Benchmarking Secure Sampling Protocols for Differential Privacy

Yucheng Fu
University of Virginia
Charlottesville, USA
zdp8uu@virginia.edu

Tianhao Wang
University of Virginia
Charlottesville, USA
tianhao@virginia.edu

Abstract

Differential privacy (DP) is widely employed to provide privacy protection for individuals by limiting information leakage from the aggregated data. Two well-known models of DP are the *central model* and the *local model*. The former requires a trustworthy server for data aggregation, while the latter requires individuals to add noise, significantly decreasing the utility of aggregated results. Recently, many studies have proposed to achieve DP with Secure Multi-party Computation (MPC) in distributed settings, namely, the *distributed model*, which has utility comparable to *central model* while, under specific security assumptions, preventing parties from obtaining others' information. One challenge of realizing DP in *distributed model* is efficiently sampling noise with MPC. Although many secure sampling methods have been proposed, they have different security assumptions and isolated theoretical analyses. There is a lack of experimental evaluations to measure and compare their performances. We fill this gap by benchmarking existing sampling protocols in MPC and performing comprehensive measurements of their efficiency. First, we present a taxonomy of the underlying techniques of these sampling protocols. Second, we extend widely used distributed noise generation protocols to be resilient against Byzantine attackers. Third, we implement discrete sampling protocols and align their security settings for a fair comparison. We then conduct an extensive evaluation to study their efficiency and utility. Our experiments show that (1) malicious protocols based on a technique called bitwise sampling are more efficient than other methods, and using an oblivious data structure can reduce the circuit size in high-security regimes, (2) the cost of realizing malicious security is high, under the assumption of semi-honest, using a method named distributed noise generation is much more efficient, and (3) the utility loss caused by sampling noise in MPC is small, which to a certain extent eliminates utility concerns when using the DDP protocol in practice. We open-source our code at https://github.com/yuchengxj/Secure-sampling-benchmark.

CCS Concepts

• **Security and privacy → Privacy-preserving protocols**.

Keywords

Differential Privacy, Privacy-Preserving Protocol, Secure Multi-party Computation

CCS '24, October 14–18, 2024, Salt Lake City, UT, USA.

ACM ISBN 979-8-4007-0636-3/24/10
https://doi.org/10.1145/3658644.3690257

ACM Reference Format:
Yucheng Fu and Tianhao Wang. 2024. Benchmarking Secure Sampling Protocols for Differential Privacy. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024.* Salt Lake City, UT, USA , 15 pages. https://doi.org/10.1145/3658644.3690257

1 Introduction

Differential Privacy (DP) is a strong notion of privacy-preserving algorithms [28]. DP has been widely used in many scenarios, such as the analysis of personal interest, medical analysis, and machine learning. The classic definition of DP assumes that a trusted central server can collect sensitive user information and then add noise to the results of specific queries, namely, the *central model*. However, in situations where the server is not trustworthy, this definition raises privacy concerns. One possible solution is *local model*, also known as Local DP (LDP) [62], where each user perturbs the input locally and then sends the results to the server. However, the *local model* usually requires significantly more samples to achieve the same utility as the *central model*.

An alternative is to take advantage of *secure multi-party computation* (MPC), which enables users to jointly compute a function without revealing their inputs. By evaluating the aggregation function and then adding perturbations to the output in MPC, multiple users can obtain the final result satisfying DP without a trustworthy server. This is also called Distributed DP (DDP). The main challenge to realize DDP is how to produce random noise in MPC. Recent works have proposed many *sampling protocols* to efficiently sample noise from particular distributions [4, 14, 18, 27, 35, 39, 48].

Definition 1.1 (Distributed Differential Privacy). *A randomized protocol Π_f implemented among m computing parties $p = \{p_0, ..., p_{m-1}\}$, satisfies Distributed Differential Privacy w.r.t. a coalition $c \subset p$ of semi-honest computing parties of size t, if the following condition holds: for any neighbouring datasets D, D' differing in a single entry, and any possible set S of views for protocol Π,*

$$\Pr\left[\mathsf{VIEW}_{\Pi}^{p}(D) \in S\right] \leq e^{\epsilon}\Pr\left[\mathsf{VIEW}_{\Pi}^{p}(D') \in S\right] + \delta_{\kappa}, \qquad (1)$$

where δ_κ is a negligible term associated with a security parameter κ.

The sampling protocols work as bridges between MPC and DP, significantly affecting privacy-preserving algorithms' performance in the distributed setting. We give an example in Figure 1 to show the running time of using different sampling protocols to generate $n = 4096$ noise variable under security parameters $\lambda \in \{64, 128, 256, 512\}$. As we can see, the efficiency of sampling protocols varies, and an inappropriate choice of protocol can result in a long execution time in practice. While these sampling protocols can be integrated into similar DDP pipelines, their methods vary and have different security assumptions. More importantly,

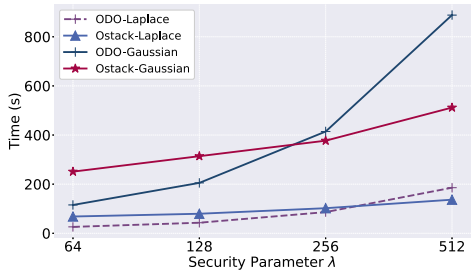


Figure 1: The running time of sampling protocols for $n = 4096$ discrete Laplace (ODO-Laplace, Ostack-Laplace) and Gaussian (ODO-Gaussian, Ostack-Gaussian) samples under different security parameters $\lambda \in \{64, 128, 256, 512\}$.

prior evaluations of these methods were conducted in isolation, with varying settings. The lack of a consistent evaluation hinders researchers from understanding and selecting the most efficient sampling algorithms in diverse settings, which motivates us to benchmark different sampling protocols and conduct comprehensive evaluations. Our contributions are as follows.

(1) Review and Taxonomy. We first review existing sampling protocols for DDP and propose a taxonomy to categorize them based on their sampling techniques into three general approaches: *distributed noise generation*, *uniform transformations*, and *bitwise sampling*. We also analyze the security model of these protocols and identify the gap in the semi-honest *distributed noise generation*.

(2) Benchmark and Alignment. We implement existing sampling protocols with the MP-SPDZ framework [42]. Our implementations are versatile and compatible with various underlying MPC protocols, which also work as a library, enabling practitioners to realize DDP on their aggregation results in MPC easily. In the benchmark, we also extend the *distributed noise generation* with an additional MPC implementation of a statistical test, which checks *poisoning attacks* to partial noise. We also create a framework that aligns the statistical distances by different sampling protocols. This guarantees protocols can achieve identical DDP protection, where we can fairly compare their efficiency.

(3) Experimental Evaluation. With the benchmark, we conduct comprehensive experiments, aiming to understand the question of which is the most effective sampling protocol under different scenarios. We vary the targeted number of samples, the security parameter, the privacy budget, and the number of computing parties to measure the efficiency of different sampling protocols. Our metric spans from empirical results like running time and communication to static indicators like the number of AND gates and random bits input from parties. We also instantiate the basic counting query on the real-world dataset to verify secure sampling protocols' utilities.

(4) Findings. Given the experimental results, we identify two interesting trends: protocols relying on bitwise sampling tend to be more efficient than alternative methods, and employing oblivious data structures can reduce the circuit size when heightened security is required. We give detailed guidelines for choosing the proper sampling protocol when considering different scenarios. Additionally, through our empirical study, we demonstrate that noise sampled in DDP and pure CDP have almost the same utility guarantee, which to a certain extent eliminates utility concerns when using the DDP protocol in practice.

Roadmap. The remainder of this paper is organized as follows. In Section 2, we introduce related works. Section 3 provides the preliminary for differential privacy and secure multi-party computation. In Section 4, we review existing sampling protocols. In Section 5, we analyse and extend widely used distributed noise generation protocols. In Section 6, we describe our benchmark and categorize their statistical distance with the same framework. Section 7 presents evaluation results. We give some takeaways from our evaluation in Section 8 and a conclusion in Section 9.

2 Related Work

Differential Privacy. Differential privacy (DP) [28, 29] is a strong notion preventing individuals from being inferred from the aggregated data. In the central DP (CDP) setting which assumes the data curator is trusted, many types of data statistics methods with privacy guarantee have been proposed, including range query [36, 49], data stream release [19, 56], synthetic dataset generation [54, 65] and machine learning [1, 60], etc. Moreover, a more stringent definition named local DP (LDP) [40] is widely used when the curator is untrusted. There are also many works designing LDP algorithms to protect the privacy of users against the adversarial curator [23, 50, 55, 64]. However, because LDP perturbs each user's input, the final aggregated results have poorer utility than CDP. An alternative is called the shuffle DP [21, 22], which uses a non-colluding shuffler to permute LDP random reports from clients. The utility in the shuffle model can be superior to the LDP setting, but it remains less optimal than CDP [5].

Distributed Differential Privacy. We focus on another approach that leverages secure multi-party computation (MPC). It can achieve the same trust model as shuffle DP (introducing a non-colluding party) but can be more versatile. It achieves better utility than shuffle DP, but with a sacrifice on computational cost. We call this distributed DP (DDP). Several works have proposed practical MPC protocols to realize different types of DDP statistics, including median queries [12, 13], heavy hitter [14, 15], graph query [51], and machine learning [22, 48]. Sampling noise in MPC is the foundation for achieving DDP. One branch of sampling protocol is distributed noise generation [4, 14, 35, 37, 39]. It lets parties locally sample noise, which is then summed in MPC. This approach can be easily and efficiently implemented. However, it can only achieve semi-honest security, and the byzantine adversaries can use incorrect partial noise to violate differential privacy [27, 41, 61]. To realize malicious security, Dwork et al. [27] propose to first securely sample biased coins and compose them into the sample from the geometric distribution. Subsequently, Champion et al. [18] construct an oblivious data structure to improve the efficiency of sampling biased coins. Wei et al. [59] use rejection sampling to convert the discrete Laplace samples into discrete Gaussian samples, where they use the method in Dwork et al. [27] as a building block. Other transform-based protocols [30, 48] require float-point non-linear arithmetic, which is significantly slower in MPC protocols. Biswas et al. [11] propose a verifiable mechanism sampling noise from Binomial distribution to realize DP. Due to the unbiased sampled coin, it can not directly be applied to sample noise from Laplace distribution.

While there are different protocols using various approaches, we identify a lack of comprehensive understanding of their security guarantees and performance under different settings. In this paper, we benchmark widely used sampling protocols to conduct evaluations.

3 Preliminaries

3.1 Differential Privacy

Differential privacy (DP) is a privacy notion to protect the privacy of any specific individuals under the aggregate statistics.

Definition 3.1 (Differential Privacy [28]). *A randomized algorithm* $\mathrm{M} : \mathbb{F} \to \mathbb{O}$ *is* (ϵ, δ)*-differential privacy (DP) if for any pair of neighbouring datasets* $D, D' \in \mathbb{F}$ *that differ by a single record, and for any possible subset* S *of* M*'s output,*

$$\Pr\left[\mathrm{M}(D) \in S\right] \leq e^{\epsilon} \Pr\left[\mathrm{M}(D') \in S\right] + \delta, \tag{2}$$

where the parameter $\epsilon \geq 0$ *denotes the privacy budget, and* $\delta \geq 0$ *denotes the probability that privacy is violated.*

When $\delta \neq 0$, the mechanism is approximate DP ((ϵ, δ)-DP). Otherwise, we call it pure DP (ϵ-DP).

