Secure Sampling with Sublinear Communication

Seung Geol Choi[1], Dana Dachman-Soled[2],
S. Dov Gordon[3], Linsheng Liu[4], and Arkady Yerukhimovich[4]

[1] United States Naval Academy
choi@usna.edu
[2] University of Maryland
danadach@ece.umd.edu
[3] George Mason University
gordon@gmu.edu
[4] George Washington University
lls@gwu.edu, arkady@gwu.edu

Abstract. Random sampling from specified distributions is an important tool with wide applications for analysis of large-scale data. In this paper we study how to randomly sample when the distribution is partitioned among two parties' private inputs. Of course, a trivial solution is to have one party send a (possibly encrypted) description of its weights to the other party who can then sample over the entire distribution (possibly using homomorphic encryption). However, this approach requires communication that is linear in the input size which is prohibitively expensive in many settings. In this paper, we investigate secure 2-party sampling with *sublinear communication* for many standard distributions. We develop protocols for L_1, and L_2 sampling. Additionally, we investigate the feasibility of sublinear product sampling, showing impossibility for the general problem and showing a protocol for a restricted case of the problem. We additionally show how such product sampling can be used to instantiate a sublinear communication 2-party exponential mechanism for differentially-private data release.

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 [2,37,28,6].

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 $\frac{w_{1,i} \cdot w_{2,i}}{\langle \mathbf{w}_1, \mathbf{w}_2 \rangle} = \frac{w_{1,i} \cdot w_{2,i}}{\sum_j (w_{1,j} \cdot w_{2,j})}$[1].

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 [35].

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 2.

L_2 sampling. We also provide a protocol for secure L_2 sampling that relies on fully homomorphic encryption (see Section 3). In this case, however, achieving

[1] 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 $\perp$.

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 $\tilde{O}(1)$ communication[2]. 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 for justification).

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 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 [3].

Two party exponential mechanism. As mentioned previously, one important application of product sampling is the exponential mechanism for providing differential privacy [35]. In Section 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.

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

[2] 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.

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 2.

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

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 4.

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 [3]. 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 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. $\mathcal{D}_2$).[3] 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 4.1 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.

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

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 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 4.2 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 [3]!

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. [19,20] 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 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 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 $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle \in \omega(\log n/n)$.

1.3 Related Work

Sampling from streaming data. Many prior papers (e.g. [10,22,29,37,46]) 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 [41,42] 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. [7] 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. [13] there has been a good amount of work (e.g. [1,9,17,23,39,40]) 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 [16,26,36,45,8] 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.

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 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.

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.

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.

Theorem 1. *Protocol 1 securely realizes $\mathcal{F}_{L_1}$ with semi-honest security in the $(\mathcal{F}_{\mathsf{osample(L_1)}}, \mathcal{F}_{\mathsf{biasCoin}})$-hybrid.*

Protocol 1 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: $i_1 \cdot (1 - b) + i_2 \cdot b$.

The proof is found in Appendix C.1.

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 2.

Protocol 2 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.

Theorem 2. *Assuming the existence of threshold FHE with* IND-CPA *security, Protocol 2 securely realizes $\mathcal{F}_{\mathsf{osample(L_1)}}$ in the semi-honest security model.*

The proof is found in Appendix C.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 Appendix D.

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

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 3.1, the construction is found in Protocol 3. We then show how to implement the protocol securely in Section 3.2.

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 1. *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 1. *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:

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

□

Definition 2. *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 3. *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 2. *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 3. We show the correctness and efficiency of the protocol.

Lemma 3. *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 2. Finally, each round has $\tilde{O}(1)$ communication, since Π_{ignore} has communication $\tilde{O}(1)$ (by Lemma 1) 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 1. 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.

Protocol 3 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
$$\frac{f_c(\mathbf{w}_1, \mathbf{w}_2, i)}{2 \cdot \Pr_{D_{\mathsf{ignore}}(\mathbf{w}_1,\mathbf{w}_2)}[i]} = \frac{c \cdot \Pr_{D_{L_2}(\mathbf{w}_1,\mathbf{w}_2)}[i]}{2 \cdot \Pr_{D_{\mathsf{ignore}}(\mathbf{w}_1,\mathbf{w}_2)}[i]} = \frac{\Pr_{D_{L_2}(\mathbf{w}_1,\mathbf{w}_2)}[i]}{2/c \cdot \Pr_{D_{\mathsf{ignore}}(\mathbf{w}_1,\mathbf{w}_2)}[i]}$$
and otherwise return to step 1.

