$\Pr_{H_2}[I_{X,m}] = \frac{\Pr[I_{R'_i,m} \wedge I_{Z,0}]}{\Pr[I_{Z,0}]} \leq \frac{\Pr[I_{R'_i,m}]}{{\eta_{-\theta}}^{zk_b/n'_R}} = \frac{E_m^{n'_R}}{{\eta_{-\theta}}^{zk_b/n'_R}}.$$

We now bound $E_m^{n'_R}$ for $m \in (r-\zeta, r]$. Let $m^* := \arg\max_m\{E_m^{n'_R} : m \in (r-\zeta, r]\}$. Using Corollary 5.11 we have $E_{m^*}^{n'_R} \leq (e^{\epsilon/3})^{\zeta} \cdot E_r^{n'_R} + \mathsf{negl}(\kappa)$. Therefore, we have

$$\begin{aligned}
\Pr_{H_2}[I_{R'_i,r} \wedge I_{R'_j,r}] &\leq \mathsf{negl}(\kappa) + E_r^{n'_R} \cdot \sum_{m=r-\zeta+1}^{r} \Pr[I_{X,m}] \;\leq\; \mathsf{negl}(\kappa) + \zeta \cdot E_r^{n'_R} \cdot \Pr[I_{X,m^*}] \\
&= \mathsf{negl}(\kappa) + \zeta \cdot E_r^{n'_R} \cdot \frac{E_{m^*}^{n'_R}}{\eta_{-\theta}{}^{zk_b/n'_R}} = \mathsf{negl}(\kappa) + \zeta \cdot E_r^{n'_R} \cdot \frac{e^{\zeta\epsilon/3} E_r^{n'_R} + \mathsf{negl}(\kappa)}{\eta_{-\theta}{}^{zk_b/n'_R}} \\
&\leq \left(1 + \frac{\zeta \cdot (e^{\zeta\epsilon/3}+1)}{\eta_{-\theta}{}^{zk_b/n'_R}}\right)\left(E_r^{n'_R}\right)^2 \qquad \blacksquare
\end{aligned}$$

We set the parameters for H_1, k_b, a and b as stated in Lemma 5.13. Let $\mathcal{D}$ be a distribution with the geometric collision property. Then, we show that for every $a \leq r \leq b$, we have

$$\mathsf{Var}_{H_2}[D_r] \leq \frac{16\lg^3(\kappa)}{\sqrt{n_R}}\left(E_{k_b,r}^{n'_R}\right)^2.$$

Consider any $r \in [a, b]$. Recall that $D_r := \sum_{R' \in \mathsf{Supp}(\tilde{\mathcal{D}})} \rho(R') \cdot I_{R',r}$.

$$\begin{aligned}
\mathsf{Var}_{H_2}[D_r] \quad &= \sum_{R'_i,R'_j} \rho(R'_i) \cdot \rho(R'_j) \cdot (\mathbb{E}[I_{R'_i,r} \cdot I_{R'_j,r}] - \mathbb{E}[I_{R'_i,r}] \cdot \mathbb{E}[I_{R'_j,r}]) \\
&\leq \sum_{R'_i,R'_j : |R'_i \cap R'_j| \geq 1} \rho(R'_i) \cdot \rho(R'_j) \cdot \mathbb{E}[I_{R'_i,r} \cdot I_{R'_j,r}] \\
&= \sum_{z=1}^{n'_R} \Pr_{R'_i,R'_j \sim \mathcal{D}}[|R'_i \cap R'_j| = z] \cdot \mathbb{E}[I_{R'_i,r} \cdot I_{R'_j,r}] \\
&\leq \sum_{z=1}^{n'_R} \left(\frac{1}{\sqrt{n_R}}\right)^z \cdot \left(1 + \frac{\zeta \cdot (e^{\zeta\epsilon/3}+1)}{\eta_{-\theta}{}^{zk/n'_R}}\right) \cdot \left(E_r^{n'_R}\right)^2 \\
&\leq \sum_{z=1}^{n'_R} \left(\frac{1}{\sqrt{n_R}}\right)^z \cdot \left(\zeta \cdot \frac{e^{\zeta}+2}{(2/5)^{\zeta/3}}\right) \cdot \left(E_r^{n'_R}\right)^2 \\
&\leq \sum_{z=1}^{n'_R} \left(\frac{1}{\sqrt{n_R}}\right)^z \cdot (8^{\zeta+1}) \cdot \left(E_r^{n'_R}\right)^2
\end{aligned}$$

The first inequality holds because if R'_i are R'_j are disjoint, then $I_{R'_i,r}$ and $I_{R'_j,r}$ are independent over the choice of H_2, and the relevant terms are canceled out. The second inequality is due to the geometric collision property of $\mathcal{D}$ and Lemma 7.15. The third inequality holds with $\epsilon \leq 3$ since $\theta < 1/10$ and k_b is much smaller than

n'_R. Therefore, we have $\mathsf{Var}_{H_2}[D_r] \leq 8 \cdot \left(E_r^{n'_R}\right)^2 \cdot \sum_{z=1}^{n'_R} \left(\frac{\lg^3 \kappa}{\sqrt{n_R}}\right)^z \leq \frac{16 \lg^3 \kappa}{\sqrt{n_R}} \left(E_r^{n'_R}\right)^2$. ■

Finally, by Chebyshev, we have that for all $a \leq r \leq b$,

$$\Pr_{H_2}\left[D_r \notin [e^{-\epsilon/3}(E_{k_b,r}^{n'_R}), e^{\epsilon/3}(E_{k_b,r}^{n'_R})]\right] \leq \Pr\left[|D_r - E_{k_b,r}^{n'_R}| \geq (1 - e^{-\epsilon/3}) \cdot E_{k_b,r}^{n'_R}\right]$$
$$\leq \frac{\mathsf{Var}[D_r]}{(1-e^{-\epsilon/3})^2 \cdot (E_{k_b,r}^{n'_R})^2} \leq \frac{16 \lg^3(\kappa)}{(1-e^{-\epsilon/3})^2 \sqrt{n_R}}.$$

7.3.7 Strong Chain Rule

Strong chain rule for a special case: achieving flatness through clustering. Fortunately, a stronger version of the chain rule is known to hold for a special leakage pattern, i.e., when elements are conditioned *in order* [164]; very roughly speaking, for every i, the min-entropy of $R_i|(R_1, \ldots, R_{i-1})$ is essentially the same as the min-entropy of $(R_1, \ldots, R_i)$ minus the min-entropy of $(R_1, \ldots, R_{i-1})$ at the sacrifice of an additional small leakage, which is called a *spoiling leakage.*

They achieve this by grouping possible sequences with a similar distributional characteristic into the same cluster. Then, in every cluster, the distribution of sequences conditioned on that cluster will be essentially flat. Now, the spoiling leakage corresponds to the cluster identifier. By making every cluster contain sufficiently many sequences (leading to sufficient min-entropy due to flatness), the total number of clusters can be small (leading to a short spoiling leakage).

Notes on notations. For brevity, in this section, we omit the subscript from n_R, i.e., we denote $n = n_R$. For any sequence of random variables $R = R_1, \ldots, R_n$ (for the secret input R), we denote $R_{<i} = R_1, \ldots, R_{i-1}$ and $R_{\leq i} = R_1, \ldots, R_i$. Likewise, we extend such subscript notations and use $R_{>i}$ and $R_{\geq i}$. We use lower case $r = r_1, \ldots, r_n$ to denote the actual set/sequence.

Strong chain rule for our setting. We first adapt the result in [164] into our setting. Then, we argue that a sufficient number of elements still have high min-entropy, even conditioned on the previous elements. Finally, we show that these high min-entropy (conditioned) elements provide the geometric collision property.

Theorem 7.16 (Block structures with few bits spoiled in our setting)**.** We consider a min-hash graph $G = (\mathcal{X}, \mathcal{Y}, \mathcal{E})$ constructed from $\mathbf{MinhashG}_{H_1}(A, I, x^*, H_2)$, while focusing on a single bundle $K_{\theta,*}$ of iterations.

Let $\mathcal{U} = U_1 \times \cdots \times U_n$ be a fixed universe and $R = (R_1, \ldots, R_n)$ be a sequence of (possibly correlated) random variables where each R_i is over U_i (and all are disjoint) and $|U_i| = \ell$ for all i. Then, for any $\epsilon \in (0, 1)$ and any $\delta > 0$, there exists a spoiling leakage function $f_G(R)$ that satisfies the following properties.

1. It holds that $\Pr_R[f(R) = \perp] \leq \epsilon n$.

2. Let $Im(f)$ be the set of images of f. Every $y \in Im(f) \setminus \{\perp\}$ specifies two disjoint sets V and W such that $V \cup W = [n]$.
3. Conditioned on any $y \in Im(f) \setminus \{\perp\}$, for every $i \in V$, every element in distribution $R_i \mid R_{<i}$ has low probability weight, i.e.,

$$\forall y \in Im(f) \setminus \{\perp\}, \forall r \text{ s.t. } f(r) = y, \forall i \in V : \quad \Pr\left[R_i = r_i \;\middle|\; R_{<i} = r_{<i},\ y\right] \leq \frac{2^\delta}{n^{1.5}}.$$

4. Conditioned on any $y \in Im(f) \setminus \{\perp\}$, for every $i \in W$, it holds that $R_i \mid R_{<i}$ has small support size, i.e.,

$$\forall y \in Im(f) \setminus \{\perp\}, \forall r \text{ s.t. } f(r) = y, \forall i \in W :$$
$$|\{r_i : \Pr[R_i = r_i | R_{<i} = r_{<i}, y]] \geq 0\}| \leq 2^\delta \cdot n^{1.5}.$$

5. $|Im(f)| \leq n \cdot (2e)^{n/2} \cdot \frac{(n+k_b)!}{n!} \cdot (2(\lg(\ell) + \lg(1/\epsilon))/\delta)^n$.
6. $\mathsf{Avail}_G(K_{\theta,*}, R_W)$ can be computed from $f(R)$, where $R_W := \{R_i : i \in W\}$.