Primitives for DP. In order to achieve DP for numerical statistics, one can add noise to the function output. Let Δ be the *sensitivity* of function f, which is the largest change to the function output after changing a single record.

A common way to achieve ϵ-DP is adding noise sampled from the Laplace distribution with parameter Δ/ϵ (Δ is measured in l_1 distance), i.e., the Laplace mechanism [29].

Theorem 3.2 (Laplace Mechanism [29]). *Given a function* $f: \mathbb{F} \to \mathbb{O}^k$*, and a dataset* $D \in \mathbb{F}$*, the Laplace Mechanism is defined as* $\mathrm{M}_{\mathsf{Lap}}(D, f(\cdot), \epsilon) = f(D) + (L_0, ..., L_{k-1})$*, where* L_i *are i.i.d. random variables sampled from zero-mean Laplace distribution* $\mathsf{Lap}\left(\frac{\Delta}{\epsilon}\right)$.

Another mechanism to achieve DP is the Gaussian Mechanism M_{N}, which adds noise sampled from the zero-mean Gaussian distribution $\mathsf{N}\left(\sigma^2\right)$ and the sensitivity Δ is measured by l_2 distance.

Theorem 3.3 (Gaussian Mechanism [29]). *Let* $\epsilon \in (0, 1)$*. The Gaussian Mechanism with parameter* $\sigma = \Delta\sqrt{2\ln(1.25/\delta)}/\epsilon$ *satisfies* (ϵ, δ)*-DP.*

There are recent improved results [6, 47] about Gaussian mechanism. In this paper, for presentation simplicity, we use the classic result. There are also discrete versions of these mechanisms with similar privacy guarantees.

Theorem 3.4 (Discrete Laplace Mechanism). *Given a function* $f: \mathbb{F} \to \mathbb{O}^k$*, and a dataset* $D \in \mathbb{F}$*, the discrete Laplace Mechanism is defined as* $\mathrm{M}_{\mathsf{Lap}_{\mathbb{Z}}}(D, f(\cdot), \epsilon) = f(D) + (L_0, ..., L_{k-1})$*, where* L_i *are i.i.d. random variables sampled from zero-mean discrete Laplace distribution* $\mathsf{Lap}_{\mathbb{Z}}\left(\frac{\Delta}{\epsilon}\right)$*. And the* $\mathrm{M}_{\mathsf{Lap}_{\mathbb{Z}}}$ *satisfies* ϵ*-DP.*

Theorem 3.5 (Discrete Gaussian Mechanism). *Given a function* $f: \mathbb{F} \to \mathbb{O}^k$*, and a dataset* $D \in \mathbb{F}$*, the discrete Gaussian Mechanism is defined as* $\mathrm{M}_{\mathsf{N}_{\mathbb{Z}}}(D, f(\cdot), \epsilon) = f(D) + (Y_0, ..., Y_{k-1})$*, where* Y_i *are i.i.d. random variables sampled from zero-mean discrete Gaussian distribution* $\mathsf{N}_{\mathbb{Z}}\left(\sigma^2\right)$*. And the* $\mathrm{M}_{\mathsf{N}_{\mathbb{Z}}}$ *satisfies* (ϵ, δ)*-DP for* $\delta = \mathsf{Pr}_{Y \leftarrow \mathsf{N}_{\mathbb{Z}}(\sigma^2)}\left[Y > \frac{\epsilon\sigma^2}{\Delta} - \frac{\Delta}{2}\right] - e^{\epsilon} \cdot \mathsf{Pr}_{Y \leftarrow \mathsf{N}_{\mathbb{Z}}(\sigma^2)}\left[Y > \frac{\epsilon\sigma^2}{\Delta} + \frac{\Delta}{2}\right]$.

In this work, we benchmark DP protocols in MPC. We focus on discrete protocols because those are more naturally supported in MPC, and most existing work is discrete. That said, our library also includes continuous protocols. Other primitives, like Exponential Mechanism, Report Noisy Max, and Sparse Vector Technique, can be reduced to sampling Laplace or Gaussian noise. We present their details in Appendix A of our technical report [33].

3.2 Secure Multi-party Computation

Secure Multiparty Computation (MPC) [34, 63] allows a set of parties to jointly compute a function $y = f(D_0, ..., D_{m-1})$ without revealing their inputs D_i ($0 \leq i \leq m-1$). After the computation, all the parties can only know the result y. Currently, the two main paradigms to implement MPC are *garble circuits* [44, 46] and *secret sharing* [9, 24]. In this paper, we focus on a secret sharing scheme, which offers better scalability in the multi-party setting; our evaluation also involves garbled circuits for two-party settings.

The secret-sharing-based-MPC splits each party's input d_i into m pieces of shares $\langle d_i \rangle_j$ ($0 \leq j \leq m-1$) by a `Share` function. Next, each party i runs the protocol with m shares $\langle d_1 \rangle_i, ..., \langle d_{m-1} \rangle_i$ and gets the shares of output $\langle y \rangle_i$, which are used to reconstruct the plain text output y with a `Rec` function. In a (t, m)-secret sharing protocol, t shares are sufficient to reconstruct the plain text with `Rec`. For simplicity, we use $\langle s \rangle$ to denote shares $\{\langle s \rangle_1, ..., \langle s \rangle_{m-1}\}$ of s among m parties.

The adversary models of MPC protocols can be classified into the *semi-honest* model and the *malicious* model. The former assumes that the adversaries are curious about others' private data but still follow the protocol, even if their deviation may not be detected; the latter assumes that adversaries can deviate from the protocol arbitrarily, thus some detection/prevention scheme has to be implemented. For more details, one can refer to [31].

Recently, some frameworks, such as SCALE-MAMBA [2] and MP-SDPZ [42] have improved the MPC performance. In this paper, we implement existing solutions in both semi-honest and malicious settings for realizing DP in MPC using MP-SPDZ [42] framework and compare their performance. We use MP-SPDZ since it has been adopted in many MPC applications [14, 25, 38, 59] and supports various popular underlining protocols in both secret shares and garbled circuits. The MPC subprotocols mentioned in this paper are listed in Appendix C of [33]. In most cases, they operate on secret shares.

3.3 Distributed Differential Privacy

The conventional definition of DP holds against computationally unbounded adversaries. In this paper, we consider distributed DP (DDP) against computationally bounded adversaries in the distributed setting [30].

Specifically, in DDP, we assume there is a set of m computing parties and m' input parties (an input party can also serve as a computing party). Each input party i gives a private input d_i to the computationally bounded, untrusted and non-colluding computing parties to construct a dataset D. Then, computing parties jointly compute the final statistic results on D by running the protocol Π_f. Let $\mathsf{VIEW}^p_{\Pi}(D)$ be the view of any computing party p when executing the protocol Π on dataset D, including all exchanged

messages and internal state, and κ be the view security parameter. The protocol Π_f satisfies the following definition:

Definition 3.6 (Distributed Differential Privacy [30]). *A randomized protocol Π_f implemented among m computing parties $p = \{p_0, ..., p_{m-1}\}$, satisfies Distributed Differential Privacy w.r.t. a coalition $c \subset p$ of semi-honest computing parties of size t, if the following condition holds: for any neighbouring datasets D, D′ differing in a single entry, and any possible set S of views for protocol Π,*

$$\Pr\left[\mathsf{VIEW}^p_\Pi(D) \in S\right] \leq e^\epsilon \Pr\left[\mathsf{VIEW}^p_\Pi(D') \in S\right] + \delta_\kappa, \qquad (3)$$

where δ_κ is a negligible term associated with a security parameter κ.

General Pipeline of DDP. At the high level, the end-to-end process of a DDP protocol works in four steps. In the process of aggregation statistic, each input party i sends its d_i to the untrusted but non-colluding computing parties to form a combined dataset $D = \{d_0, ..., d_{m'-1}\}$. Each d_i can be a single data point (in the fully distributed setting) or a collection of samples (the input party already collects data in raw format). Formally, the pipeline is as follows.

- *Secret Sharing.* Each input party i splits its data d_i into shares $\langle d_i \rangle = \mathsf{Share}(d_i)$ and sends them to m computing parties. After that, the input parties can go offline.
- *Statistics Aggregation.* The computing parties run secret-sharing-based protocols Π_f of some function f to get the shares of aggregation result $\langle f(D) \rangle$.
- *Noise Sampling.* The computing parties run an additional secure sampling protocol $\Pi_s(x_0, ..., x_{m-1})$ (where x_i is from computing party i) to get the shares of noises $\langle \eta \rangle$.
- *Addition and Reconstruction.* The computing parties add the noise to the aggregation result to get shares of the final output $\langle y \rangle = \langle f(D) \rangle + \langle \eta \rangle$. Then, the reconstruction function is called to reveal the differential private result $\mathsf{Rec}(\langle y \rangle)$. Note that $\langle f \rangle$ and $\langle \eta \rangle$ can be in vector forms.

4 Overview of Sampling Protocols

Now, we focus on secure sampling protocols Π_s for the Laplace or Gaussian mechanism proposed by recent studies. According to their underlying sampling process, we classify these methods into three main categories: (1) *distributed noise generation*, (2) *uniform transformation*, and (3) *bitwise sampling*. Table 1 shows a summary of methods based on this categorization (along with two more dimensions of what DP mechanism they support and the adversary model in MPC). In what follows, we review these approaches in more detail and discuss the limitations of semi-honest protocols.

4.1 Semi-honest Sampling Protocols

Distributed noise generation (DNG) is the simplest way to add DP noise in MPC. In this approach, each party locally samples partial noise and sends it to the MPC protocol Π_s. These partial noises can be combined to obtain the required distribution. The combining process usually requires only ADD operations in MPC, which is highly efficient. Below, we briefly review the existing works using DNG protocols for Laplace and Gaussian noise.

DNG for Gaussian Noise [27, 37, 39]. The distributed sampling for Gaussian noise is direct since the sum of Gaussian variables is also a Gaussian variable. Thus the generation of $Y \sim \mathsf{N}(\sigma^2)$ can be achieved by $Y = \sum_{i=0}^{m-1} Y_i$, where Y_i is drawn by party i with $Y_i \sim \mathsf{N}\left(\frac{\sigma^2}{m}\right)$ [37, 39]. In our evaluation, we assume that the summation of discrete Gaussian samples is a Gaussian sample since they have been proven extremely close [39].

DNG for Laplace Noise [14, 35]. Different from Gaussians, Laplacians do not add up to Laplacian. However, distributed parties can generate other kinds of noise that add up to a Laplace noise. In particular, partial noises from the Negative Binomial distribution $Y_i \sim \mathsf{NB}(1/m, e^{-\frac{\Delta}{\epsilon}})$ can add up to a geometric noise. Then, an additional unbiased bit is used to convert them to discrete Laplace samples. Goryczka et al. [35] summarized the partial noise to generate continuous Laplace and geometric samples. Böhler et al.[14] also apply distributed noise generation to compute heavy hitters.

We summarize the partial noise generated by computing parties locally and arithmetics/operations to obtain target noise in MPC in Table 9 in Appendix D of our technical report [33].