3.2 Secure L_2 Sampling From FHE

L_2 sampling protocol. We present our secure L_2 sampling protocol in Protocol 4. 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 Appendix E.

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 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 3. *Assuming the existence of threshold FHE with* IND-CPA *security, Protocol 4 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 Appendix C.3.

Protocol 4 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 $\mathcal{F}_{L_1}^{ss}$ 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
   $$\frac{f_c(\mathbf{w}_1, \mathbf{w}_2)_{i_j}}{2 \cdot \Pr_{D_{\mathsf{ignore}}(\mathbf{w}_1, \mathbf{w}_2)}[i_j]} = \frac{w_{1,i_j}^2 + 2w_{1,i_j}w_{2,i_j} + w_{2,i_j}^2}{2(w_{1,i_j}^2 + w_{2,i_j}^2)}.$$
   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: $(\widehat{\mathsf{out}}_1, \ldots, \widehat{\mathsf{out}}_B)$ 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.

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 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:

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

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

Definition 4. *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 $\frac{w_{1,i}^p + w_{2,i}^p}{||\mathbf{w}_1||_p^p + ||\mathbf{w}_2|_p^p}$.*

Protocol 5 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.

Lemma 4. *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 5. *For $i \in [n]$, let the corrective parameter function be defined as*
$$f_c(\mathbf{w}_1, \mathbf{w}_2, i) := \frac{(w_{1,i} + w_{2,i})^p}{||\mathbf{w}_1||_p^p + ||\mathbf{w}_2||_p^p}.$$

Definition 6. *The constant $c := c(\mathbf{w}_1, \mathbf{w}_2)$ is defined as*
$$c(\mathbf{w}_1, \mathbf{w}_2) := \frac{||\mathbf{w}_1 + \mathbf{w}_2||_p^p}{||\mathbf{w}_1||_p^p + ||\mathbf{w}_2||_p^p}$$
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 5. *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 Appendix C.4.

We now present the L_p sampling protocol Π_{L_p} in Protocol 5. We show the correctness and efficiency of the protocol below.

Lemma 6. *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 Appendix C.5. We note that this result strictly generalizes Lemma 3. In particular, setting $p = 2$ in the above protocol yields a protocol with exactly the same parameters as the L_2 sampling protocol.

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.

4.1 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 ([43,31,4]). *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 5. *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 5 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 7. *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 8. *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 (Proof of Lemma 7). 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 (Proof of Lemma 8). 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

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

□

4.2 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 4.1 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 $\mathbf{w}_1, \mathbf{w}_2$[4]. 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,[5] 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 6). This protocol is in the $\{\mathcal{F}_{\mathsf{osample(L_1)}}, \mathcal{F}_{2PC}\}$-hybrid model.

Security. We will prove the following theorem.

[4] 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.

[5] This can be shown by a simple modification of the lower bound proof from Section 4.1.

Protocol 6 Product sampling (Π^{IP}_{prod}) in the $\{\mathcal{F}_{\mathsf{osample}(\mathsf{L}_1)}, \mathcal{F}_{2PC}\}$-hybrid.

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

1. Invoke the $\mathcal{F}_{\mathsf{osample}(\mathsf{L}_1)}$ ideal functionality with P_1 as the sender with input $\mathbf{w}_1$ and P_2 as the receiver. Let $i_{1,1}$ and $i_{1,2}$ be the output from the ideal functionality to P_1 and P_2 respectively.
2. Invoke the $\mathcal{F}_{\mathsf{osample}(\mathsf{L}_1)}$ ideal functionality with P_2 as the sender with input $\mathbf{w}_2$ and P_1 as the receiver. Let $i_{2,1}$ 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 6. *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 $\{\mathcal{F}_{\mathsf{osample}(\mathsf{L}_1)}, \mathcal{F}_{2PC}\}$-hybrid model with semi-honest security.*

Proof. We describe the simulator Sim in the $\{\mathcal{F}_{\mathsf{osample}(\mathsf{L}_1)}, \mathcal{F}_{2PC}\}$-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 $\mathcal{F}_{\mathsf{osample}(\mathsf{L}_1)}$ and Sim returns to it a random value in $\mathbb{Z}_n$. Party 1 sends its input to the second invocation of $\mathcal{F}_{\mathsf{osample}(\mathsf{L}_1)}$ 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 $\mathcal{F}_{\mathsf{osample}(\mathsf{L}_1)}$ and Sim returns to it a random value in $\mathbb{Z}_n$. Party 1 sends its input to the second invocation of $\mathcal{F}_{\mathsf{osample}(\mathsf{L}_1)}$ 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 5. 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 $\mathit{\Pi}_{\mathsf{prod}}$ 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.