Proof of Theorem 7.16
By following the general idea of [164], we will build clusters, and the spoiling leakage will be the cluster identifier. However, we will slightly change the way we build clusters.

Condition 1. Throughout our proof, we let $\Pr[r_i]$ denote $\Pr[R_i = r_i]$ for brevity, whenever the referred random variable is clear. Before forming the clusters, we will first like to exclude all sequences $r \in \mathcal{U} = U_1 \times \cdots \times U_n$ having a very small probability $\Pr_R[R_i = r_i \mid R_{<i} = r_{<i}] < \epsilon/\ell$ for any $i \in [n]$ and only consider the remaining $\mathcal{U}' \subset \mathcal{U}$. Specifically, we let $f(r) = \perp$ for all $r \notin \mathcal{U}'$. As we will see later, this probability lower bound is vital to upper bound $|Im(f)|$.

Claim 7.17. Let $\mathcal{U}'$ be the set containing all the sequences r such that $\Pr_R[R_i = r_i \mid R_{<i} = r_{<i}] \geq \epsilon/\ell$ for all $i \in [n]$. Then, we have $\Pr[r \in \mathcal{U}'] \geq 1 - \epsilon n$.

Proof. For each $i \in [n]$, and any $r_{<i} \in U_1 \times \cdots \times U_{i-1}$, we have

$$\sum_{u \in U_i : \Pr_R[R_i = u | R_{<i} = r_{<i}] < \epsilon/\ell} \Pr_R[R_i = u \mid R_{<i} = r_{<i}] < \sum_{u \in U_i} \epsilon/\ell = \epsilon.$$

Therefore, using a union bound across all $i \in [n]$, we have $\Pr[r \notin \mathcal{U}'] \leq \epsilon \cdot n$. ■

Building clusters. For each $r \in \mathcal{U}'$, we describe how to compute $f(r) = (f_1(r), f_2(r), \ldots, f_n(r))$, which will serve as the cluster identifier. Let $\mathsf{r}(a)$ denote a rounding function that rounds a to the closest multiple of $\delta/2$. We say $a \approx_{\mathsf{r}} a'$ if $\mathsf{r}(a) = \mathsf{r}(a')$.

For each r, do the following:

1. Let $f_{>n}(r) = \bot$ for any r, and initialize $W = \emptyset$.
2. For $i = n, \ldots, 1$, do the following:
   (a) Let $\textsc{sp}_i^1(r)$ denote the surprise of the ith element of r. More formally,
   $$\textsc{sp}_i^1(r) = -\lg \Pr_R[R_i = r_i \mid R_{<i} = r_{<i},\ f_{>i}(R) = f_{>i}(r)].$$
   This surprise measure represents how rare and surprising the event r_i is, conditioned on $r_{<i}$, $f_{>i}(r)$. In a sense, we will group sequences with similar surprises into a cluster.
   (b) Let $\textsc{sp}_i^2(r)$ denote the surprise of the sequences with a similar surprise level in aggregate.
   $$\textsc{sp}_i^2(r) = -\lg \Pr_R[\textsc{sp}_i^1(R) \approx_{\mathsf{r}} \textsc{sp}_i^1(r) \mid R_{<i} = r_{<i},\ f_{>i}(R) = f_{>i}(r)].$$
   Note $\textsc{sp}_i^1(r) \geq \textsc{sp}_i^2(r)$, since at least sequence r has $\textsc{sp}_i^1(r)$ and possibly more points may approximately share the surprise. Note also that $\textsc{sp}_i^2(r)$ is a deterministic function of $\textsc{sp}_i^1(r), r_{<i}$, $f_{>i}(r)$.
   (c) If $\mathsf{r}(\textsc{sp}_i^1(r)) - \mathsf{r}(\textsc{sp}_i^2(r)) \geq 1.5\lg(n)$ then let $f_i(r) = (\mathsf{r}(\textsc{sp}_i^1(r)), \mathsf{true})$.
   (d) Otherwise, let $f_i(r) = (\mathsf{r}(\textsc{sp}_i^1(r)), \mathsf{false}, H_i)$ and add i to W. Here, H_i is defined as $N(\{r_i\}) \setminus N(r_W)$, where N refers to the neighbors (restricted to $K_{\theta,*}$) of the input set of nodes in G. In other words, H_i contains the iterations newly covered by element r_i; any iterations previously covered by r_W are ruled out in H_i. In this way, we can reduce the length of the cluster identifier.
3. Set $f(r) = f_1(r), \ldots, f_n(r)$. Set $V = [n] \setminus W$.

Conditions 2 and 3. Condition 2 follows from how V is computed in step 3. We now show that condition 3 holds. In particular, $\forall y \in Im(f(\cdot)) \setminus \{\bot\}, \forall r$ s.t. $f(r) = y, \forall i \in V$ we have

$$\Pr[r_i \mid r_{<i}, y] = \Pr[r_i \mid r_{<i}, y_{\geq i}] = \frac{\Pr[r_i \wedge r_{<i} \wedge y_{\geq i}]}{\Pr[r_{<i} \wedge y_{\geq i}]} = \frac{\Pr[r_i \wedge r_{<i} \wedge y_{>i}]}{\Pr[r_{<i} \wedge y_{>i}]\Pr[y_i \mid r_{<i} \wedge y_{>i}]}$$

The first equality is due to $y_{<i}$ being a deterministic function of $r_{<i}$, $y_{\geq i}$. Similarly, the nominator of the final fraction is due to y_i being a deterministic function of $r_{\leq i}$, $y_{>i}$. Moreover, $y_{i,2}$ (i.e., true) can be deterministically computed from $y_{i,1}$(i.e., $\mathsf{r}(\textsc{sp}_i^1(r))$), $r_{<i}, y_{>i}$. Therefore, the above is equal to

$$\frac{\Pr[r_i \wedge r_{<i} \wedge y_{>i}]}{\Pr[y_{i,1} \mid r_{<i} \wedge y_{>i}]\Pr[r_{<i} \wedge y_{>i}]} = \frac{\Pr[r_i \mid r_{<i} \wedge y_{>i}]}{\Pr[y_{1,i} \mid r_{<i} \wedge y_{>i}]} = \frac{2^{-\textsc{sp}_i^1(r)}}{2^{-\textsc{sp}_i^2(r)}} \leq \frac{2^\delta}{n^{1.5}}. \quad (10)$$

The last inequality holds since $i \in V$, $\mathsf{r}(\textsc{sp}_i^1(r)) - \mathsf{r}(\textsc{sp}_i^2(r)) \geq 1.5\lg(n)$.

Condition 4. For r, y, i as quantified in the theorem statement, we have

$$\begin{aligned}
&|\{r_i : \Pr[R_i = r_i | R_{<i} = r_{<i}, y]] \geq 0\}| \\
&= |\{r_i : \Pr[R_i = r_i \wedge R_{<i} = r_{<i} \wedge y]] \geq 0\}| \\
&= |\{r_i : \Pr[R_i = r_i \wedge R_{<i} = r_{<i} \wedge y_{i,1}, y_{i,2}, y_{>i}]] \geq 0\}| \\
&\leq |\{r_i : \Pr[R_i = r_i | R_{<i} = r_{<i}, y_{i,1}, y_{i,2}, y_{>i}]] \geq 0\}|
\end{aligned}$$

By a similar argument as above, for all r_i s.t. $\Pr[R_i = r_i | R_{<i} = r_{<i}, y_{i,1}, y_{i,2}, y_{>i}]] \geq 0$, it holds

$$\Pr[R_i = r_i | R_{<i} = r_{<i}, y_{i,1}, y_{i,2}, y_{>i}] = \frac{\Pr[r_i \mid r_{<i} \wedge y_{>i}]}{\Pr[y_{i,1} \mid r_{<i} \wedge y_{>i}]} = \frac{2^{-\textsc{sp}_i^1(r)}}{2^{-\textsc{sp}_1^2(r)}} \geq \frac{2^{-\delta}}{n^{1.5}}$$

where the inequality holds since $i \in W$, we know that $\mathsf{r}(\textsc{sp}_i^1(r)) - \mathsf{r}(\textsc{sp}_i^2(r)) \leq 1.5 \lg(n)$. This means that $|\{r_i : \Pr[R_i = r_i | R_{<i} = r_{<i}, y_{\geq i}]] \geq 0\}| \leq 2^{\delta} \cdot n^{1.5}$.

Condition 5. To bound $|Im(f)|$, we first upper bound $y_{i,1}$. Recall that $\Pr_R[R_i = r_i \mid R_{<i} = r_{<i}] \geq \epsilon/\ell$ for all $i \in [n]$ and $r \in \mathcal{U}'$. Therefore, $\Pr_R[R_i = r_i \mid R_{<i} = r_{<i}, y] \geq \epsilon/\ell$ for all $i \in [n]$, and $\forall r$ such that $f(r) = y$.

Therefore, for all $r \in \mathcal{U}', i \in [n]$, we have $\textsc{sp}_i^1(r) \leq \lg(\ell) + \lg(1/\epsilon)$, which implies that $y_{i,1}$ has at most $2(\lg(\ell)) + \lg(1/\epsilon))/\delta$ different possibilities.