There is a privacy risk in DNG protocols, namely, the colluded attack. The corrupt parties can collude with each other to "subtract away" their noise samples from the final result and reduce the noise to below the amount needed for the desired differential privacy guarantee. Therefore, for example, if 50% parties collude, each party needs to add additional (twice as much) noise as in the central model because up to half of the parties may be corrupt and subtract their portion of the noise. For the original partial Gaussian noise with variance σ^2, when there are α proportion of colluded parties, each party should provide Gaussian noise with variance $\hat{\sigma}^2$,

$$\hat{\sigma}^2 = \sigma^2/(1-\alpha) \qquad (4)$$

Such additional noise will cause a reduction of utility for the final results. We measure this impact in Section 7.6.3.

4.2 Poisoning Attacks to Semi-honest Protocols

The most significant limitation of semi-honest protocols is that they initially provide no guarantee when parties change their input arbitrarily, i.e., poisoning attack. It can result in incorrect aggregated noise to destroy privacy or utility. As for the computing parties cooperatively sampling noise, there are two types of attacks in the presence of malicious computing parties. Also, there is a possible attack named data poisoning attacks, which is caused by the malicious input parties.

Zero Noise. It is possible that an adversarial server keeps providing 0 as the partial noise, which causes the violation of the desired guarantee for privacy since the protocol eventually adds noises with smaller variance to the outputs. One possible solution is that each computing party provide a larger magnitude of noise. Taking the DNG protocol for the Gaussian mechanism proposed by Dwork as an example [27], m computing parties can sample partial Gaussian noise with variance $\frac{3\sigma^2}{2m}$, which can still realize required DP in the setting of at least $\frac{2}{3}$ honest servers. The required number of honest

Table 1: Summary of existing secure sampling methods to achieve DP with MPC.

Sampling Process	Mechanism	Adversary Model	Method
Distributed Noise Generation	Continuous Laplace	Semi-honest	Summing Gamma Noise [4, 14]
	Continuous Laplace	Semi-honest	Summing Laplace Noise [35]
	Continuous Gaussian	Semi-honest	Summing Gaussian Noise [37]
	Discrete Gaussian	Semi-honest	Convolution [39]
Uniform Transformation	Continuous Laplace	Malicious	Inverse Transformation [30]
	Discrete Laplace	Malicious	Inverse Transformation [30]
	Continuous Gaussian	Malicious	Box and Muller Transformation [48]
Bitwise Sampling	Discrete Laplace	Malicious	ODO Sampling [27]
	Discrete Laplace	Malicious	Ostack-based Sampling [18]
	Discrete Gaussian	Malicious	Rejection Sampling [59]

computing parties can be reduced to one if each party provides a noise with variance σ^2. Although this method provides a privacy guarantee while resulting in an expected error of $m\sigma^2$ [41].

Sampling Larger Amount of Noise. Another poisoning attack is that the malicious computing party generates a significant amount of noise as the partial noise input to the MPC protocol. Due to the property of MPC, this illegal input can not be viewed by other participants. Moreover, the numerical noise from parties is unbounded and directly added to the numerical answer, significantly decreasing the protocol's utility. Although [61] uses non-MPC methods, applying statistical tests can check whether inputs from computing parties fall in some interval and whether they can pass the null hypothesis that they are sampled from the specific distribution, these methods can not be directly applied to the MPC setting since the noise are no longer accessible to be tested. Otherwise, the statistical test in MPC needs to be designed.

Data Poisoning Attack. The data poisoning attack also exists in the *Statistic Aggregation* phase of the DDP pipeline. The input parties can manipulate the inputs to skew the query results before the *Noise Sampling* phase, which is out of the scope of secure sampling protocols. Next, we discuss such malicious settings and corresponding defense approaches other than secure sampling.

The secure aggregation in federated learning averages gradient across multiple local models, where the DP sampling can be further used to provide output protection [61]. However, the malicious client can input large gradients to skew the aggregated gradients. To this end, the zero-knowledge proof (ZKP) is often used by the computing parties, which checks whether the inputs satisfy L_1, L_2 and L_∞ bounds. One direct solution is the Bulletproof protocol for bound checking [16]. There is a sequence of studies on checking gradient bound in federated learning [7, 8, 52]. Such strategies can also be applied in other scenarios requiring mean statistics.

The adversary in DDP statistics queries is similar to those in LDP settings, where each input party provides invalid inputs or dishonest responses. For example, the DDP heavy hitter collects each user's binary response [14]. With ZKP, each input is proved to be in $\{0, 1\}$. Thus, a coalition of t malicious parties only changes each aggregated count of values at most t, which can be further mitigated by the idea of normalization by observing consistency [58]. Also, in the DDP median, the malicious inconsistency of inputted value-rank pairs should be checked [13]. Note that the defense strategies for the data poisoning attack can only mitigate the effect of poisoning attacks instead of eliminating them since the inputs are determined by clients.

There are standard ways to transform semi-honest to malicious security [31], but they focus on the process of computation rather than inputs, assuming data is deterministic and cannot test random variables. We propose a method based on the Kolmogorov–Smirnov (KS) test later in Section 5. The high level of our KS test is to reject the hypothesis that the generated samples come from a specific distribution when the results are poisoned. A common way to realize malicious for the randomness is using XOR. The secure sampling protocols for DP with XOR originated from the protocol proposed by Dwork et al. [27] (we call it the ODO protocol in the following). The idea is simple: XORing the input bits from each party can obtain randomness as long as at least one party is inputting a truly random bit. Compared to previous methods, this is extremely simple but much more efficient. Still, the drawback is this only gives a random bitstring, and we need to design sampling protocols to transform randomness into specific distributions.

4.3 Malicious Sampling Protocols

Uniform transformation. Given the random bitstring, each from $\{0, 1\}$, constructed from XOR described above. Next, the bitstring can be viewed as a uniformly random number $u \sim \mathrm{U}(0, 1)$. Next, the most straightforward idea is to compare it with a 'distribution table' from the target distributions. That is, one can pre-compute the cumulative density function (CDF) $F(x)$ of the target distribution and then compare it with a uniformly generated random number $u \in \mathrm{U}(0, 1)$ converted from the XORed random bitstring. If the u lies between $F(x_i)$ and $F(x_{i+1})$, the sample corresponds to the value x_{i+1}. However, in MPC, we have to traverse F and perform comparison to prevent sensitive data leakage, which can be slow, especially when F is a very detailed CDF. Note that one can also compute $F^{-1}(u)$ with a logarithm arithmetics to get a Laplace sample [30], where F^{-1} is the inverted CDF of the exponential distribution.

Bitwise sampling. Dwork et al. [27] proposed a protocol (we call it *ODO-sampling*) based on their observation that each bit in the

geometric sample can be sampled independently. Based on ODO, Champion et al. [18] proposed to use an oblivious stack data structure to obliviously pop and push bits (we call it *Ostack-sampling*), which avoids complete iteration over the binary expansion of bias to hide the access pattern thus improve efficiency. The procedure of ODO-sampling and Ostack-sampling is shown in Appendix B in [33]. With many biased coins from $\mathcal{B}(p)$ from different biases p, we can get the geometric distribution by concatenating them. Then, the two-sided discrete Laplace sample can be obtained by transforming the one-side geometric sample. Compared to the uniform transformation mentioned above, this approach is typically more efficient but relies on key observations (bits can be sampled independently). Finally, we can get discrete Gaussian samples by applying rejection sampling to Laplace samples [59].

4.4 Statistical Distances Caused by MPC

Our benchmark focuses on implementing discrete sampling protocols and aligning their security demand. Sampling from the continuous distribution relies on the fixed point or float point representation in MPC, and the statistical distance between the actual continuous distribution and that with a finite number representation is hard to measure. In fact, existing works [14] using continuous noise in MPC did not formally quantify additional statistical distance. In this paper, we focus on discrete noise generation in MPC.

The components of statistical distance in discrete sampling protocol also vary, and incorrectly setting protocol parameters will result in a loss of security. Thus, we summarize the allocation of security parameter δ_κ. The sources of extra statistical distance in sampling discrete noise are δ_t caused by truncating ideal distribution from $\mathbb{O} \in \mathbb{Z}$ to $\mathbb{O} \in (-N, N) \cap \mathbb{Z}$, where $N \in \mathbb{N}$ control the truncating range since MPC operates in fixed-length integers, δ_b caused by representing the bias p in a finite number when sampling biased coins, δ_r from the potential failure of rejection sampling, and δ_p from the potential failure of filling the Ostack. Note that existing works [18, 59] have analysed this for ODO-Gaussian and Ostack-Laplace, while there is no such theoretical analysis for others. We expand this landscape for all the protocols in our benchmark. More details about how to set different security parameters are given in Section 6.2.

5 Verifying DNG protocol

The poisoning attacks mentioned in Section 4.2 deviate the generated noise from its original distribution. One way to constrain the probability distribution deviation is using the Kolmogorov–Smirnov test (KS test) [3] to check the partial noise from the computing parties or the aggregated noise. The definition of the two-sample KS test is as follows.

Definition 5.1 (Two-sample KS test [10]). *Given empirical distribution functions F_a and F_b of samples a and b with length n_a and n_b, Kolmogorov–Smirnov test is defined by*

$$D_{a,b} = \sup_x |F_a(x) - F_b(x)|.$$

Algorithm 1 Check-Lap_Z

Input: Length of aggregated sample n, shares of aggregated noise $\langle y_1 \rangle, ..., \langle y_n \rangle$.
Output: The shares of Boolean variable specifying whether to reject the null hypothesis $\langle b \rangle$.
Precompute: CDF table $\langle F[j] \rangle = \langle \mathcal{F}_{\mathsf{Lap}}(j - N) \cdot n \rangle$ for $j = 1, ..., 2N - 1$, KS distance $\langle p'_a \rangle = \langle c(a) \cdot n \cdot \sqrt{\frac{n+2N+1}{n \cdot (2N+1)}} \rangle$ of significance value α.

Initialize array $\langle obs \rangle$ of size n.
Initialize $\langle d \rangle \leftarrow \langle 0 \rangle$
for $i \leftarrow 1$ **to** n **do**
  for $j \leftarrow 1$ **to** $2N - 1$ **do**
    $\langle e \rangle \leftarrow \mathbf{EQ}(\langle y_i \rangle, \langle j - N \rangle)$
    $\langle obs[i] \rangle \leftarrow \mathbf{MUX}(\langle e \rangle, \langle F[j] \rangle, \langle obs[i] \rangle)$
  end for
end for
$\mathbf{SORT}(\langle obs \rangle)$
for $i \leftarrow 1$ **to** n **do**
  $\langle t \rangle \leftarrow \mathbf{ABS}(\mathbf{SUB}(\langle i \rangle, \langle obs[i] \rangle))$
  $\langle d \rangle \leftarrow \mathbf{MUX}(\mathbf{LE}(\langle d \rangle, \langle t \rangle), \langle d \rangle, \langle t \rangle)$
end for
$\langle b \rangle \leftarrow \mathbf{LE}(\langle p' \rangle, \langle d \rangle)$
return $\langle b \rangle$

The null hypothesis that a and b are sampled from the same distribution is rejected at a significance level α if

$$D_{a,b} > c(\alpha) \sqrt{\frac{n_a + n_b}{n_a \cdot n_b}},$$

where $c(\alpha) = \sqrt{-\ln(\frac{\alpha}{2} \cdot \frac{1}{2})}$, and the significance level α is the probability of rejecting the null hypothesis when the null hypothesis is true.