5 Product Sampling in Constant Rounds

Achieving constant rounds through parallel repetition. In Sections 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 $\mathit{\Pi}_{\mathsf{prod}}$ 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 5.1). Using this approximation, we can then realize our sublinear communication, constant round protocol for product sampling as follows in the next subsection.

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) [27,24] 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:

```
approxIP($\mathbf{w}_1, \mathbf{w}_2$):      ▷ $\mathbf{w}_1$ 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 $\frac{1}{k} \cdot \langle \mathbf{Mw}_1, \mathbf{Mw}_2 \rangle$. (Here, we slightly abuse the notation and treat the vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ as column vectors.)
```

Lemma 9. (cf. [30, Corollary 3.1]) *For all $\mathbf{w}_1, \mathbf{w}_2$ such that $\cos(\mathbf{w}_1, \mathbf{w}_2) \geq t$, the procedure* 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 [21]. In particular, we prove the following:

Lemma 10. *The output of* 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 Appendix C.6.

Private protocol via JLT. Using the JLT sketch, we can design a private protocol approximating the inner product. See Protocol 7. The protocol uses threshold FHE (e.g., [38]).

Protocol 7 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 10, the leakage profile of the functionality is $\langle \mathbf{w}_1, \mathbf{w}_1 \rangle$, $\langle \mathbf{w}_2, \mathbf{w}_2 \rangle$, and $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle$.

5.2 Constant-round protocol for product sampling

Note that the Protocol 6 has the following structure. In particular:

- The probability that Protocol 6 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

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

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

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

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

In summary, by running $\frac{(1+\epsilon)\cdot\omega(\log\lambda)}{\tilde{p}}$ 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.

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 6.1.

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 λ 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 11. *If* $|D| \geq 40(\log n + \lambda)/\epsilon$, *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

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

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 $|D| \geq 40(\log n + \lambda)/\epsilon$, 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 [15] to our setting, we have

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

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

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

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.

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 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 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 9.
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 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 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 7. *If $|D| \geq 40 \cdot (\log n + \lambda)/\epsilon$, 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 $\langle \mathbf{M}\mathbf{w}_1, \mathbf{M}\mathbf{w}_2 \rangle$ 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 $|D| > \Delta u \cdot \lambda/\epsilon$, this probability is at most $e^{-\lambda}$, 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.

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

Protocol 8 Exponential Mechanism Protocol ($\mathit{\Pi}_{\mathsf{EM}}$) in the $\{\mathcal{F}_{2PC}, \mathcal{F}_{\mathsf{osample(L_1)}}\}$-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 $v_{b,i} = e^{-\epsilon \cdot \frac{f^{c_i}_{\mathsf{loss}}(D^b)}{2\Delta u}}$, where $\Delta u = \frac{\epsilon}{40(\log n + \lambda)}$. 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 $m = \frac{(1+2\epsilon)\cdot\omega(\log \lambda)}{\eta}$ times in parallel.
   (a) 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 $i^j_{1,1}$ and $i^j_{1,2}$ be the output of the jth execution to P_1 and P_2 respectively.
   (b) 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 $i^j_{2,1}$ and $i^j_{2,2}$ 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: $(i^j_{1,1}, i^j_{2,1}, i^j_{1,2}, i^j_{2,2})$ for $j = 1, \ldots, m$.
   (a) Let $i^j_1 = i^j_{1,1} \oplus i^j_{1,2}$, $i^j_2 = i^j_{2,1} \oplus i^j_{2,2}$.
   (b) Find the smallest j such that i^j_1 equals i^j_2, and output i^j_1 to both P_1 and P_2. If no such j exists, output abort.

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

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.

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

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

The first inequality holds form the DP of protocol $\mathsf{DP\text{-}approxIP}$, and the second inequality holds from the DP of the exponential mechanism.

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

Acknowledgments

Seung Geol Choi is supported in part by NSF grant CNS-1955319. Dana Dachman-Soled is supported in part by NSF grants #IIS-2147276, #CNS-1933033, #CNS-1453045 (CAREER), and by financial assistance awards

70NANB15H328 and 70NANB19H126 from the U.S. Department of Commerce, National Institute of Standards and Technology. Dov Gordon is supported by NSF grant #CNS-1955264. Arkady Yerukhimovich is supported in part by NSF grant #CNS-1955620.