To upper bound the number of possibilities of the remaining parts, it suffices to upper bound the number of choices for set W of size m, as well as the number of possibilities for H_i's in each slot $i \in W$. Clearly, the former is $\binom{n}{m}$. For the latter part, note that each iteration appears at most once over all m slots. Therefore, the problem becomes how we can assign k_b different iterations into $m + 1$ positions (with some positions possibly containing none) while assigning them to the $m + 1$th position when they never appear in any slot of W. This is a well-known problem of stars and bars with $m + 1$ variables and sum k_b, which has $\binom{m+k_b}{m}$ possibilities. Since we have $k_b!$ different orderings for k_b iterations, the upper bound is $\binom{m+k_b}{m} \cdot (k_b!)$. We have:

$$\begin{aligned}
|Im(f)| &\leq (2(\lg(\ell) + \lg(1/\epsilon))/\delta)^n \left(\sum_{m=0}^{n} \binom{n}{m} \binom{m + k_b}{m} \cdot k_b! \right) \\
&= (2(\lg(\ell) + \lg(1/\epsilon))/\delta)^n \left(\sum_{m=0}^{n} \binom{n}{m} \frac{(m + k_b)!}{m!} \right) \\
&\leq n \cdot \binom{n}{n/2} \cdot \frac{(n + k_b)!}{n!} \cdot (2(\lg(\ell) + \lg(1/\epsilon))/\delta)^n \\
&\leq n \cdot (2e)^{n/2} \cdot \frac{(n + k_b)!}{n!} \cdot (2(\lg(\ell) + \lg(1/\epsilon))/\delta)^n .
\end{aligned}$$

Condition 6. Finally, condition 6 follows from the definition of the clustering procedure. In particular, $H_W = \bigcup_{i \in W} H_i$ contains all the iterations that r_W covers. The available set can be computed by $K_{\theta,*} \setminus H_W$. This concludes our proof.

Generalization
It can be seen that in the above proof, the only properties that we used of the additional leakage H_i is that for $i \in W$, H_i depends only on $R_i, y_{>i}$ and that the number of choices for the output of the sequence of leakages $[H_i]_{i\in W}$ is bounded by some B. Theorem 5.14 stated in Section 5.8.5 is restatement of Theorem 7.16 with respect to any such leakage function. Note that the leakage functions ℓ_i specified above can model leakage with respect to a random oracle h, by letting $\rho_i = h(R_i)$.

7.3.8 Proof of Lemma 7.10

For brevity, we denote $\mathcal{D} = \mathcal{D}_{2,i}$ and $\mathcal{D}_{\mathsf{leak}} = \mathcal{D}_{1,i}$ in the experiment $\mathsf{IsAGoodBundle}$. We show that $\mathcal{D}$ has the geometric collision property. In other words, we would like to show that when R is chosen uniformly at random from universe $\mathcal{U}$ then the distribution of these $n_R = n_B - n_I$ elements has the geometric collision property even with the leakage.

Towards this goal, by applying Theorem 7.16 to this distribution, we show that even with the leakage, there are at least $n_R/3$ elements that preserves enough min-entropy. We next show how these elements with sufficient min-entropy give the geometric collision property.

Remark 7.18 (Getting rid of tiny parts)**.** Similar to [164, Remark 2], we can further require that each cluster should have a probability that is "not too small". Therefore, we define a new leakage function f' by substituting the ϵ in the above theorem with $\epsilon/2$, and additionally letting $f'(r) =\perp$ for all r such that $y \in f(r)$ and $\Pr_R[f(R) = y] < \epsilon n/(2|Im(f)|)$ (their total probability is at most $\epsilon n/2$), we obtain the following: f' satisfies all conditions in Theorem 7.16. Additionally, $\forall y \in Im(f')$, we have $\Pr_R[f'(R) = y] \geq \epsilon n/(2|Im(f)|)$.

Fraction of blocks with high entropy. Using Theorem 7.16, with setting $\ell \geq 4n^3$ and assuming sufficient min-entropy of R, we first show that one can ensure more than 1/3 fraction of the blocks having min-entropy at least $1.5\lg(n)$, upon leaking the outcome of f' and all previous blocks.

First notice that with all but ϵn probability, $f'(R) \neq \perp$. Therefore, it suffices to let $\epsilon = 2^{-\kappa}$. Then, by setting $\delta = 1$, we have

$$\begin{aligned}
\lg(|Im(f)|) &\leq n \cdot (2e)^{n/2} \cdot (n + k_b)^{k_b} \cdot (2(\lg(\ell) + \lg(1/\epsilon))/\delta)^n \\
&= \Big(\lg(n) + n/2 \cdot \lg(2e)\Big) + k_b \cdot \lg(n + k_b) + n \cdot (1 + \lg(\lg(\ell) + \kappa)) \\
&< 3n/2 + 2k_b \lg n + n(2 + \lg \kappa) < 0.5n \lg n
\end{aligned}$$

for sufficiently large n with $k_b = \Omega(\kappa)$ and $n/k_b^2 = \Omega(\kappa)$.

Combining the above with Remark 7.18, we have $\Pr_{\mathcal{D}_{\mathsf{leak}}}[f'(R) = y] \geq \epsilon n/(2 \cdot 2^{0.5n\lg(n)})$ and for every $y \in Im(f') \setminus \{\perp\}$. Moreover, for every r such that $f'(r) = y$, we have

$$\Pr_{\mathcal{D}}[r] = \Pr_{\mathcal{D}_{\mathsf{leak}}}[r \mid y] = \frac{\Pr_{\mathcal{D}_{\mathsf{leak}}}[r \wedge y]}{\Pr_{\mathcal{D}_{\mathsf{leak}}}[y]} \leq \frac{2^{-(\frac{8n}{9}\lg \ell + n)}}{(\epsilon n/2)\cdot 2^{-0.5n\lg n}} = 2^{-(\frac{8}{9}\log \ell - 0.5\lg n + 1)\cdot n}\cdot(2/\epsilon n), \tag{11}$$

which suggests $\mathcal{D}$ has min-entropy at least $(\frac{8}{9}\log \ell - 0.5\lg n + 1)\cdot n - \lg(2/\epsilon n)$. We show that the following holds: The min-entropy of at least $n' = n/3$ blocks, *conditioned on the outcome of all prior blocks* as well as y, is at least $\lg(n^{1.5})$.

Towards a contradiction, assume otherwise. Let V be the set of blocks with min-entropy at least $\lg(n^{1.5})$ and let W be the set of blocks with min-entropy less than $\lg(n^{1.5})$ (as defined in Theorem 7.16). We will show that if $|V| \leq n/3$ there exists a point r in the support of $\mathcal{D}$ such that $\Pr_{\mathcal{D}}[r] > 2^{-(\frac{8}{9}\log \ell - 0.5\lg n + 1)\cdot n}\cdot(2/\epsilon n)$, contradicting the min-entropy of $\mathcal{D}$.

First, find any value r^*_V such that $\Pr_{\mathcal{D}}[R_V = r^*_V] \geq \frac{1}{\ell^{|V|}}$. Note that r^*_V must exist since the support size of R_V is at most $\ell^{|V|}$. Let $\mathsf{Supp}_W(r^*_V) = \{r : r_V = r^*_V \wedge \Pr_{\mathcal{D}}[R = r] > 0\}$. Then, we have $\Pr_{\mathcal{D}}[R \in \mathsf{Supp}_W(r^*_V)] = \Pr[R_V = r^*_V] \geq \frac{1}{\ell^{|V|}}$.

Second, we show that $|\mathsf{Supp}_W(r^*_V)| \leq (2\cdot n^{1.5})^{|W|}$. Consider any $r \in \mathsf{Supp}_W(r^*_V)$. Applying the fourth condition of Theorem 7.16 with $\delta = 1$, condition on any $y \in Im(f') \setminus \{\perp\}$, for any $i \in W$ and any fixing of $R_{<i} = r_{<i}$, the number of elements in the support of $R_i \mid r_{<i}$ is at most $2\cdot n^{1.5}$, which implies that $|\mathsf{Supp}_W(r^*_V)|$ must be at most $(2\cdot n^{1.5})^{|W|}$, since the positions for V are fixed to r^*_V.

Based on the above two arguments, by the averaging argument, there must be some $r^* \in \mathsf{Supp}_W(r^*_V)$ for which $\Pr_{\mathcal{D}}[R = r^*] \geq \frac{1}{(\ell)^{|V|}}\cdot\frac{1}{(2\cdot n^{1.5})^{|W|}}$. Therefore, we have

$$\begin{aligned}
-\lg \Pr_{\mathcal{D}}[r^*] &= |V|\lg(\ell) + |W|\lg(2n^{1.5}) = |V|\lg \ell + |W| + 1.5(n - |V|)\lg n \\
&\leq n + |V|\lg(\ell/n) + 1.5n\lg n \leq n + n/3\lg(\ell/n) + 1.5n\lg n \\
&= n + n/3\lg(\ell) - 1/3n\lg n + 1.5n\lg n,
\end{aligned}$$

where the second to last line follows assuming $|V| < n/3$.

To reach contradiction to (11), we require that

$$n + n/3\lg(\ell) - 1/3n\lg n + 1.5n\lg n \leq \left(\frac{8}{9}\lg \ell - 0.5\lg n + 1\right)\cdot n - \lg(2/\epsilon n).$$

The above is implied by $5/3n\lg n \leq 5/9n\lg \ell - \lg(2/\epsilon n)$.

When $\ell \geq 4n^3$ the above is implied by $5/3n\lg n \leq 5/3n\lg n + 10/3n - \lg(2/\epsilon n)$, which is true for $n \geq \lg(1/\epsilon) = \kappa$. Thus we reach contradiction to (11). We therefore conclude that $|V| \geq n/3$.

Geometric collision property. Note that we can equivalently view R' in the support of $\mathcal{D}$ as a set of size n', or as a stream of elements of length n', where the element in the i-th block (for $i \in [n']$) comes from universe U_i, and

$\{U_1, \ldots, U_{n'}\}$ are mutually disjoint. Taking the second view, given R', S' in the support of $\mathcal{D}$, we have that $|R' \cap S'| = z$ if and only if there exists some set $Z \subseteq [n']$ of size z such that (1) the *ordered* set of elements in the blocks of R' indexed by Z (denoted R'_Z) is equal to the *ordered* set of elements in the blocks of S' indexed by Z (denoted S'_Z) and (2) the set of elements in the blocks of R' indexed by $[n'] \setminus Z$ (denoted $R'_{\overline{Z}}$) and the set of elements in the blocks of S' indexed by $[n'] \setminus Z$ (denoted $S'_{\overline{Z}}$) are disjoint.