Applying the KS test framework in our setting (assuming discrete Laplace; discrete Gaussian naturally follows), we hold the null hypothesis that the aggregated noises $y = y_0, ..., y_{n-1}$ are sampled from the discrete Laplace distribution $\mathsf{Lap}\left(\frac{\Delta}{\epsilon}\right)$. The KS test computes its empirical Cumulative Distribution Function (eCDF) by $F_{obs}(y') = \frac{1}{n} \sum_{i=0}^{n-1} \mathbf{1}_{y_i < y'}$, which is the number of samples smaller than y' in samples y. Next, the KS statistics $D_{obs,\mathsf{Lap}} = \sup_{y'} |F_{obs}(y') - F_{\mathsf{Lap}}(y')|$ is computed, where the second sample F_{Lap} is the true CDF of $\mathsf{Lap}\left(\frac{\Delta}{\epsilon}\right)$. Assuming that the discrete Laplace samples are truncated into $(-N, N) \cap \mathbb{Z}$, where N denote the range of finite representation of integers in DNG protocol, we can precompute a F_{Lap} of length $2N - 1$. Then, fixing the significance level α, if $D_{KS} > c(\alpha) \sqrt{\frac{n+2N-1}{n \cdot (2N-1)}}$, we can reject the null hypothesis that the aggregated samples are not from $\mathsf{Lap}\left(\frac{\Delta}{\epsilon}\right)$ at the significance level of α. In this paper, we set $\alpha = 0.05$, a widely adopted significance level. Note that a similar method is proposed in [26]. Their check is applied to the sum of noise and gradient sent by participants, aiming to find whether they are from the same distribution, whereas we directly check the noises in MPC and find whether they are from the specific distribution.

We implement the procedure of the one-sample KS test in MPC protocol Check, as shown in Algorithm 1. We assume all the partial noises and generated noise are truncated into $(-N, N) \cap \mathbb{Z}$. After the Secure Aggregation, an additional step Check is performed on the aggregated noise samples. A table F of rank based on the CDF of each element $j \in [2N-1]$, i.e. $F_{\mathsf{Lap}}(j-N) \cdot n$ is precomputed. The $p'_\alpha = c(\alpha)\sqrt{\frac{n+2N+1}{n\cdot(2N+1)}} \cdot n$ to be used for comparison is also precomputed. Here we use $F_{\mathsf{Lap}}(j-N) \cdot n$ instead of $F_{\mathsf{Lap}}(j-N)$ because the prior can be directly compared to the rank i of the element, avoiding additional divination in MPC to compute i/n and compare it with $F(x_i)$. For each aggregated sample y_i. we iterate over F to find its rank in a cumulative probability $\langle F[y_i]\rangle$ and store it in location i of list $\langle obs[i]\rangle$ (Lines 4-7). Subsequently, we sort $\langle obs[i]\rangle$ and compute the $\langle D\rangle = \sup_i |\langle i\rangle - \langle obs[i]\rangle|$. Finally, we check the validity of $\langle D\rangle$. If $\langle D\rangle < \langle p'_\alpha\rangle$, we reject the null hypothesis that the aggregated samples are not from $\mathsf{Lap}\left(\frac{\Delta}{\epsilon}\right)$.

We use the two sample KS test to constraint the KS distance between the aggregated noise and targeted noise distribution $D_{obs,\mathsf{Lap}} = \sup_{y'} |F_{obs}(y') - F_{\mathsf{Lap}}(y')|$. We can bound $D_{obs,\mathsf{Lap}}$ with probability α, the set significance level. We implement the protocol Check in our benchmark to mitigate the poisoning attacks on DNG-based protocols. We compare DNG with Check and other malicious sampling protocols in Section 7.2. Note that the KS-test is generic and can check any distribution. Therefore, it can also be applied to check the validity of each partial noise and trace back the malicious participants. Although this is slower than only checking the aggregated noise, it is currently not supported by other malicious sampling protocols. Since Check relies on traversing ordered arrays, its efficiency can also be improved by secure merging protocols [32], i.e., the equivalent values in ordered arrays $\langle y_1\rangle, \ldots \langle y_n\rangle$ and $1, \ldots, 2N-1$ can be grouped together to derive $\langle obs\rangle$, thus reduce the complexity from $O(nN)$ to $O(n+N)$.

6 Benchmark and Parameter Alignment

In this section, we first introduce our benchmark to evaluate the performance of secure sampling protocols. This benchmark includes eight MPC sampling protocols for generating samples from *discrete Laplace* and *discrete Gaussian* distributions. We focus on the discrete sampling protocols since their samples can be compared with the 'perfect' discrete samples by the statistical distance [18, 59]. Therefore, we can fix the δ_λ and compare their efficiency. Then, we give details of the relationship of security parameter λ other parameters determining the efficiency of the protocol.

6.1 Benchmark Protocols

We implemented the following eight sampling protocols: the first six are the bitwise sampling protocols. DNG-Laplace and DNG-Gaussian are DNG-based methods with Check. The transformation-Laplace is a uniform-transformation-based protocol.

- ODO-Laplace [27]. It uses the direct ODO-Sampling protocol (see Appendix B in our technical report for details [33], the protocol with the smallest number of AND gates in [27]) to produce Bernoulli samples and compose them into discrete Laplace samples. Note that generating biased bits is the basic block for bitwise sampling.
- Ostack-Laplace [18]. It generates Bernoulli samples using Ostack-sampling in Appendix B in our technical report [33], which includes two types of operation on an oblivious data structure as stacks. It supports RPOP operation to obliviously produce bits in the binary expansion of the bias p, and CPUSH to obliviously save the accepted biased coin.
- Ostack-Laplace* [18]. An improved version of Ostack-Laplace saving the number of AND gates in RPOP [18]. That is, when ϵ is in the form of $2^{-i}\ln 2$, $i = 0, 1, 2, \ldots$, one can predefine a periodic binary sequence to give the bits in p in Ostack-sampling. We implement this version and always use the largest $2^{-i}\ln 2$ that is smaller than the required ϵ as an approximation to give a slightly tighter guarantee than ϵ-DP.
- ODO-Gaussian [27, 59]. This is the implementation of [59], which uses rejection sampling to discard a portion of the samples drawn from the discrete Laplace distribution. The ODO sampling produces all the biased bits in this protocol.
- Ostack-Gaussian [18, 59]. We fill the gap between the Ostack-sampling and sampling protocol for Gaussian [59] by integrating Ostack-sampling into all the procedures that generate biased coins. We also adjust the parameters (see Table 2). We allocate a fraction of security parameter λ to compute the number of operations in Ostack-sampling, which bounds the probability of failing to produce the required number of coins.
- DNG-Laplace [35]. We implement the DNG in [35], which first generates partial noise from negative binomial distribution locally in input parties to form discrete Laplace noise. Since the above bitwise sampling method can achieve malicious security, We perform additional Check with KS tests in MPC to validate whether the aggregated noise is from the Laplace distribution, which limits the Byzantine attacks from active adversaries.
- DNG-Gaussian [39]. Similarly, we implement the DNG in [39], in which parties generate partial noises from Gaussian distribution locally. The noises are then simply summed in the data aggregation phase. We also perform the KS test in MPC to check whether the generated noise is from the Gaussian distribution.
- Transformation-Laplace [30]. This protocol is the sampling protocol for the Geometric distribution in [30], which conducts uniform sampling with the XORing technique and then transforms the uniform variable $u \in (0, 1]$ into the inverted CDF $F^{-1}(u)$ of Exponential distribution. This protocol is clearly suboptimal since it requires expansive logarithm computation in MPC.

Our benchmark[1] implemented by MP-SPDZ can also act as a library for the designers of MPC protocols. With the aggregated statistics obtained in MPC, say in secret shares, one can call our library with only one line of code to generate Laplace and Gaussian noise in MPC and add to the results, which follows the pipeline mentioned in Section 3.3 to achieve distributed differential privacy. Our benchmark is compatible with all the underlying target use case protocols in MP-SPDZ (including binary circuits like BMR, garbled circuits, and arithmetic circuits SPDZ and Shamir) as long as they support basic secure bit operations like AND and XOR.

[1] https://github.com/yuchengxj/Secure-sampling-benchmark

6.2 Parameter Alignment

We give details about the sources of statistical distance in all the protocols into four parts, summarized in Table 2. The Statistical distance is due to imperfect sampling of MPC and must be quantified (resulting in increased δ in DP, as shown in Theorem 6.2. The statistical distance is defined as follows, which is also called total variation distance.

Definition 6.1 (Statistical Distance). *Let $\mathcal{V}$ and $\mathcal{W}$ be the probability distributions over $\mathbb{F}$. The statistical distance between $\mathcal{V}$ and $\mathcal{W}$ is defined by*

$$\mathrm{SD}(\mathcal{V}, \mathcal{W}) \triangleq \frac{1}{2} \sum_{x \in \mathbb{F}} |f_{\mathcal{V}}(x) - f_{\mathcal{W}}(x)|$$

Truncation (δ_t). The statistical distance is caused by truncating the targeted discrete sample into $(-N, N) \cap \mathbb{Z}$. Since we use a finite number of bits to represent the generated noise sample, all eight protocols have δ_t in their statistical distance. The statistical distance between the truncated discrete Gaussian $\mathsf{N}_{\mathbb{Z},N}\left(\frac{2\sigma^2 \ln(1.25/\delta)}{\epsilon^2}\right)$ and discrete Gaussian $\mathsf{N}_{\mathbb{Z}}\left(\frac{2\sigma^2 \ln(1.25/\delta)}{\epsilon^2}\right)$ is $2ne^{-\frac{N^2\epsilon^2}{4\sigma^2 \ln(1.25/\delta)\Delta^2}}$ [59]. The statistical distance between truncated discrete Laplace $\mathsf{Lap}_{\mathbb{Z},N}\left(\frac{\Delta}{\epsilon}\right)$ and discrete Laplace $\mathsf{Lap}_{\mathbb{Z}}\left(\frac{\Delta}{\epsilon}\right)$ is $\frac{2n \cdot e^{-\epsilon(N-1)/\Delta}}{e^{\epsilon/\Delta}+1}$ [17]. Note that we generate samples of length κ, thus, we have $N = 2^{\kappa} + 1$.

Representing Bias (δ_b). There is a need to represent the bias p with a binary expansion of length l when sampling biased coins in ODO-Sampling, Ostack-Sampling and Transformation-Laplace. ODO-Laplace, Ostack-Laplace and Ostack-Laplace* all generate samples from the geometric distribution of length κ to approximate discrete Laplace, and the total number of samples is n, thus they all have $\delta_b = n(\kappa+1)2^{-l}$, which have been proven in [59]. Moreover, the number of additional biased coins for rejection sampling is m, resulting in $\delta_b = \frac{n}{p^*}(2\kappa + m + 2)2^{-l}$, where p^* is the acceptance rate in rejection sampling [59]. As for Trans-Laplace transforming the uniform variable $u \in (0, 1]$ into the inverted CDF $F^{-1}(u)$, the statistical distance between a fix-point number u in MPC protocol and $u \in \mathbb{R}$ is 2^{-l}. Therefore, sampling n fix-point number u has a total statistical distance of $n2^{-l}$.

Rejection Sampling (δ_r). [59] proposes to use rejection sampling to convert discrete Laplace samples to discrete Gaussian samples. Given the targeted number of discrete Gaussian samples n and actual accept rate p'_*, the number of required Laplace samples n' should be set larger than n to constrain the probability of failing to generate enough Gaussian samples after rejection sampling. The relationship between δ_r, p'_*, n and n' should be $\delta_r = e^{\frac{-2(n'p^*-n)^2}{n'}}$.