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. 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.
3. 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.
4. 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.
5. 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.
6. Vladimir Braverman, Rafail Ostrovsky, and Carlo Zaniolo. Optimal sampling from sliding windows. *J. Comput. Syst. Sci.*, 78(1):260–272, 2012.
7. 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.
8. 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.
9. 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.
10. Graham Cormode and Hossein Jowhari. L p samplers and their applications: A survey. *ACM Computing Surveys (CSUR)*, 52(1):1–31, 2019.
11. 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.
12. 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.
13. 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.

14. 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.
15. Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.
16. 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.
17. 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.
18. 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.
19. 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.
20. 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.
21. 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.
22. Sumit Ganguly. Counting distinct items over update streams. *Theoretical Computer Science*, 378(3):211–222, 2007.
23. 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.
24. 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.
25. 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.
26. 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.
27. William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space, 1984.
28. 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.
29. 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.

30. 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.
31. Bala Kalyanasundaram and Georg Schnitger. The probabilistic communication complexity of set intersection. *SIAM J. Discret. Math.*, 5(4):545–557, 1992.
32. Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 2007.
33. 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.
34. 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.
35. Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th FOCS*, pages 94–103. IEEE Computer Society Press, October 2007.
36. Luca Melis, George Danezis, and Emiliano De Cristofaro. Efficient private statistics with succinct sketches. In *NDSS 2016*. The Internet Society, February 2016.
37. Morteza Monemizadeh and David P. Woodruff. 1-pass relative-error lp-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.
38. Christian Mouchet, Juan Ramón Troncoso-Pastoriza, Jean-Philippe Bossuat, and Jean-Pierre Hubaux. Multiparty homomorphic encryption from ring-learning-with-errors. *PoPETs*, 2021(4):291–311, October 2021.
39. Manas Pathak, Shantanu Rane, and Bhiksha Raj. Multiparty differential privacy via aggregation of locally trained classifiers. *Advances in neural information processing systems*, 23, 2010.
40. 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.
41. 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.
42. 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.
43. Alexander A. Razborov. On the distributional complexity of disjointness. *Theor. Comput. Sci.*, 106(2):385–390, 1992.
44. 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.
45. 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.

46. 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.

A Definitions

We assume that readers are familiar with security notions of standard cryptographic primitives [32] and formal definitions of a protocol securely realizing an ideal functionality (cf. [18]).

A.1 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 [33].

A.2 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 (Differential privacy with a trusted curator [12,14]). *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 [5,44,8]. 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 $\mathsf{view}^{\Pi}_{A}(\mathbf{d}_1, \mathbf{d}_2)$ be the view of the protocol to A (consisting of input, the random tape, the protocol transcript, and the output).

Definition 8. *Let $\epsilon > 0$ and $0 \leq \delta < 1$. 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,*

$$\Pr[\mathcal{D}(\mathsf{view}^{\Pi}_{A}(\mathbf{d}_1, \mathbf{d}_2), 1^{\lambda}) = 1] \leq e^{\epsilon} \cdot \Pr[\mathcal{D}(\mathsf{view}^{\Pi}_{A}(\mathbf{d}'_1, \mathbf{d}'_2), 1^{\lambda}) = 1] + \delta + \mathsf{negl}(\lambda)$$

B 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.

$C_{\mathsf{coinflip}}(\|\mathbf{w}_1\|_1, \{r_{1,j}\}_{j=1}^{\lambda}, b_1, \|\mathbf{w}_2\|_1, \{r_{2,j}\}_{j=1}^{\lambda}, b_2)$ $\quad \triangleright$ 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 $\|\mathbf{w}_1\|_1, \|\mathbf{w}_2\|_1, \{r_{1,j}\}, \{r_{2,j}\} \in \{0,1\}^{\lambda}$, 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 $\mathsf{mask} = 0^{\lambda-h}1^h$ 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 $2^{h-1} \leq s < 2^h$. This computation can be done by checking all possible candidates of h one by one in $O(\lambda)$ 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 $r_{j^*} \leq s$. If there is no such j^* output error.
5. Compute $b = b_1 \oplus b_2$.
6. If $r_{j^*} \leq s_1$, 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 $\Pr[r_{j^*} \leq s_1 | r_{j^*} \leq s] = \frac{s_1}{s} = p$.

C Proofs

C.1 Proof of Theorem 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 $\mathcal{F}_{\mathsf{osample(L_1)}}$ and Sim returns to it three random values in $\mathbb{Z}_n$. Party 1 sends its input to the second three invocations of $\mathcal{F}_{\mathsf{osample(L_1)}}$ 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 $\mathcal{F}_{\mathsf{osample(L_1)}}$ and Sim returns to it three random values in $\mathbb{Z}_n$. Party 1 sends its input to the second three invocations of $\mathcal{F}_{\mathsf{osample(L_1)}}$ 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.