We are now ready to analyze the probability that $|R' \cap S'| = z$ for R', S' drawn from $\mathcal{D}$, and for $z \in [n']$:

$$\begin{aligned}\Pr_{R',S' \leftarrow \mathcal{D}}[|R' \cap S'| = z] &= \sum_{Z \subseteq [n'], |Z| = z} \Pr_{R',S' \leftarrow \mathcal{D}} \left[(R'_Z = S'_Z) \wedge \left(R'_{\overline{Z}} \cap S'_{\overline{Z}}) = \emptyset\right)\right] \\ &\leq \sum_{Z \subseteq [n'], |Z| = z} \Pr_{R',S' \leftarrow \mathcal{D}}[R'_Z = S'_Z] \quad \leq \sum_{Z \subseteq [n'], |Z| = z} \left(\frac{1}{n^{1.5}}\right)^z\end{aligned}$$

The second inequality holds since each element in the stream has min-entropy at least $\lg(n^{1.5})$. Therefore, we have $\Pr_{R',S' \leftarrow \mathcal{D}}[|R' \cap S'| = z] \leq \binom{n/3}{z} \cdot \left(\frac{1}{n^{1.5}}\right)^z \leq \left(\frac{1}{n^{0.5}}\right)^z$.

7.4 FACTS

7.4.1 Proofs for tail bound probabilistic analysis

We now complete the proofs of lemmas and theorems in Section 6.4.3.

We start with the following standard way to approximate numbers near 1 with exponentials.

Lemma 7.19. For any real constant $\alpha > 0$, and any real x with $0 < x \leq \alpha$, we have

$$\exp\left(-\tfrac{1}{\alpha} \ln \tfrac{1}{1-\alpha} \cdot x\right) \leq 1 - x < \exp(-x).$$

We also re-state this straightforward consequence of the Hoeffding/Chernoff bound on the sum of random variables:

Lemma 7.20. Let $X_1, \ldots, X_n$ be independent Poisson trials, and write $Y = \sum_i X_i$ for their sum. If $\mathbb{E}[Y] = \mu$, then for any $\delta > 0$, each of $\Pr(Y \geq \mu + \delta)$ and $\Pr(Y \leq \mu - \delta)$ are at most $\exp(-2\delta^2/n)$.

We now recall and prove the building-block lemmas from Section 6.4.3.

Lemma 6.2. Let x be an item such that at most τ of x's item slots are filled. If the CCBF parameters s, u, v satisfy $v \geq 7.042652\tau$ and $u \geq 0.5184846\frac{s}{\tau}$, then the probability that a call to $\mathsf{Increment}(x, C)$ fills in one more of x's item slots is at least 0.956414.

Proof. From (4), we know this probability is exactly $p_w = 1 - \frac{(s-u)^{\underline{w}}}{s^{\underline{w}}}$, where $w = v - \tau$ is the number of unfilled slots remaining. Using Lemma 7.19 we have

$\frac{(s-u)^{\underline{w}}}{s^{\underline{w}}} \le \left(1-\frac{u}{s}\right)^w \le \exp(-uw/s)$, which means that

$$p_w \ge 1 - \exp\left(-\frac{u(v-\tau)}{s}\right) = 1 - \exp\left(-\frac{u\tau}{s} \cdot \left(\frac{v}{\tau} - 1\right)\right).$$

Applying the two lower bounds on $\frac{u\tau}{s}$ and $\frac{v}{\tau}$ from the lemma statement yields the claimed result. ■

Lemma 6.3. Let x be any item. If the CCBF parameters s, u, v satisfy $371 \le v \le 0.00386s$ and $u \le 3.65151\frac{s}{v}$, then the probability that a call to $\mathsf{Increment}(x, C)$ fills in one more of x's item slots is at most 0.974876.

Proof. Using again (4), the probability is exactly $p_w = 1 - \frac{(s-u)^{\underline{w}}}{s^{\underline{w}}}$, where again $w \le v$ is the number of unfilled slots for item x. Then

$$\frac{(s-u)^{\underline{w}}}{s^{\underline{w}}} \ge \frac{(s-u)^{\underline{v}}}{s^{\underline{v}}} \ge \left(\frac{s-u-v+1}{s-v+1}\right)^v > \left(1 - \frac{u}{s-v}\right)^v.$$

Using upper bounds on $\frac{v}{s}$ and u from the lemma statement, we have

$$p_w < 1 - \left(1 - \frac{u}{s-v}\right)^v \le 1 - \left(1 - 3.66567\frac{1}{v}\right)^v.$$

Finally, the lower bound on v from the lemma statement shows $3.66567/v \le 0.00989$, and so we can finally use the lower exponential bound of Lemma 7.19 to obtain the stated result. ■

Lemma 6.4. Let s, u, v be CCBF parameters that satisfy the conditions of Lemma 6.3, and suppose m, t are integers such that $s \ge 96m$ and $v \le 7.409t$. Then the tipping point τ, for threshold t and with m total set bits in the table T, is at most $1.0520553t$.

Proof. The tipping point τ is the expected number of slots filled in the table if t of the m total calls to $\mathsf{Increment}$ were actually called on this particular item.

We can divide the calls to $\mathsf{Increment}$ into two groups: the t calls for item x, and the $m - t$ calls for other items. The expected number of slots within x's item set filled by the first group is at most $0.974876t$, from Lemma 6.3.

For the second group, these calls to $\mathsf{Increment}$ on unrelated items are distributed uniformly at random among all table indices, and so their expected fraction within this item set is the same as their overall fraction in the table. Therefore, the expected number of slots filled by calls to $\mathsf{Increment}$ on other items is at most

$$\frac{(m-t)v}{s} < \frac{mv}{s} \le \frac{7.409}{96}t.$$

By linearity of expectation, we can sum these two to obtain an upper bound on the total expected tipping point as given in the lemma statement. ■

Now we can proceed to the proofs of the main theorems on the accuracy of the CCBF.

Theorem 6.5. Let n be an upper bound on the total number of calls to Increment, and t be a desired threshold for TestCount. Suppose the parameters s, u, v for a CCBF data structure satisfy the conditions of Lemma 6.2, and furthermore that $v \leq 8t$. If the actual number of calls to Increment(x, C) is at most $t - 2.1\sqrt{\lambda t}$, then the probability TestCount(x, t) gives a false positive is at most $2^{-\lambda}$.

Proof. Let τ_t be the tipping point for any actual number $m \leq n$ of total set bits in the table T and for the given threshold t. And consider random variables $X_1, \ldots, X_v$ for the v slots assigned to item x, where each X_1 is 0 or 1 depending on whether the corresponding slot in table T is 0 or 1. We want to know the probability that the sum of the X_i's is at least τ_t, which is what would cause TestCount(x, t) to produce a false positive.

Let $k = t - 2.1\sqrt{\lambda t}$ be the actual number of calls to Increment on item x, and write τ_k for the tipping point at threshold k. By definition and the exact calculations for τ_k outlined earlier, we know that $\mathbb{E}[\sum X_i] = \tau_k$.

The difference between these two tipping points $\tau_t - \tau_k$ is the expected number of extra slots filled by $t - k$ calls to Increment, which from Lemma 6.2 is at least

$$0.956414(t-k) = 0.956414 \cdot 2.1\sqrt{\lambda t} \geq \sqrt{\tfrac{\lambda v}{2}},$$

where in the last step we used the upper bound on v from the assumptions of the theorem.

The variables X_i are not independent, but they are *negatively correlated*, meaning that the whenever one slot is filled, it only decreases the likelihood that another is filled; intuitively, this is because there are now fewer chances to fill the other slot. Therefore we can apply the Hoeffding bound in this direction (Lemma 7.20) to say that

$$Pr\left(\sum X_i \geq \tau_k + \sqrt{\tfrac{\lambda v}{2}}\right) \leq \exp(-\lambda),$$

as required. ■

Theorem 6.6. Let n be an upper bound on the total number of calls to Increment, and t be a desired threshold for TestCount. Suppose the parameters s, u, v for a CCBF data structure satisfy the conditions of Lemmas 6.2 and 6.4. If the actual number of calls to Increment(x, C) is at least

$$1.1t + .4\lambda + .7\sqrt{\lambda t}, \tag{6}$$

then the probability TestCount(x, t) gives a false negative is at most $2^{-\lambda}$.

Proof. Writing k for the actual number of complaints given in (6), we need a tail bound on the probability that, after k calls to Increment on the same item x, there are still fewer than $1.0520553t$ slots of x's item set filled in, where the latter constant comes from applying the upper bound on the tipping point from Lemma 6.4.

For this, we need a lower bound on the *expected* number of bits set after k calls to Increment on item x; from Lemma 6.2 this is at least $0.956414k$.

Now we can apply the Hoeffding bound (Lemma 7.20), with $\mu = 0.956414k$ and $\mu + \delta = 1.0520553t \geq \tau$ to see that the probability that less than τ bits of x's user set are flipped is at most

$$\begin{aligned}
&\exp\left(-2(1.0520553t - 0.956414k)^2/k\right) \\
\leq\, &\exp\left(-2\left(0.38\lambda + 0.66\sqrt{\lambda t}\right)^2/k\right) \\
\leq\, &\exp\left(-\frac{.28\lambda^2 + \lambda\sqrt{\lambda t} + .87\lambda t}{.4\lambda + .7\sqrt{\lambda t} + 1.1t}\right) \\
\leq\, &\exp(-.7\lambda) \leq 2^{-\lambda}.
\end{aligned}$$

■

8 Timeline

The following timeline outlines the completion of the remaining research items and the preparation of the final dissertation. As noted, sublinear secure protocol works are largely complete. The schedule focuses on the execution of the follow-up work on FACTS.