Filling Ostack (δ_p). For the protocols using Ostack to sample biased coins, there is a probability that u times of **CPUSH** operation does not fill the Ostack of size g. Formally, this probability is $\delta_p = M \cdot e^{\frac{-2(\frac{u}{2}-(g-1))^2}{u}}$, where M is the number of calls to Ostack [18]. For Ostack-Laplace and Ostack-Laplace*, using Ostack of size g to sample biased coin have $M = \kappa\lceil\frac{n}{g}\rceil$, since we need to generate n geometric samples, and each sample has a binary expansion of length κ [18]. As for Ostack-Gaussian, we have $M = (\kappa+m)\lceil\frac{n'}{g}\rceil$ because for each Laplace sample, we need to generate κ biased coins to represent the Geometric sample and m biased coins to perform rejection sampling.

Our benchmark fixes $\delta_t + \delta_b + \delta_r + \delta_p = 2^{-\lambda}$ and equally assigns $2^{-\lambda}$ to all the non-zero terms of sampling the protocol (e.g $\delta_t + \delta_b + \delta_r = \frac{2^{-\lambda}}{3}$ for ODO-Gaussian). Then, we derive the parameters in protocols. As a result, their security guarantees are aligned, and we can compare their efficiency by running time, communication, and number of AND gates. Here, we describe the Distributed Differential Privacy achieved by the benchmarked discrete sampling protocols, following the definition of DDP (Definition 3.6). The proof can be found in [59].

Theorem 6.2. *If the targeted discrete mechanism M is (ϵ, δ)-DP, then this mechanism realized by sampling protocol Π_s is $(\epsilon, \delta+\delta_\lambda)$-DP ,where $\delta_\lambda = 2(e^{\epsilon}+1)(\delta_t + \delta_b + \delta_r + \delta_p)$.*

7 Experimental Evaluation

7.1 Setup

Using our benchmark, we conduct comprehensive experiments to compare the efficiency of discrete sampling protocols in various security and privacy settings, which aims to find which protocol to use in the specific setting. Moreover, we perform a case study using the DP counting query to compare their utility with CDP settings.

We implemented the discrete sampling protocols to sample Laplace and Gaussian noise with MP-SDPZ [42] framework using Shamir-BMR [43] with honest majority since it supports three or more computing parties, which enables us to investigate the scalability of sampling protocols. Moreover, the BMR is a binary protocol that efficiently evaluates our benchmark with a large number of bit operations. Note that MP-SPDZ supports running the same implementation with a variety of underlining protocols. Without changing the code, our benchmark can also be executed with other protocols, such as Yao's protocol and SPDZ. All the results reported below were run on servers with Intel i7-11700K, Ubuntu 20.04, and 64GB memory. Our settings of range for parameters are $\lambda \in \{64, 128, 256, 512\}$, $\epsilon \in \{0.001, 0.01, 0.1, 1, 10\}$ and $n \in [2^2, 2^{18}]$ and number of computing parties $m \in [2, 8]$, in the setting of $m = 2$ we uses Yao's protocol for evaluation. We measure the efficiency of sampling protocols by the number of AND gates, running time and communication, which are the widely used metrics in related works [12, 13, 18, 59].

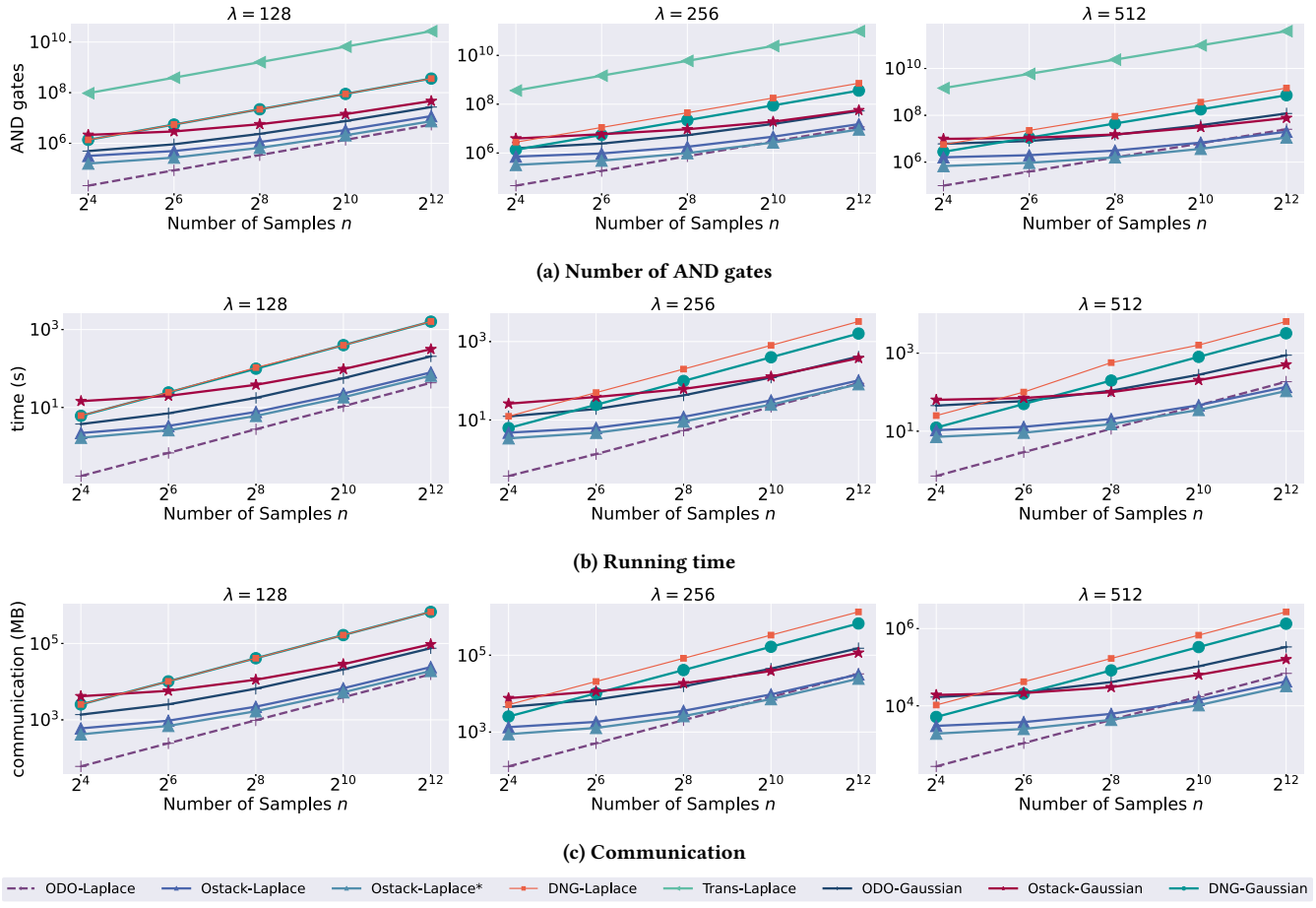
7.2 Efficiency of Sampling Protocols

In this section, we fix the privacy budget $\epsilon = 0.1$, $m = 3$ and then set the numbers of generated samples n and security parameters λ to compare the efficiency of protocols by the number of AND gates, running time and communication. We also compare the number of required random bits.

Efficiency Comparison. The number of AND gates of all the eight sampling protocols in our benchmark are shown in Figure 2a. As for running time and communication in Figure 2b and Figure 2c, we do not include the Trans-Laplace since it has a number of AND gates significantly larger than all the other protocols and it takes days for a single run. From the experimental results, we have the following observations. (1) The overall trend for all three metrics

Table 2: The relationship of statistical distance δ_λ and the parameters of sampling protocols, with $\delta_\lambda = 2(e^\epsilon + 1)(\delta_t + \delta_b + \delta_r + \delta_p)$ for all the protocols, where δ_t is caused by truncation, δ_b is caused by sampling biased coins, and the potential fail of rejection sampling and CPUSH cause δ_r and δ_p. We assign δ_λ equally to the non-zero terms for each protocol in our benchmark.

δ_λ \ Πs	ODO-Laplace	Ostack-Laplace	Ostack-Laplace*	DNG-Laplace	Trans-Laplace	ODO-Gaussian	Ostack-Gaussian	DNG-Gaussian
δ_t	$\frac{2n\cdot e^{-\epsilon(N-1)/\Delta}}{e^{\epsilon/\Delta}+1}$	$\frac{2n\cdot e^{-\epsilon(N-1)/\Delta}}{e^{\epsilon/\Delta}+1}$	$\frac{2n\cdot e^{-\epsilon(N-1)/\Delta}}{e^{\epsilon/\Delta}+1}$	$\frac{2n\cdot e^{-\epsilon(N-1)/\Delta}}{e^{\epsilon/\Delta}+1}$	$\frac{2n\cdot e^{-\epsilon(N-1)/\Delta}}{e^{\epsilon/\Delta}+1}$	$2ne^{-\frac{N^2\epsilon^2}{4\sigma^2\ln(1.25/\delta)\Delta^2}}$	$2ne^{-\frac{N^2\epsilon^2}{4\sigma^2\ln(1.25/\delta)\Delta^2}}$	$2ne^{-\frac{N^2\epsilon^2}{4\sigma^2\ln(1.25/\delta)\Delta^2}}$
δ_b	$n(\kappa+1)2^{-l}$	$n(\kappa+1)2^{-l}$	$n(\kappa+1)2^{-l}$	0	$n2^{-l}$	$\frac{n}{p^*}(2\kappa+m+2)2^{-l}$	$\frac{n}{p^*}(2\kappa+m+2)2^{-l}$	0
δ_r	0	0	0	0	0	$e^{\frac{-2(n'p^*-n)^2}{n'}}$	$e^{\frac{-2(n'p^*-n)^2}{n'}}$	0
δ_p	0	$\kappa\lceil\frac{n}{g}\rceil e^{\frac{-2(\frac{u}{2}-(g-1))^2}{u}}$	$\kappa\lceil\frac{n}{g}\rceil e^{\frac{-2(\frac{u}{2}-(g-1))^2}{u}}$	0	0	0	$(\kappa+m)\lceil\frac{n}{g}\rceil e^{\frac{-2(\frac{u}{2}-(g-1))^2}{u}}$	0



(a) Number of AND gates

(b) Running time

(c) Communication

Figure 2: Overview of sampling protocols' number of AND gates, running time, communication in Shamir-BMR under different $\lambda \in \{128, 256, 512\}$ and $n \in \{2^4, 2^6, 2^8, 2^{10}, 2^{12}\}$.

is the same, i.e., protocols with more AND gates also have longer runtimes and larger communication in real execution. Thus, we mainly use the number of AND gates to present the efficiency in the following several experiments. (2) The AND gates, running time, and communication all increase with the number of samples n and security parameter λ for all the sampling protocols, which is intuitive mainly because, as the n and λ increase, the number of bits .i.e, κ, required to represent the noise samples increase for all the protocols. Moreover, for the protocol with rejection sampling and Ostack-sampling, the higher λ requires larger numbers of Laplace samples n' and CPUSH u are required to reduce the failing rate of generating Laplace samples and biased coins. (3) The ODO-Laplace is the most efficient among the protocols for Laplace noise when $\lambda = 128$, while under the large λ and n, both