C.2 Proof of Theorem 2

We first give the simulation of the sender. The simulator proceeds as follows:

- Send $\mathbf{w}$ to $\mathcal{F}_{\mathsf{osample(L_1)}}$ 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 $\mathcal{F}_{\mathsf{osample(L_1)}}$.
- 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.

C.3 Proof of Theorem 3

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.

C.4 Proof of Lemma 5

We want to show that

$$\begin{aligned}\Pr_{D_{L_p}(\mathbf{w}_1,\mathbf{w}_2)}[i] &= \frac{(w_{1,i}+w_{2,i})^p}{||\mathbf{w}_1+\mathbf{w}_2||_p^p}\\ &= \frac{(w_{1,i}+w_{2,i})^p}{c\cdot(||\mathbf{w}_1||_p^p+||\mathbf{w}_2||_p^p)}\\ &\le \frac{2^{p-1}(w_{1,i}^p+w_{2,i}^p)}{c\cdot(||\mathbf{w}_1||_p^p+||\mathbf{w}_2||_p^p)}\\ &= 2^{p-1}/c\cdot\Pr_{D_{\mathsf{ignore},p}(\mathbf{w}_1,\mathbf{w}_2)}[i].\end{aligned}$$

The inequality holds due to Jensen's inequality with convex function $f(x)=x^p$:

$$\begin{aligned}1/2^p\cdot(w_{1,i}+w_{2,i})^p &= f(1/2\cdot w_{i,1}+1/2\cdot w_{i,2})\\ &\le 1/2\cdot f(w_{i,1})+1/2\cdot f(w_{2,i})\\ &= 1/2(w_{i,1}^p+w_{i,2}^p).\end{aligned}$$

This completes the proof of Lemma 5.

C.5 Proof of Lemma 6

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 $i\in[n]$,

$$\Pr_{D_{L_p}(\mathbf{w}_1,\mathbf{w}_2)}[i]\le 2^p/c\cdot\Pr_{D_{\mathsf{ignore},p}(\mathbf{w}_1,\mathbf{w}_2)}[i], \tag{3}$$

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 (3) 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 (3) is met due to Lemma 5. Finally, each round has $\tilde{O}(1)$ communication, since Π_{ignore} has communication $\tilde{O}(1)$ (by Lemma 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_p} has expected communication $\tilde{O}(1)$ and worst case (with all but negligible probability) communication $\tilde{O}(1)$.

C.6 Proof of Lemma 10

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 [11]):

$$\mu = \frac{z}{\langle \mathbf{w}_1, \mathbf{w}_1 \rangle} \cdot \mathbf{w}_1, \qquad \mathbf{\Sigma} = \mathbf{I} - \frac{\mathbf{w}_1 \mathbf{w}_1^T}{\langle \mathbf{w}_1, \mathbf{w}_1 \rangle}$$

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 [34] 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

$$\sigma' = \mathbf{w}_2^T \boldsymbol{\Sigma} \mathbf{w}_2 = \mathbf{w}_2^T \mathbf{w}_2 - \frac{\mathbf{w}_2^T \mathbf{w}_1 \mathbf{w}_1^T \mathbf{w}_2}{\langle \mathbf{w}_1, \mathbf{w}_1 \rangle} = \langle \mathbf{w}_2, \mathbf{w}_2 \rangle - \frac{(\langle \mathbf{w}_1, \mathbf{w}_2 \rangle)^2}{\langle \mathbf{w}_1, \mathbf{w}_1 \rangle}$$

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)

D 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 $\Delta := x - \tilde{x} \cdot v$ 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 $\mu$.
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'_is 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'_is 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 [25, 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 9 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 $i = 1, \ldots, n$, 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 $\mathbf{v} = (v_1, \ldots, v_m)$. 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 9.

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 9. *Protocol 9 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.

E L_1 Sampling Protocol with Secret Shared Output

We give a more formal description of the functionality $\mathcal{F}^{ss}_{L_1}$.

$\mathcal{F}_{L_1}$: 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 $\mathcal{F}^{ss}_{L_1}$ in the $(\mathcal{F}_{\mathsf{osample(L_1)}}, \mathcal{F}_{\mathsf{biasCoin}})$-hybrid.

Protocol 10 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.

Security of the protocol can be shown similarly to Theorem 1.