Period	**Key Objectives & Milestones**
Completed – Present	• Completed research and publication for Secure Search/Sampling (Sublinear-communication Secure MPC). • Completed research and publication for Privacy-preserving set similarity via Min-Hash. • Completed design, analysis, and evaluation of FACTS (Accountability tracking on End-to-End Encrypted Messaging Systems). • Follow-up Work to enable spreader tracking via new protocol designs (Accountability tracking on End-to-End Encrypted Messaging Systems).
Month 1 – Month 2	**Line 3 Implementation:** • Analyzing the system loads that our new design could handle and identify bottlenecks if any. • Estimating the actual cost of the new designed FACTS for spreaders with two non-colluding servers.
Month 1 – Month 3	**Line 3 Security and Privacy Analysis:** • Proof of security in malicious 2-party setting. • Compare performance against the original FACTS baseline. • *Milestone: Submit the paper to [TBD].*
Month 3 – Month 4	**Thesis Writing (Part I):** • Update Introduction and Literature Review based with feedback and further research. • Integrate all finished lines of work and write down the full chapter of thesis research results.
Month 5	**Thesis Writing (Part II) & Defense:** • Finish and integrate the results of the follow-up work of FACTS on spreader tracking results into the final dissertation. • Final review with advisor. • *Milestone: Thesis Defense.*

Table 2: Proposed timeline for the completion of the Ph.D. thesis.

9 Preliminary Results

As detailed in previous sections, the research objectives for our first line of work (Secure Search, Sampling, and Similarity) and the second line of work (the threshold traceback for originator tracking via FACTS) have been successfully met. The results have been published and evaluated.

Therefore, this section presents preliminary approach solely for our **proposed follow-up work of FACTS for messaging accountability tracking, targeting spreader instead of originator**: using *efficient secure 2-party computation with malicious security.* The next steps include experiments and cryptography analysis. The goal of these experiments is to demonstrate and justify that with the most state-of-art 2-party computation building blocks such as oblivious sorting, our design is practical and efficient with today's large scale of number of users and messages on End-to-End Encrypted messaging systems. The goal of the cryptography analysis is to deliver strict security and privacy guarantee while keeping the protocol practical and cost-friendly.

9.1 Microbenchmarks

Unlike the hash-based structures (CCBF) used in the original FACTS system, our proposed follow-up design runs by two non-colluding servers with malicious security. A primary concern is whether these secure 2-party computations introduces prohibitive data transfer or latency, which in turns hurts the capability of the design to handle the large scale of users and messages.

We benchmarked all major building blocks on two standard CloudLab servers with average network bandwidth and latency inside United States. Table 3 shows the major communication-intensive 2-party operations and their servers' communication cost.

9.2 Bandwidth Overhead Analysis

This new follow-up work focuses on the actual spreader of reported messages instead of the originator. A fundamental change of this is that we rely on non-colluding 2-party to make the design practical. With strong security guarantee, this often comes at the cost of increased communication overhead as well as computation cost. So far the main concern is the bandwidth and latency between two servers would set a limit of how many complaints we could handle in each epoch.

Since our design is sequential across different components and fully parallel within each component, to assess this feasibility, we micro-benchmarked each building blocks using the state-of-art protocols on 1 million reports per epoch.

Protocols	**Communication throughput**	**Communication rounds**
Poseidon Hash	21.4GB	400
Schnorr Signature Verification	1.1TB	2250
Oblivious Sorting	1.01TB	2610
Oblivious Deduplication	82.3GB	60

Table 3: Throughput and Round for each components in the follow-up FACTS for spreaders with one million complaints per epoch

9.3 Conclusion of Preliminary Data

These benchmarks indicate that while the proposed tools for this follow-up work are communicational heavier than the original FACTS primitives, they provide efficient and lightweight protocols for the large scale of popular E2EE messaging systems such as Signal. This confirms the capability of our design. We are also working on evaluation of end-to-end server runtime to show the actual cost of our design.

10 Conclusion

This proposal outlines a comprehensive research agenda addressing the tension between data privacy and utility, culminating in a rigorous examination of accountability in end-to-end encrypted systems.

Our first line of work has established efficient, privacy-preserving primitives for **Secure Search**, **Secure Sampling**, and **Privacy-Preserving Set Similarity via Min-Hash**, demonstrating that sublinear communication is achievable without compromising security and privacy guarantees. Our second line of work, **FACTS** and the ongoing follow-up work, focused on a more specific application, enabled threshold tracebacks of originator or spreader of reported messages on E2EE messaging systems, proving that accountability mechanisms can coexist with encryption, while still remain sublinear to the number of total messages but only number of complaints.

Together, these three lines of work will contribute a unified framework for **scalable, privacy-preserving protocols and applications**, providing both theoretical advancements and practical tools.

References

[1] Abbas Acar, Z Berkay Celik, Hidayet Aksu, A Selcuk Uluagac, and Patrick McDaniel. Achieving secure and differentially private computations in multiparty settings. In *2017 IEEE Symposium on Privacy-Aware Computing (PAC)*, pages 49–59. IEEE, 2017.

[2] Adi Akavia, Dan Feldman, and Hayim Shaul. Secure search on encrypted data via multi-ring sketch. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 985–1001. ACM Press, October 2018.

[3] Adi Akavia, Craig Gentry, Shai Halevi, and Max Leibovich. Setup-free secure search on encrypted data: Faster and post-processing free. *Proc. Priv. Enhancing Technol.*, 2019(3):87–107, 2019.

[4] Asra Ali, Tancrède Lepoint, Sarvar Patel, Mariana Raykova, Phillipp Schoppmann, Karn Seth, and Kevin Yeo. Communication–computation

trade-offs in pir. Usenix Security (To appear), 2021. Available at `https://ia.cr/2019/1483`.

[5] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Streaming algorithms via precision sampling. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 363–372. IEEE Computer Society, 2011.

[6] Sebastian Angel, Hao Chen, Kim Laine, and Srinath T. V. Setty. PIR with compressed queries and amortized query processing. In *2018 IEEE Symposium on Security and Privacy*, pages 962–979. IEEE Computer Society Press, May 2018.

[7] Gilad Asharov, Ilan Komargodski, Wei-Kai Lin, Kartik Nayak, Enoch Peserico, and Elaine Shi. OptORAMa: Optimal oblivious RAM. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 403–432. Springer, Heidelberg, May 2020.

[8] Vikas G. Ashok and Ravi Mukkamala. A scalable and efficient privacy preserving global itemset support approximation using bloom filters. In *Data and Applications Security and Privacy XXVIII - 28th Annual IFIP WG 11.3 Working Conference, DBSec 2014, Vienna, Austria, July 14-16, 2014. Proceedings*, pages 382–389, 2014.

[9] Idan Attias, Edith Cohen, Moshe Shechner, and Uri Stemmer. A framework for adversarial streaming via differential privacy and difference estimators. In *ITCS 2023*, pages 8:1–8:19. LIPIcs, January 2023.

[10] Martin Aumüller, Anders Bourgeat, and Jana Schmurr. Differentially private sketches for jaccard similarity estimation. In *SISAP 2020*, LNCS, pages 18–32. Springer, 2020.

[11] László Babai, Noam Nisan, and Mario Szegedy. Multiparty protocols and logspace-hard pseudorandom sequences (extended abstract). In *21st ACM STOC*, pages 1–11. ACM Press, May 1989.

[12] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.*, 68(4):702–732, 2004.

[13] Gilles Barthe and Federico Olmedo. Beyond differential privacy: Composition theorems and relational logic for f-divergences between probabilistic programs. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *ICALP 2013, Part II*, volume 7966 of *LNCS*, pages 49–60. Springer, Heidelberg, July 2013.

[14] Raef Bassily, Adam Groce, Jonathan Katz, and Adam Smith. Coupled-worlds privacy: Exploiting adversarial uncertainty in statistical data privacy. In *54th FOCS*, pages 439–448. IEEE Computer Society Press, October 2013.

[15] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Guha Thakurta. Practical locally private heavy hitters. *Advances in Neural Information Processing Systems*, 30, 2017.

[16] Raef Bassily and Adam Smith. Local, private, efficient protocols for succinct histograms. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 127–135, 2015.

[17] Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In *Crypto 2008*, pages 451–468. Springer, 2008.

[18] Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 451–468. Springer, Heidelberg, August 2008.

[19] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 535–552. Springer, Heidelberg, August 2007.

[20] Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. A framework for adversarially robust streaming algorithms. *J. ACM*, 69(2):17:1–17:33, 2022.

[21] 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.

[22] Laura Blackstone, Seny Kamara, and Tarik Moataz. Revisiting leakage abuse attacks. In *NDSS 2020*. The Internet Society, February 2020.

[23] Marina Blanton and Everaldo Aguiar. Private and oblivious set and multiset operations. Cryptology ePrint Archive, Report 2011/464, 2011. `https://eprint.iacr.org/2011/464`.

[24] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. The johnson-lindenstrauss transform itself preserves differential privacy. In *53rd FOCS*, pages 410–419. IEEE Computer Society Press, October 2012.

[25] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[26] Carlo Blundo, Emiliano De Cristofaro, and Paolo Gasti. Espresso: Efficient privacy-preserving evaluation of sample set similarity. *J. Comput. Secur.*, 22(3):355–381, 2014.

[27] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. Order-preserving symmetric encryption. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 224–241. Springer, Heidelberg, April 2009.

[28] Alexandra Boldyreva, Nathan Chenette, and Adam O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 578–595. Springer, Heidelberg, August 2011.

[29] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 506–522. Springer, Heidelberg, May 2004.

[30] Elette Boyle, Rio LaVigne, and Vinod Vaikuntanathan. Adversarially robust property-preserving hash functions. In *Innovations in Theoretical Computer Science Conference (ITCS)*, pages 16:1–16:20, San Diego, CA, USA, January 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[31] Russell Brandom. Apple says collision in child-abuse hashing system is not a concern. *The Verge*, August 2021.