Table 3: Sampling protocols' number of random bits under different n and λ

Protocol	Number of Generated Samples n							
	2^4	2^6	2^8	2^{10}	2^{12}	2^{14}	2^{16}	2^{18}
ODO-Laplace	63,360	255,360	1,021,440	4,085,760	16,343,040	65,372,160	261,488,640	1,045,954,560
Ostack-Laplace	13,140	18,090	37,050	102,000	336,360	1,213,770	4,438,140	17,319,240
Ostack-Laplace*	13,140	18,090	37,050	102,000	362,370	1,213,770	4,728,390	18,803,850
DNG-Laplace	**576**	**2,304**	**9,216**	**36,864**	**147,456**	**589,824**	**2,359,296**	**9,437,184**
Transfrom-Laplace	43,776	177,408	718,848	2,912,256	11,796,480	47,800,000	193,000,000	783,000,000
ODO-Gaussian	1,474,200	2,734,920	7,037,280	22,508,820	80,958,960	309,042,000	1,215,536,400	4,852,593,900
Ostack-Gaussian	74,610	98,460	179,640	420,480	1,314,900	4,960,890	17,555,580	68,887,440
DNG-Gaussian	**576**	**2,304**	**9,216**	**36,864**	**147,456**	**589,824**	**2,359,296**	**9,437,184**

the Laplace-Ostack and Laplace-Ostack* can be better. Moreover, since Laplace-Ostack* does not require AND get for RPOP, it is always better than Laplace-Ostack. These results are consistent with those in [18]. The situation is similar for the Gaussian noise, i.e., the Ostack-based protocol can be superior to the ODO-based one when $\lambda \geq 256$. The reason is that although the Ostack-based method theoretically has lower complexity than the ODO-based one, the CPUSH, and RPOP in Ostack-Sampling [18] have a number of constant operations, which domain the AND gates especially when λ is small. (4) Protocols using DNG with KS testing and uniform transformation are significantly less efficient than protocols using bitwise sampling, especially Transfrom-Laplace, which always has the largest circuit size. That is because Transform-Laplace involves expensive evaluation of logarithmic arithmetic.

Random Bits. In addition to AND gates, we also evaluate the number of required random bits from the computing parties for the eight protocols in Table 3. We observe that Table 3 shows the number of required random bits under different λ. We have the following observations. (1) ODO-Gaussian and ODO-Laplace require the largest number of random bits on their corresponding mechanisms since they need $O(l)$ bits to sample a biased coin, while in the Ostack-based protocol, this is reduced to $O(\log l)$ using the oblivious data structure [18]. (2) The random bits used by DNG-based methods DNG-Laplace and DNG-Gaussian are the least and the same since in these two protocols, the inputs from the computing parties are only partial noise with the same length of κ. Therefore, DNG-based methods have an advantage over other protocols in the number of random bits required.

Table 4: Running time (S) of DNG protocols DNG-Laplace, DNG-Gaussian, DNG-Laplace* and DNG-Gaussian*, the latter two are semi-honest DNG protocols without additional Check. We set $\lambda \in \{128, 256, 512\}$.

λ \ Πs	Laplace	Gaussian	Laplace*	Gaussian*
128	1619.07	1599.98	5.92	3.30
256	3239.51	1617.30	6.20	3.32
512	6533.12	3219.33	6.44	3.38

Cost of Malicious Security. We also compare the running time of semi-honest DNG protocols with only Aggregation and the Byzantine-resilient DNG protocols, adding Check in Algorithm 1. We examine both the DNG-Laplace and DNG-Gaussian, setting $\lambda \in \{128, 256, 512\}$. The result is presented in Table 4. We observe that the running time of semi-honest DNG-Laplace* and DNG-Gaussian* are both significantly lower than that of DNG-Laplace and DNG-Gaussian with additional checks because the semi-honest protocol contains only ADD operations while Check requires iteration over all the samples and for each sample, iterating over the CDF table $\langle F \rangle$. Compared to the results in Figure 2, directly aggregating partial noise is markedly more efficient. It takes only seconds to generate 4096 samples even under high security demand. Therefore, in applications requiring only semi-honest security, employing the naive DNG approach that only aggregates partial noise is advisable.

7.3 Trade-off between Efficiency and Security of Bitwise Sampling

Since Bitwise sampling protocols have the least AND gates in all the parameter settings, we conduct further evaluations to give more details by increasing the range of security parameter λ to conduct evaluations. We compare the number of AND gates and random bits of all the bitwise sampling protocols using ODO and Ostack.

Figure 3 shows the number of AND gates. For all protocols, the AND gates increase with λ. Specifically, for the Laplace noise, the circuit size of ODO-based methods is smaller than Ostack-based ones when $n < 2^8$. And when $n \geq 2^8$, the Ostack-Laplace* has fewer AND gates than ODO-Laplace after λ is large enough. Moreover, when $n \geq 2^8$, the Ostack-Laplace is also more efficient than OOD-Laplace. On the other hand, the sampling protocols for Gaussian noise also benefit from using Ostack, i.e., the Ostack-Gaussian can have a smaller number of AND gates when $n \geq 2^8$. We also can clearly see the cross-over points of ODO-based and Ostack-based methods. For the Laplace noise, when λ is very small, i.e., 64, the ODO-Laplace always has the fewest AND gates. Furthermore, when n is larger than n^{10}, the cross-over point for ODO-Laplace and Ostack-Laplace lays in 256 and that for ODO-Laplace and Ostack-Laplace* lays in a smaller number, i.e., 192. As for Gaussian noise, when n is small, the ODO-Gaussian has fewer AND gates than Ostack-Gaussian for all λ, and when n is large enough, say, larger than 2^{10}, the cross-over point of ODO-Gaussian and Ostack-Gaussian is also fixed at $\lambda = 256$. It demonstrates that when λ and n are small, it is more efficient to use OOD-based sampling methods, while in the situation with high demand for security and a number of noise samples, the Ostack-based methods are more suitable. Figure 4 presents the random bits input by computing parties. In line with our expectations,

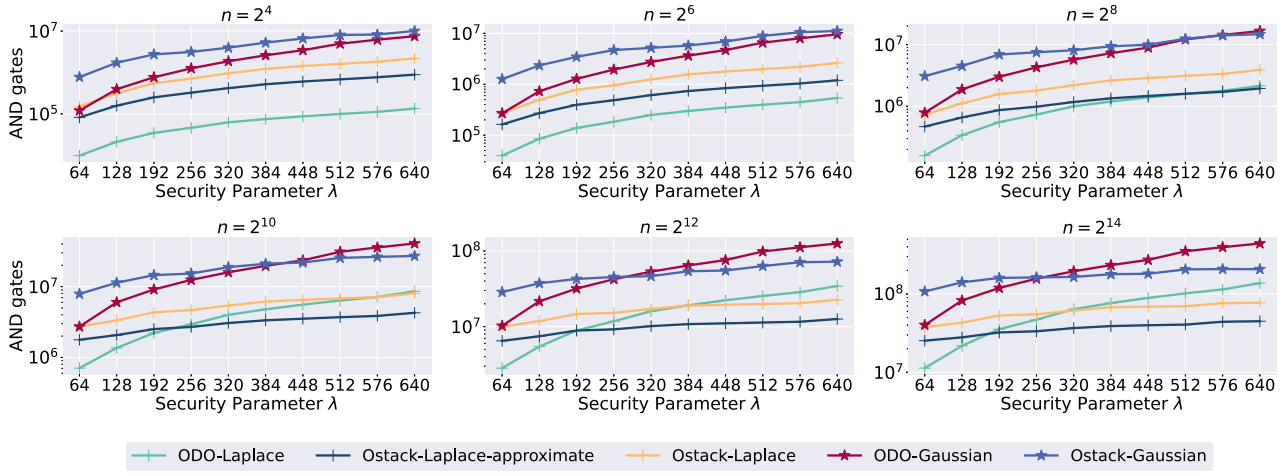


Figure 3: Sampling protocols' trade-off between efficiency and security, measured by security parameter λ and AND gates.

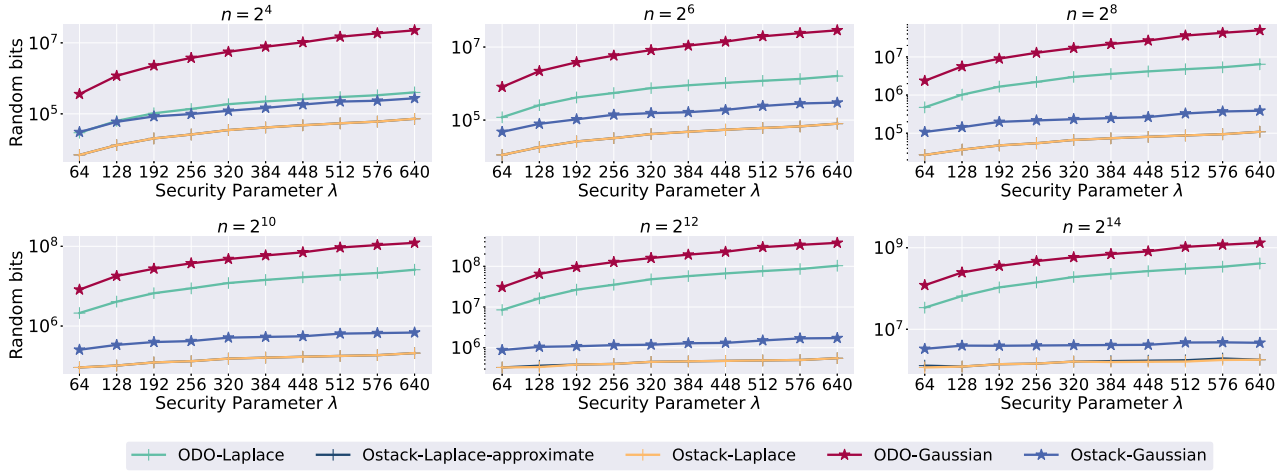


Figure 4: Sampling protocols' trade-off between efficiency and security, measured by security parameter λ and random bits.

the number of random bits required by the ODO-based method is significantly larger than that of the Ostack-based method in all the settings of λ and n for both the Laplace and the Gaussian noise.

7.4 Efficiency under Various Privacy Demands

Recall that at larger variances, in order to get the desired statistical distance δ_t, we need to set a larger N in the truncated distribution, which means that the length κ of binary expansion used to represent the variable needs to be larger. Therefore, the privacy budget ϵ can affect all of the eight sampling protocols. This section explores the impact of privacy budget ϵ on sampling efficiency.

Figure 5 presents the number of AND gates in eight sampling protocols with $\epsilon \in \{0.001, 0.01, 0.1, 1, 10\}$ and $\lambda \in \{64, 128, 192, 256\}$. We have the following observations. 1) Circuit size increases for all of the eight protocols as ϵ decreases, which is typical for all the security parameters λ. 2) The circuit size of Transform-Laplace has the smallest change with ϵ and stays at a significantly high level. This is because the decimal part domains the length of the fixed-point number used for logarithm arithmetic, and changing the integer part has little impact on the number of AND gates. 3) Moreover, the circuit size of DNG-based protocols has the largest change with ϵ since the KS-test requires scanning a table of length N to check each sample, resulting in a complexity of $O(N)$.

7.5 Varying the Number of Computing Parties

In this section, we set different numbers of computing parties. We measure the running time and the communication of seven protocols except Transform-Laplace. The results are shown in Figure 6. We use Yao's protocol to run our code in the two-party setting, and for other m, the Shamir-BMR is used. We observe that both running time and communication increase linearly with the increase of computing party number m. This demonstrates that the running

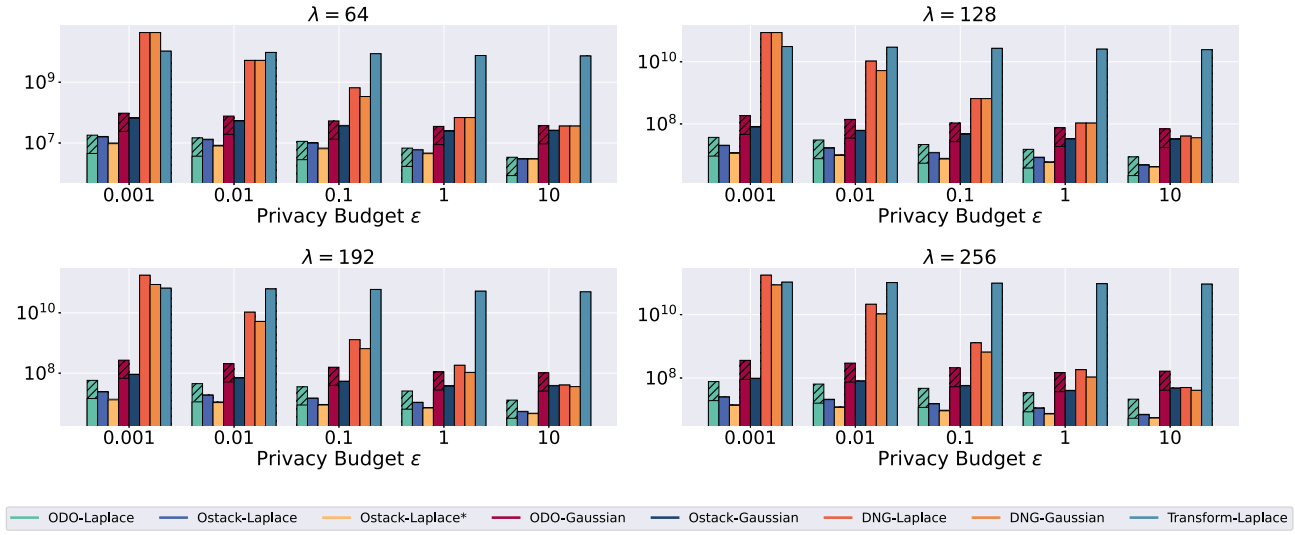


Figure 5: Comparison of sampling protocols' number of AND gates + random bits under different ϵ and λ.

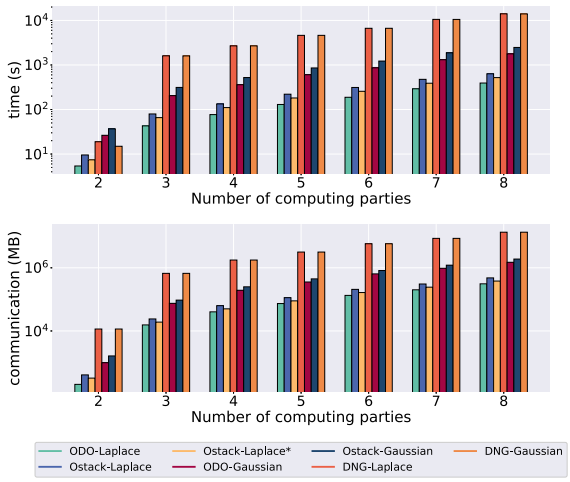


Figure 6: The time and communication of protocols under different numbers of parties, where Yao's protocol is used for a two-party setting, and Shamir-BMR is used for others.

time of secure sampling protocols is still acceptable in a practical setting with multiple computing parties. Moreover, Yao's protocol has significantly lower communication and running time since all the protocols can be finished within 100 seconds. Since the main computations in sampling protocols are AND operations of secret bits, the

7.6 Utility of Distributed DP: Case Study

In this section, we validate the utilities of sampling protocols in our benchmark and compare them with center DP, local DP, and shuffle DP under different λ and ϵ.

7.6.1 Setup. We query the differentially private frequency on the Kosarak dataset[2] . Kosarak is the click record of the Hungarian news website, which contains about 8 million clicks for $n = 41,270$ different web pages. Specifically, we estimate each web page's clicked times. Then, the noise generated in the *central model* (continuous Laplace and Gaussian) and *distributed model* (discrete Laplace and Gaussian) were added to numbers' frequency $f(D)$. We also use OLH [55] and its shuffle DP version, SOLH [57], to construct responses on each number (representing single users) to satisfy LDP and shuffle DP. We use MSE to measure and compare the utility of noisy frequency. The Mean Squared Errors is defined as: MSE = $\frac{1}{n}\sum_{i\in[n]}[f(D)-y]^2$. We also show the results in Mean Absolute Error (MAE) and Relative Error (RE) in Appendix E of the technical report [33].

7.6.2 Comparison Result. Table 5 shows the MSE in $\epsilon \in [0.1, 0.5]$ of eight sampling protocols as well as two baseline methods, CDP-Laplace and CDP-Gaussian sampling continuous noise on one server. The results on LDP and shuffle DP are also reported. First, we can see that in all ϵ, LDP and shuffle DP have significantly higher MSEs than those of DDP, although they all do not assume a trusted server. Such a large utility gap is due to the random report of the 8 million click records acting as users. Third, the utilities of DDP protocols are close to those of CDP protocols for both Laplace and Gaussian mechanisms, which matches the expectation for the utility of secure sampling protocols. Second, the utility (MSE) of continuous CDP-Laplace and Laplace in the DDP model is similar for all ϵ. The only exception is that the Ostack-Laplace* has a larger MSE than other methods because it uses an approximated ϵ, which is the largest number in the form of $2^{-i}\ln 2$, ($i \in \mathbb{N}$) smaller than the required one. We also fix $\epsilon = 0.1$ and set $\lambda = \{1, 2, 4, 8, 16\}$ to measure only the MSE for DDP and CDP protocols in Table 6. We observe that the utilities of DDP Laplace protocols under different λ

[2]Kosarak. http://fimi.ua.ac.be/data

Table 5: The MSE of frequency on Kosarak with protocols Π_s under different privacy budgets ϵ and security parameter $\lambda = 128$.

ϵ \ Πs	CDP Laplace	CDP Gaussian	ODO Laplace	Ostack Laplace	Ostack Laplace*	DNG Laplace	ODO Gaussian	Ostack Gaussian	DNG Gaussian	LDP OLH	Shuffle SOLH
0.1	200.03	2408.16	203.49	204.06	271.65	203.15	2380.93	2380.93	2385.65	1.45×10^{13}	6.51×10^{8}
0.2	48.53	584.16	49.64	51.08	66.36	49.72	576.65	576.65	573.34	1.30×10^{13}	4.46×10^{7}
0.3	21.75	266.42	21.26	22.88	61.43	21.90	272.52	272.52	279.68	1.16×10^{13}	1.24×10^{7}
0.4	12.36	147.62	11.77	12.35	17.47	12.24	146.48	146.48	151.42	1.04×10^{13}	7.12×10^{6}
0.5	7.91	95.16	7.31	7.85	16.13	8.16	94.89	94.89	95.89	9.16×10^{12}	5.72×10^{6}

Table 6: The MSE of frequency on Kosarak with protocols Π_s under privacy budget $\epsilon = 0.1$ and different security parameter λ.

λ \ Πs	CDP Laplace	ODO Laplace	Ostack Laplace	Ostack Laplace*	DNG Laplace	CDP Gaussian	ODO Gaussian	Ostack Gaussian	DNG Gaussian
2		195.24	178.07	198.85	204.37		2355.68	2355.68	2323.40
4		191.42	192.88	198.85	205.96		2323.81	2323.81	2350.15
8	200.03	204.35	198.85	265.94	194.98	2408.16	2332.28	2332.28	2350.15
16		203.06	190.69	266.95	217.22		2312.47	2312.47	2286.67
32		189.71	193.03	260.05	196.97		2369.28	2369.28	2401.93

are similar. By comparing the experimental results of CDP-Laplace, CDP-Gaussian, and the secure sampling protocols, we can see that the statistical distance caused by MPC in Section 6.2 does not introduce additional errors to the answers of the counting query (even when λ is very small).

7.6.3 Utility under Colluded Adversaries and Zero Noise Attack. As discussed in Section 4.1, when the parties providing partial noise in DNG collude with each other, by subtracting their noise samples, they may reduce the noise in the final output, which breaks the claimed DP protection of DDP protocols. The impact is identical to the zero noise attacks in Section 4.2, where these parties always provide 0 as their partial noise. To this end, we must ask each party to provide noise with a larger variance.

Assuming different ratios of collusion $\alpha \in [0\%, 90\%]$ ($\alpha = 0\%$ also represents the utility in CDP Gaussian), we use Equation (4) to recalculate the variance of partial Gaussian noise in DNG-Gaussian. The generation is repeated on $\epsilon \in [0.2, 1.0]$. We add the generated noise to frequencies on Kosarak and show the MSE in Figure 7. We observe that the impact of colluded attacks is large. Under 90% collusion, the MSE is nearly 100× compared to the non-collusion one (also the utility of CDP Gaussian). Even under the honest majority setting (1/3 collusion), the MSE also increases 5 times. Therefore, for DNG, additional post-processing must be applied to the results (for example, securely shuffling [20] the noisy results before revealing them).

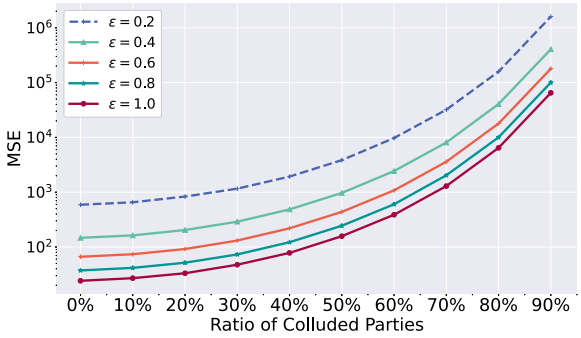


Figure 7: The MSE of frequency on Kosarak using noise from DNG-Gaussian, assuming ratios of collusion $\alpha \in [0\%, 90\%]$ and privacy budgets $\epsilon \in [0.2, 1.0]$.

8 Takeaways

This section summarizes the key findings and guidelines considering different factors from our evaluations.

Threat Model. We first discuss the trade-off on threat models.

- **Semi-honest vs Malicious.** To realize malicious security, the bitwise sampling-based methods are the most effective, while for the semi-honest setting, DNG-based protocols using only `Aggregation` are the best. That is because the semi-honest DNG-Gaussian involves only `ADD` operations to sum the partial noises. As for the semi-honest DNG-Laplace, it requires additional operations to convert Geometric samples to discrete Laplace samples [59]. Thus, the number of AND gates for each sample in these two distributions is $O(1)$. For malicious settings, the minimum number of AND gates can be found in Ostack-based bitwise sampling for Laplace distribution, which is $O(\log^2(\lambda + \log n))$ [18].
- **Number of Colluded Parties.** The fraction of corrupt parties affects the utility of DNG protocols for the Gaussian mechanism since each party should sample Gaussian with additional variances $\hat{\sigma}^2 = \sigma^2/(1-\alpha)$, where σ^2 is the original variance of partial noise and α is the proportion of colluded parties (discussed in Section 4.1). Thus, the final Gaussian noise also has a variance of $\frac{1}{1-\alpha}$ times compared to the situation without colluded parties. Note that the bitwise sampling protocols can maintain the same utility because the XOR results of input bits are not revealed.
- **Data Poisoning Attack.** The malicious behavior also exists in the input parties. They can change the local data to skew the final aggregation results in MPC before sampling DDP noise, which is out of the scope of sampling protocols. The data poisoning attack

can not be completely addressed. Instead, a promising way is leveraging zero-knowledge proof depending on the specific use case, which ensures the validity of inputs [7, 8, 12, 13, 52], thus limiting the malicious parties' manipulation of the results.