[32] Vladimir Braverman, Rafail Ostrovsky, and Carlo Zaniolo. Optimal sampling from sliding windows. *J. Comput. Syst. Sci.*, 78(1):260–272, 2012.

[33] A. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences, International Conference on*, page 21. IEEE Computer Society, 1997.

[34] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syntactic clustering of the web. *Comput. Networks*, 29(8-13):1157–1166, 1997.

[35] J Brooks et al. Ricochet: Anonymous instant messaging for real privacy, 2016.

[36] D. Cantor and H. Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, 36:587–592, 1981.

[37] David Cash, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Highly-scalable searchable symmetric encryption with support for Boolean queries. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 353–373. Springer, Heidelberg, August 2013.

[38] Anrin Chakraborti and Radu Sion. Concuroram: High-throughput stateless parallel multi-client ORAM. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019.* The Internet Society, 2019.

[39] Jeffrey Champion, abhi shelat, and Jonathan Ullman. Securely sampling biased coins with applications to differential privacy. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 603–614. ACM Press, November 2019.

[40] T.-H. Hubert Chan, Kai-Min Chung, Bruce M. Maggs, and Elaine Shi. Foundations of differentially oblivious algorithms. In Timothy M. Chan, editor, *30th SODA*, pages 2448–2467. ACM-SIAM, January 2019.

[41] Melissa Chase and Seny Kamara. Structured encryption and controlled disclosure. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 577–594. Springer, Heidelberg, December 2010.

[42] Wei-Ning Chen, Ayfer Ozgur, and Peter Kairouz. The poisson binomial mechanism for unbiased federated learning with secure aggregation. In *ICML 2022*, volume 162, pages 3490–3506. PMLR, 2022.

[43] Jung Hee Cheon, Miran Kim, and Myungsun Kim. Optimized search-and-compute circuits and their application to query evaluation on encrypted data. *IEEE Trans. Inf. Forensics Secur.*, 11(1):188–199, 2016.

[44] Jung Hee Cheon, Miran Kim, and Kristin E. Lauter. Homomorphic computation of edit distance. In Michael Brenner, Nicolas Christin, Benjamin Johnson, and Kurt Rohloff, editors, *FC 2015 Workshops*, volume 8976 of *LNCS*, pages 194–212. Springer, Heidelberg, January 2015.

[45] Seung Geol Choi, Dana Dachman-Soled, Mukul Kulkarni, and Arkady Yerukhimovich. Differentially-private multi-party sketching for large-scale statistics. *PoPETs*, 2020(3):153–174, July 2020.

[46] Seung Geol Choi, Dana Dachman-Soled, Mukul Kulkarni, and Arkady Yerukhimovich. Differentially-private multi-party sketching for large-scale statistics. *Proceedings on Privacy Enhancing Technologies*, 2020(3):153–174, 2020.

[47] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.

[48] Chris Clifton and Balamurugan Anandan. Challenges and opportunities for security with differential privacy. In *International Conference on Information Systems Security*, pages 1–13. Springer, 2013.

[49] Graham Cormode and Hossein Jowhari. L p samplers and their applications: A survey. *ACM Computing Surveys (CSUR)*, 52(1):1–31, 2019.

[50] Graham Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.

[51] Henry Corrigan-Gibbs, David Isaac Wolinsky, and Bryan Ford. Proactively accountable anonymous messaging in verdict. In *Presented as part of the 22nd {USENIX} Security Symposium ({USENIX} Security 13)*, pages 147–162, 2013.

[52] Emiliano De Cristofaro, Sky Faber, Paolo Gasti, and Gene Tsudik. Genodroid: are privacy-preserving genomic tests ready for prime time? In *WPES 2012*, pages 97–108. ACM, 2012.

[53] Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. Fast and private computation of cardinality of set intersection and union. In *CANS 2012*, volume 7712, pages 218–231. Springer, 2012.

[54] Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 79–88. ACM Press, October / November 2006.

[55] Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. LWE with side information: Attacks and concrete security estimation. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 329–358. Springer, Heidelberg, August 2020.

[56] Charlie Dickens, Justin Thaler, and Daniel Ting. Order-invariant cardinality estimators are differentially private. *Advances in Neural Information Processing Systems*, 35:15204–15216, 2022.

[57] Yevgeniy Dodis, Paul Grubbs, Thomas Ristenpart, and Joanne Woodage. Fast message franking: From invisible salamanders to encryptment. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 155–186. Springer, Heidelberg, August 2018.

[58] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, jan 2008.

[59] Benjamin Doerr. Probabilistic tools for the analysis of randomized optimization heuristics. *CoRR*, abs/1801.06733, 2018.

[60] Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II 33*, pages 1–12. Springer, 2006.

[61] Cynthia Dwork. Differential privacy (invited paper). In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP 2006, Part II*, volume 4052 of *LNCS*, pages 1–12. Springer, Heidelberg, July 2006.

[62] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 486–503. Springer, 2006.

[63] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography (TCC 2006)*, pages 265–284. Springer, 2006.

[64] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 265–284. Springer, Heidelberg, March 2006.

[65] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

[66] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[67] Stefan Dziembowski, Tomasz Kazana, and Maciej Zdanowicz. Quasi chain rule for min-entropy. *Inf. Process. Lett.*, 134:62–66, 2018.

[68] Rolf Egert, Marc Fischlin, David Gens, Sven Jacob, Matthias Senker, and Jörn Tillmanns. Privately computing set-union and set-intersection cardinality via bloom filters. In *Information Security and Privacy - 20th Australasian Conference, ACISP 2015, Brisbane, QLD, Australia, June 29 - July 1, 2015, Proceedings*, pages 413–430, 2015.

[69] Tariq Elahi, George Danezis, and Ian Goldberg. PrivEx: Private collection of traffic statistics for anonymous communication networks. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 1068–1079. ACM Press, November 2014.

[70] Reo Eriguchi, Atsunori Ichikawa, Noboru Kunihiro, and Koji Nuida. Efficient noise generation to achieve differential privacy with applications to secure multiparty computation. In *International Conference on Financial Cryptography and Data Security*, pages 271–290. Springer, 2021.

[71] David Evans, Vladimir Kolesnikov, and Mike Rosulek. A pragmatic introduction to secure multi-party computation. *Foundations and Trends in Privacy and Security*, 2(2-3):70–246, 2018.

[72] Sky Faber. *Variants of Privacy Preserving Set Intersection and their Practical Applications*. PhD thesis, University of Maryland, 2016.

[73] Facebook. How is Facebook addressing false information through independent fact-checkers? `https://www.facebook.com/help/1952307158131536`.

[74] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, 2012:144, 2012.

[75] Li Fan, Pei Cao, Jussara M. Almeida, and Andrei Z. Broder. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM Trans. Netw.*, 8(3):281–293, 2000.

[76] Joan Feigenbaum, Yuval Ishai, Tal Malkin, Kobbi Nissim, Martin Strauss, and Rebecca N. Wright. Secure multiparty computation of approximations. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *ICALP 2001*, volume 2076 of *LNCS*, pages 927–938. Springer, Heidelberg, July 2001.

[77] Joan Feigenbaum, Yuval Ishai, Tal Malkin, Kobbi Nissim, Martin J. Strauss, and Rebecca N. Wright. Secure multiparty computation of approximations. *ACM Trans. Algorithms*, 2(3):435–472, 2006.

[78] Joan Feigenbaum, Yuval Ishai, Tal Malkin, Kobbi Nissim, Martin J. Strauss, and Rebecca N. Wright. Secure multiparty computation of approximations. *ACM Trans. Algorithms*, 2(3):435–472, 2006.

[79] Ellis Fenske, Akshaya Mani, Aaron Johnson, and Micah Sherr. Distributed measurement with private set-union cardinality. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2295–2312. ACM Press, October / November 2017.

[80] Nils Fleischhacker, Kasper Green Larsen, and Mark Simkin. Property-preserving hash functions for hamming distance from standard assumptions. In *EUROCRYPT 2022, Part II*, LNCS, pages 764–781. Springer, Cham, Switzerland, June 2022.

[81] Nils Fleischhacker and Mark Simkin. Robust property-preserving hash functions for hamming distance and more. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part III*, volume 12698 of *LNCS*, pages 311–337. Springer, Cham, Switzerland, October 2021.

[82] Benjamin Fuller, Mayank Varia, Arkady Yerukhimovich, Emily Shen, Ariel Hamlin, Vijay Gadepally, Richard Shay, John Darby Mitchell, and Robert K. Cunningham. SoK: Cryptographically protected database search. In *2017 IEEE Symposium on Security and Privacy*, pages 172–191. IEEE Computer Society Press, May 2017.

[83] Sumit Ganguly. Counting distinct items over update streams. *Theoretical Computer Science*, 378(3):211–222, 2007.

[84] R.Stuart Geiger. Bot-based collective blocklists in twitter: the counterpublic moderation of harassment in a networked public space. *Information, Communication & Society*, 19:787–803, 06 2016.

[85] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

[86] Eu-Jin Goh and Philippe Golle. Event driven private counters. In Andrew Patrick and Moti Yung, editors, *FC 2005*, volume 3570 of *LNCS*, pages 313–327. Springer, Heidelberg, February / March 2005.

[87] Oded Goldreich. Towards a theory of software protection and simulation by oblivious RAMs. In Alfred Aho, editor, *19th ACM STOC*, pages 182–194. ACM Press, May 1987.

[88] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

[89] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious RAMs. *J. ACM*, 43(3):431–473, 1996.

[90] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996.

[91] Michael T. Goodrich. Data-oblivious external-memory algorithms for the compaction, selection, and sorting of outsourced data. In *SPAA 2011: Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures, San Jose, CA, USA, June 4-6, 2011 (Co-located with FCRC 2011)*, pages 379–388, 2011.

[92] S. Dov Gordon, Jonathan Katz, Mingyu Liang, and Jiayu Xu. Spreading the privacy blanket: - differentially oblivious shuffling for differential privacy. In Giuseppe Ateniese and Daniele Venturi, editors, *ACNS 22: 20th International Conference on Applied Cryptography and Network Security*, volume 13269 of *LNCS*, pages 501–520, Rome, Italy, June 20–23, 2022. Springer, Cham, Switzerland.

[93] Slawomir Goryczka, Li Xiong, and Vaidy Sunderam. Secure multiparty aggregation with differential privacy: A comparative study. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, pages 155–163, 2013.

[94] Adam Groce, Peter Rindal, and Mike Rosulek. Cheaper private set intersection via differentially private leakage. *Proc. Privacy Enhancing Technologies (PETS)*, 2019(3):6–25, 2019.

[95] Paul Grubbs, Jiahui Lu, and Thomas Ristenpart. Message franking via committing authenticated encryption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 66–97. Springer, Heidelberg, August 2017.

[96] Paul Grubbs, Richard McPherson, Muhammad Naveed, Thomas Ristenpart, and Vitaly Shmatikov. Breaking web applications built on top of encrypted data. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1353–1364. ACM Press, October 2016.

[97] Paul Grubbs, Kevin Sekniqi, Vincent Bindschaedler, Muhammad Naveed, and Thomas Ristenpart. Leakage-abuse attacks against order-revealing encryption. In *2017 IEEE Symposium on Security and Privacy*, pages 655–672. IEEE Computer Society Press, May 2017.

[98] Zichen Gui, Oliver Johnson, and Bogdan Warinschi. Encrypted databases: New volume attacks against range queries. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 361–378. ACM Press, November 2019.

[99] Shai Halevi, Robert Krauthgamer, Eyal Kushilevitz, and Kobbi Nissim. Private approximation of NP-hard functions. In *33rd ACM STOC*, pages 550–559. ACM Press, July 2001.

[100] Xi He, Ashwin Machanavajjhala, Cheryl J. Flynn, and Divesh Srivastava. Composing differential privacy and secure computation: A case study on scaling private record linkage. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1389–1406. ACM Press, October / November 2017.

[101] Jonathan Hehir, Daniel Ting, and Graham Cormode. Sketch-flip-merge: Mergeable sketches for private distinct counting. In *ICML 2023*, volume 202, pages 12846–12865. PMLR, 2023.

[102] Thang Hoang, Rouzbeh Behnia, Yeongjin Jang, and Attila A. Yavuz. MOSE: Practical multi-user oblivious storage via secure enclaves. In *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*, CODASPY '20, page 17–28, New York, NY, USA, 2020. Association for Computing Machinery.

[103] Justin Holmgren, Minghao Liu, LaKyah Tyner, and Daniel Wichs. Nearly optimal property preserving hashing. In *CRYPTO 2022, Part III*, LNCS, pages 473–502. Springer, Cham, Switzerland, August 2022.

[104] Ziyue Huang, Yuan Qiu, Ke Yi, and Graham Cormode. Frequency estimation under multiparty differential privacy: One-shot and streaming. *arXiv preprint arXiv:2104.01808*, 2021.

[105] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *30th ACM STOC*, pages 604–613. ACM Press, May 1998.

[106] Mike Isaac and Kevin Roose. Disinformation spreads on whatsapp ahead of brazilian election. `https://www.nytimes.com/2018/10/19/technology/whatsapp-brazil-presidential-election.html`, 2018.

[107] Yuval Ishai, Eyal Kushilevitz, Steve Lu, and Rafail Ostrovsky. Private large-scale databases with distributed searchable symmetric encryption. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 90–107. Springer, Heidelberg, February / March 2016.

[108] Yuval Ishai, Tal Malkin, Martin J. Strauss, and Rebecca N. Wright. Private multiparty sampling and approximation of vector combinations. *Theor. Comput. Sci.*, 410(18):1730–1745, 2009.

[109] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *NDSS 2012*. The Internet Society, February 2012.

[110] Rawane Issa, Nicolas AlHaddad, and Mayank Varia. Hecate: Abuse reporting in secure messengers with sealed sender. Cryptology ePrint Archive, 2021.

[111] Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579, 1901.

[112] Rob Jansen and Aaron Johnson. Safely measuring tor. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1553–1567. ACM Press, October 2016.

[113] Bo Jiang, Hamid Krim, Tianfu Wu, and Derya Cansever. Refining self-supervised learning in imaging: Beyond linear metric. In *ICIP 2022*, pages 76–80. IEEE, 2022.

[114] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space, 1984.

[115] Hossein Jowhari, Mert Saglam, and Gabor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 49–58, 2011.

[116] Hossein Jowhari, Mert Saglam, and Gabor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In Maurizio Lenzerini and Thomas Schwentick, editors, *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, pages 49–58. ACM, 2011.

[117] Ata Kabán. Improved bounds on the dot product under random projection and random sign projection. In Longbing Cao, Chengqi Zhang, Thorsten Joachims, Geoffrey I. Webb, Dragos D. Margineantu, and Graham Williams, editors, *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 487–496. ACM, 2015.

[118] Bala Kalyanasundaram and Georg Schnitger. The probabilistic communication complexity of set intersection. *SIAM J. Discret. Math.*, 5(4):545–557, 1992.

[119] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography.* Chapman and Hall/CRC Press, 2007.

[120] Jonathan Katz, Steven Myers, and Rafail Ostrovsky. Cryptographic counters and applications to electronic voting. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 78–92. Springer, Heidelberg, May 2001.

[121] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. Generic attacks on secure outsourced databases. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1329–1340. ACM Press, October 2016.

[122] Marcel Keller and Peter Scholl. Efficient, oblivious data structures for MPC. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 506–525. Springer, Heidelberg, December 2014.

[123] Myungsun Kim, Hyung Tae Lee, San Ling, Benjamin Hong Meng Tan, and Huaxiong Wang. Private compound wildcard queries using fully homomorphic encryption. *IEEE Trans. Dependable Secur. Comput.*, 16(5):743–756, 2019.

[124] Benjamin Kreuter, Craig William Wright, Evgeny Sergeevich Skvortsov, Raimundo Mirisola, and Yao Wang. Privacy-preserving secure cardinality and frequency estimation. Technical report, Google, LLC, 2020.

[125] Anunay Kulshrestha and Jonathan Mayer. Identifying harmful media in end-to-end encrypted communication: Efficient private membership computation. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 893–910. USENIX Association, August 2021.

[126] Ping Li, Art B. Owen, and Cun-Hui Zhang. One permutation hashing. In *In Proceedings of the 26th Annual Conference on Neural Information Processing Systems 2012.*, pages 3122–3130, 2012.

[127] Tian Li, Zaoxing Liu, Vyas Sekar, and Virginia Smith. Privacy for free: Communication-efficient learning with differential privacy using sketches. *arXiv preprint arXiv:1911.00972*, 2019.

[128] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009.

[129] Li-Ping Liu. Linear transformation of multivariate normal distribution: Marginal, joint and posterior. `http://www.cs.columbia.edu/~liulp/pdf/linear_normal_dist.pdf`. Accessed: 2020-02-16.

[130] Joshua Lund. Technology preview: Sealed sender for signal, Oct 2018.

[131] Matteo Maffei, Giulio Malavolta, Manuel Reinert, and Dominique Schröder. Maliciously secure multi-client oram. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *Applied Cryptography and Network Security*, pages 645–664, Cham, 2017. Springer International Publishing.

[132] Ian Martiny, Gabriel Kaptchuk, Adam Aviv, Dan Roche, and Eric Wustrow. Improving signal's sealed sender. In *ISOC Network and Distributed System Security Symposium – NDSS*, 01 2021.

[133] Sahar Mazloom and S. Dov Gordon. Secure computation with differentially private access patterns. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 490–507. ACM Press, October 2018.

[134] Sahar Mazloom, Phi Hung Le, Samuel Ranellucci, and S. Dov Gordon. Secure parallel computation on national scale volumes of data. In *USENIX Security Symposium*, pages 2487–2504, Boston, MA, USA, August 2020. USENIX Association.

[135] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th FOCS*, pages 94–103. IEEE Computer Society Press, October 2007.

[136] Luca Melis, George Danezis, and Emiliano De Cristofaro. Efficient private statistics with succinct sketches. In *NDSS 2016*, San Diego, CA, February 2016. Internet Society.

[137] Luca Melis, George Danezis, and Emiliano De Cristofaro. Efficient private statistics with succinct sketches. In *Proceedings 2016 Network and Distributed System Security Symposium*, NDSS 2016. Internet Society, 2016.

[138] Darakhshan Mir, Shan Muthukrishnan, Aleksandar Nikolov, and Rebecca N Wright. Pan-private algorithms via statistics on sketches. In *ACM SIGMOD-SIGACT-SIGART*, pages 37–48, 2011.

[139] Michael Mitzenmacher. Compressed bloom filters. In Ajay D. Kshemkalyani and Nir Shavit, editors, *20th ACM PODC*, pages 144–150. ACM, August 2001.

[140] Morteza Monemizadeh and David P. Woodruff. 1-pass relative-error l_p-sampling with applications. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1143–1160. SIAM, 2010.

[141] Christian Mouchet, Juan Ramón Troncoso-Pastoriza, Jean-Philippe Bossuat, and Jean-Pierre Hubaux. Multiparty homomorphic encryption from ring-learning-with-errors. *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2021(4):291–311, October 2021.

[142] Saskia Nuñez von Voigt and Florian Tschorsch. Rrtxfm: Probabilistic counting for differentially private statistics. In *Digital Transformation for a Sustainable Society in the 21st Century: I3E 2019 IFIP WG 6.11 International Workshops*, pages 86–98. Springer, 2020.

[143] Rafail Ostrovsky. Efficient computation on oblivious RAMs. In *22nd ACM STOC*, pages 514–523. ACM Press, May 1990.

[144] Rasmus Pagh and Nina Mesing Stausholm. Efficient Differentially Private F_0 Linear Sketching. In Ke Yi and Zhewei Wei, editors, *24th International Conference on Database Theory (ICDT 2021)*, volume 186 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:19, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[145] Rasmus Pagh and Mikkel Thorup. Improved utility analysis of private countsketch. *Advances in Neural Information Processing Systems*, 35:25631–25643, 2022.

[146] Omkant Pandey and Yannis Rouselakis. Property preserving symmetric encryption. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 375–391. Springer, Heidelberg, April 2012.

[147] Vasilis Pappas, Fernando Krell, Binh Vo, Vladimir Kolesnikov, Tal Malkin, Seung Geol Choi, Wesley George, Angelos D. Keromytis, and Steve Bellovin. Blind seer: A scalable private DBMS. In *2014 IEEE Symposium on Security and Privacy*, pages 359–374. IEEE Computer Society Press, May 2014.

[148] Sarvar Patel, Giuseppe Persiano, Kevin Yeo, and Moti Yung. Mitigating leakage in secure cloud-hosted data structures: Volume-hiding for multimaps via hashing. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 79–93. ACM Press, November 2019.

[149] Manas Pathak, Shantanu Rane, and Bhiksha Raj. Multiparty differential privacy via aggregation of locally trained classifiers. *Advances in neural information processing systems*, 23, 2010.

[150] Alexander Payne. bitvec. `https://lib.rs/crates/bitvec`, 11 2021.

[151] Charlotte Peale, Saba Eskandarian, and Dan Boneh. Secure source-tracking for encrypted messaging. In *ACM CCS 21: 28th Conference on Computer and Communications Security*, Virtual Conference, Korea, November 15–19, 2021. ACM Press.

[152] Sikha Pentyala, Davis Railsback, Ricardo Maia, Rafael Dowsley, David Melanson, Anderson Nascimento, and Martine De Cock. Training differentially private models with secure multiparty computation. *arXiv preprint arXiv:2202.02625*, 2022.

[153] Manoj M Prabhakaran and Vinod M Prabhakaran. On secure multiparty sampling for more than two parties. In *2012 IEEE Information Theory Workshop*, pages 99–103. IEEE, 2012.

[154] Vinod M Prabhakaran and Manoj M Prabhakaran. Assisted common information with an application to secure two-party sampling. *IEEE Transactions on Information Theory*, 60(6):3413–3434, 2014.

[155] Alexander A. Razborov. On the distributional complexity of disjointness. *Theor. Comput. Sci.*, 106(2):385–390, 1992.

[156] M. Sadegh Riazi, Beidi Chen, Anshumali Shrivastava, Dan Wallach, and Farinaz Koushanfar. Sub-linear privacy-preserving near-neighbor search. Cryptology ePrint Archive, Report 2019/1222, 2019. `https://eprint.iacr.org/2019/1222`.

[157] Daniel S. Roche, Daniel Apon, Seung Geol Choi, and Arkady Yerukhimovich. POPE: Partial order preserving encoding. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1131–1142. ACM Press, October 2016.

[158] Elyse Samuels. How misinformation on whatsapp led to a mob killing in india. `https://www-washingtonpost-com.proxygw.wrlc.org/politics/2020/02/21/how-misinformation-whatsapp-led-deathly-mob-lynching-india/ics/2020/02/21/how-misinformation-whatsapp-led-deathly-mob-lynching-india/`, 2020.

[159] Igal Sason and Sergio Verdú. Bounds among f-divergences. *CoRR*, abs/1508.00335, 2015.

[160] Microsoft SEAL (release 3.6). `https://github.com/Microsoft/SEAL`, November 2020. Microsoft Research, Redmond, WA.

[161] Elaine Shi. Path oblivious heap: Optimal and practical oblivious priority queue. In *2020 IEEE Symposium on Security and Privacy*, pages 842–858. IEEE Computer Society Press, May 2020.

[162] Elaine Shi, T.-H. Hubert Chan, Eleanor G. Rieffel, and Dawn Song. Distributed private data analysis: Lower bounds and practical constructions. *ACM Trans. Algorithms*, 13(4):50:1–50:38, 2017.

[163] Elaine Shi, T.-H. Hubert Chan, Eleanor Gilbert Rieffel, and Dawn Song. Distributed private data analysis: Lower bounds and practical constructions. *ACM Trans. Algorithms*, 13(4):50:1–50:38, 2017.

[164] Maciej Skórski. Strong chain rules for min-entropy under few bits spoiled. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 1122–1126. IEEE, 2019.

[165] Adam Smith, Shuang Song, and Abhradeep Guha Thakurta. The flajolet-martin sketch itself preserves differential privacy: Private counting with minimal space. *NeurIPS 2020*, 33:19561–19572, 2020.

[166] Brian Smith. ring. `https://lib.rs/crates/ring`, 11 2021.

[167] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *2000 IEEE Symposium on Security and Privacy*, pages 44–55. IEEE Computer Society Press, May 2000.

[168] Hagen Sparka, Florian Tschorsch, and Björn Scheuermann. P2kmv: A privacy-preserving counting sketch for efficient and accurate set intersection cardinality estimations. *Cryptology ePrint Archive*, 2018.

[169] Rade Stanojevic, Mohamed Nabeel, and Ting Yu. Distributed cardinality estimation of set operations with differential privacy. In *2017 IEEE Symposium on Privacy-Aware Computing (PAC)*, pages 37–48. IEEE, 2017.

[170] Emil Stefanov, Marten van Dijk, Elaine Shi, T.-H. Hubert Chan, Christopher W. Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM: an extremely simple oblivious RAM protocol. *J. ACM*, 65(4):18:1–18:26, 2018.

[171] W. A. Stein et al. *Sage Mathematics Software (Version 9.2)*. The Sage Development Team, 2020. `http://www.sagemath.org`.

[172] Yang Tan and Bo Lv. Break two psi-ca protocols in polynomial time. In *Proceedings of the 2024 16th International Conference on Machine Learning and Computing*, ICMLC '24, page 65–72, New York, NY, USA, 2024. Association for Computing Machinery.

[173] Chayant Tantipathananandh, Tanya Y. Berger-Wolf, and David Kempe. A framework for community identification in dynamic social networks. In *KDD*, pages 717–726. ACM, 2007.

[174] Alexander J. Titus, Shashwat Kishore, Todd Stavish, Stephanie M. Rogers, and Karl Ni. Pyseal: A python wrapper implementation of the seal homomorphic encryption library, 2018.

[175] Nirvan Tyagi, Yossi Gilad, Derek Leung, Matei Zaharia, and Nickolai Zeldovich. Stadium: A distributed metadata-private messaging system. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 423–440. ACM, 2017.

[176] Nirvan Tyagi, Paul Grubbs, Julia Len, Ian Miers, and Thomas Ristenpart. Asymmetric message franking: Content moderation for metadata-private end-to-end encryption. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 222–250. Springer, Heidelberg, August 2019.

[177] Nirvan Tyagi, Ian Miers, and Thomas Ristenpart. Traceback for end-to-end encrypted messaging. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 413–430. ACM Press, November 2019.

[178] Ryan Wails, Aaron Johnson, Daniel Starin, Arkady Yerukhimovich, and S. Dov Gordon. Stormy: Statistics in tor by measuring securely. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 615–632. ACM Press, November 2019.

[179] Lun Wang, Iosif Pinelis, and Dawn Song. Differentially private fractional frequency moments estimation with polylogarithmic space. In *International Conference on Learning Representations*, 2022.

[180] Xiao Shaun Wang, Kartik Nayak, Chang Liu, T.-H. Hubert Chan, Elaine Shi, Emil Stefanov, and Yan Huang. Oblivious data structures. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 215–226. ACM Press, November 2014.

[181] Rui Wen, Yu Yu, Xiang Xie, and Yang Zhang. LEAF: A faster secure search algorithm via localization, extraction, and reconstruction. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1219–1232. ACM Press, November 2020.

[182] David P Woodruff and Peilin Zhong. Distributed low rank approximation of implicit functions of a matrix. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 847–858. IEEE, 2016.

[183] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples for graph data: Deep insights into attack and defense. In *IJCAI 2019*, pages 4816–4823. ijcai.org, 2019.

[184] Ziqi Yan, Jiqiang Liu, Gang Li, Zhen Han, and Shuo Qiu. Privmin: Differentially private minhash for jaccard similarity computation. *CoRR*, abs/1705.07258, 2017.

[185] Ziqi Yan, Qiong Wu, Meng Ren, Jiqiang Liu, Shaowu Liu, and Shuo Qiu. Locally private jaccard similarity estimation. *Concurr. Comput. Pract. Exp.*, 31(24), 2019.

[186] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

[187] Masaya Yasuda, Takeshi Shimoyama, Jun Kogure, Kazuhiro Yokoyama, and Takeshi Koshiba. Secure pattern matching using somewhat homomorphic encryption. In Ari Juels and Bryan Parno, editors, *CCSW'13, Proceedings of the 2013 ACM Cloud Computing Security Workshop, Co-located with CCS 2013, Berlin, Germany, November 4, 2013*, pages 65–76. ACM, 2013.

[188] Youtube. How does YouTube combat misinformation? `https://www.youtube.com/howyoutubeworks/our-commitments/fighting-misinformation/#policies`.

[189] Fuheng Zhao, Dan Qiao, Rachel Redberg, Divyakant Agrawal, Amr El Abbadi, and Yu-Xiang Wang. Differentially private linear sketches: Efficient implementations and applications. *NeurIPS 2022*, 35:12691–12704, 2022.

[190] Ferdinand Österreicher. Csiszar's f-divergence-basic properties. *RGMIA Research Report Collection*, 2002.