Deployment Model. We then discuss the deployment models.

- **Statistical Security Parameter λ.** Now we consider $n > 2^{10}$. To generate Laplace samples with $\lambda < 256$, we suggest using ODO-Laplace. Otherwise, Ostack-Laplace is more efficient. For the user accepting approximation, with $\lambda > 196$, a more efficient version, i.e., Ostack-Laplapce* can be chosen. However, Ostack-Laplapce* essentially uses smaller ϵ, which has worse utility. As for generating Gaussian samples, with $\lambda < 256$, the ODO-Gaussian is suggested, and Ostack-Gaussian for $\lambda \geq 256$. The suggestion above is from results in Figure 3. Note that one can use the predicate function in [18] to improve the Ostack-based methods. However, the predicate needs to be hard-coded previously and is hard to re-implement when ϵ changes.
- **Number of Samples n.** In general, the computation overhead and number of input bits are linear in the number of required samples. For example, it needs about 0.2 seconds to generate one discrete Gaussian sample in the malicious setting (Ostack-Gaussian). In machine learning, hundreds of thousands of seconds are needed to generate millions of noise samples, which is not practical. Thus, DNG in the semi-honest setting is usually considered [53]. When generating a small number of samples (when $n \leq 2^8$), using ODO sampling is more efficient. However, when implementing mechanisms with a large demand of Laplace or Gaussian samples like the Noisy Max Mechanism [45] (see Appendix A in [33]), it is more suitable to use Ostack-sampling.
- **Number of Parties m.** From the experimental results, the running time is linear in the number of computing parties m. However, in practice, the computing parties are usually servers with high computing ability, and the number is usually small [12, 59] (two or three non-colluded servers). As for the input parties, they only need to secret-share their local data and do not participate in the computation. For n input parties splitting shares to m computing parties, the communication complexity is $O(nm)$.
- **Privacy Budget ϵ.** The privacy budget ϵ can also affect the efficiency of protocols. Specifically, smaller ϵ means larger binary expansion length κ of generated samples is required to achieve the truncated statistical distance δ_t. As a result, the number of AND gates increases (See Figure 5). However, the overall ranking of AND gates is consistent across all protocols, so the choice of protocols is usually determined by λ and n rather than ϵ.

Utility. The utility of the counting query in DDP is close to that of CDP if the mechanism is realized by discrete Laplace and Gaussian noise, even when setting different security parameters λ and privacy budget ϵ (results in Table 5 and Table 6).

9 Conclusion

This paper presents a benchmark to study the efficiency and utility of various secure sampling protocols to realize differential privacy in the *distributed model*. We first review the existing sampling protocols and present a taxonomy for them. Then, we give further analysis to discuss the security of distributed noise generation and present the relationship of various parameters in the bitwise sampling methods. We conducted experiments and found that the bitwise bampling-based methods are most efficient under various settings since they have the least number of AND gates to evaluate in the binary circuits. Moreover, we also estimate the utility of the DDP counting query on real-world datasets.

Acknowledgement

We thank all anonymous reviewer construction feedback. Yucheng's work was done while he was an undergrad at Sichuan University and remote interning at the University of Virginia. The work was partially supported by CNS-2220433.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

[2] Abdelrahaman Aly, Karl Cong, Daniele Cozzo, Marcel Keller, Emmanuela Orsini, Dragos Rotaru, Oliver Scherer, Peter Scholl, Nigel P Smart, Titouan Tanguy, et al. 2021. SCALE–MAMBA v1. 14: Documentation. *Documentation. pdf* (2021).

[3] KOLMOGOROV AN. 1933. Sulla determinazione empirica di una legge didistribuzione. *Giorn Dell'inst Ital Degli Att* 4 (1933), 89–91.

[4] Balamurugan Anandan and Chris Clifton. 2015. Laplace noise generation for two-party computational differential privacy. In *2015 13th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 54–61.

[5] Victor Balcer and Albert Cheu. 2019. Separating local & shuffled differential privacy via histograms. *arXiv preprint arXiv:1911.06879* (2019).

[6] Borja Balle and Yu-Xiang Wang. 2018. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*. PMLR, 394–403.

[7] James Bell, Adrià Gascón, Tancrède Lepoint, Baiyu Li, Sarah Meiklejohn, Mariana Raykova, and Cathie Yun. 2023. {ACORN}: input validation for secure aggregation. In *32nd USENIX Security Symposium (USENIX Security 23)*. 4805–4822.

[8] James Henry Bell, Kallista A Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. 2020. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 1253–1269.

[9] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. 2011. Semi-homomorphic encryption and multiparty computation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 169–188.

[10] Vance W Berger and YanYan Zhou. 2014. Kolmogorov–smirnov test: Overview. *Wiley statsref: Statistics reference online* (2014).

[11] Ari Biswas and Graham Cormode. 2023. Interactive proofs for differentially private counting. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 1919–1933.

[12] Jonas Boehler and Florian Kerschbaum. 2022. Secure sublinear time differentially private median computation. US Patent 11,238,167.

[13] Jonas Böhler and Florian Kerschbaum. 2020. Secure multi-party computation of differentially private median. In *29th USENIX Security Symposium (USENIX Security 20)*. 2147–2164.

[14] Jonas Böhler and Florian Kerschbaum. 2021. Secure multi-party computation of differentially private heavy hitters. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2361–2377.

[15] Elette Boyle, Niv Gilboa, and Yuval Ishai. 2019. Secure computation with preprocessing via function secret sharing. In *Theory of Cryptography: 17th International Conference, TCC 2019, Nuremberg, Germany, December 1–5, 2019, Proceedings, Part I 17*. Springer, 341–371.

[16] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. 2018. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE symposium on security and privacy (SP)*. IEEE, 315–334.

[17] Clément L Canonne, Gautam Kamath, and Thomas Steinke. 2020. The discrete gaussian for differential privacy. *Advances in Neural Information Processing Systems* 33 (2020), 15676–15688.

[18] Jeffrey Champion, Abhi Shelat, and Jonathan Ullman. 2019. Securely sampling biased coins with applications to differential privacy. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 603–614.

[19] T-H Hubert Chan, Elaine Shi, and Dawn Song. 2011. Private and continual release of statistics. *ACM Transactions on Information and System Security (TISSEC)* 14, 3 (2011), 1–24.

[20] Melissa Chase, Esha Ghosh, and Oxana Poburinnaya. 2020. Secret-shared shuffle. In *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part III 26*. Springer, 342–372.

[21] Albert Cheu. 2021. Differential privacy in the shuffle model: A survey of separations. *arXiv preprint arXiv:2107.11839* (2021).

[22] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. 2019. Distributed differential privacy via shuffling. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I 38*. Springer, 375–403.

[23] Graham Cormode, Somesh Jha, Tejas Kulkarni, Ninghui Li, Divesh Srivastava, and Tianhao Wang. 2018. Privacy at scale: Local differential privacy in practice. In *Proceedings of the 2018 International Conference on Management of Data*. 1655–1658.

[24] Henry Corrigan-Gibbs and Dan Boneh. 2017. Prio: Private, robust, and scalable computation of aggregate statistics. In *14th USENIX symposium on networked systems design and implementation (NSDI 17)*. 259–282.

[25] Anders Dalskov, Daniel Escudero, and Marcel Keller. 2021. Fantastic four:{Honest-Majority}{Four-Party} secure computation with malicious security. In *30th USENIX Security Symposium (USENIX Security 21)*. 2183–2200.

[26] Minxin Du, Xiang Yue, Sherman SM Chow, Tianhao Wang, Chenyu Huang, and Huan Sun. 2023. DP-Forward: Fine-tuning and inference on language models with differential privacy in forward pass. *arXiv preprint arXiv:2309.06746* (2023).

[27] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology-EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28-June 1, 2006. Proceedings 25*. Springer, 486–503.

[28] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*. Springer, 265–284.

[29] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[30] Fabienne Eigner, Aniket Kate, Matteo Maffei, Francesca Pampaloni, and Ivan Pryvalov. 2014. Differentially private data aggregation with optimal utility. In *Proceedings of the 30th Annual Computer Security Applications Conference*. 316–325.

[31] David Evans, Vladimir Kolesnikov, Mike Rosulek, et al. 2018. A pragmatic introduction to secure multi-party computation. *Foundations and Trends® in Privacy and Security* 2, 2-3 (2018), 70–246.

[32] Brett Hemenway Falk, Rohit Nema, and Rafail Ostrovsky. 2023. Linear-time 2-party secure merge from additively homomorphic encryption. *J. Comput. System Sci.* 137 (2023), 37–49.

[33] Yucheng Fu and Tianhao Wang. 2024. Benchmarking Secure Sampling Protocols for Differential Privacy. arXiv:2409.10667 [cs.CR] https://arxiv.org/abs/2409.10667

[34] Oded Goldreich. 2009. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press.

[35] Slawomir Goryczka and Li Xiong. 2015. A comprehensive comparison of multiparty secure additions with differential privacy. *IEEE transactions on dependable and secure computing* 14, 5 (2015), 463–477.

[36] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. 2009. Boosting the accuracy of differentially-private histograms through consistency. *arXiv preprint arXiv:0904.0942* (2009).

[37] Mikko Heikkilä, Eemil Lagerspetz, Samuel Kaski, Kana Shimizu, Sasu Tarkoma, and Antti Honkela. 2017. Differentially private bayesian learning on distributed data. *Advances in neural information processing systems* 30 (2017).

[38] Thomas Humphries, Rasoul Akhavan Mahdavi, Shannon Veitch, and Florian Kerschbaum. 2022. Selective MPC: Distributed Computation of Differentially Private Key-Value Statistics. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 1459–1472.

[39] Peter Kairouz, Ziyu Liu, and Thomas Steinke. 2021. The distributed discrete gaussian mechanism for federated learning with secure aggregation. In *International Conference on Machine Learning*. PMLR, 5201–5212.

[40] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. 2008. What Can We Learn Privately?. In *FOCS*. IEEE Computer Society, 531–540.

[41] Dana Keeler, Chelsea Komlo, Emily Lepert, Shannon Veitch, and Xi He. 2023. DPrio: Efficient Differential Privacy with High Utility for Prio. *Proceedings on Privacy Enhancing Technologies* 3 (2023).

[42] Marcel Keller. 2020. MP-SPDZ: A versatile framework for multi-party computation. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*. 1575–1590.

[43] Marcel Keller and Avishay Yanai. 2018. Efficient maliciously secure multiparty computation for RAM. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 91–124.

[44] Yehuda Lindell and Benny Pinkas. 2009. A proof of security of Yao's protocol for two-party computation. *Journal of cryptology* 22 (2009), 161–188.

[45] Frank McSherry and Kunal Talwar. 2007. Mechanism Design via Differential Privacy. In *FOCS*. 94–103.

[46] Silvio Micali, Oded Goldreich, and Avi Wigderson. 1987. How to play any mental game. In *Proceedings of the Nineteenth ACM Symp. on Theory of Computing, STOC*. ACM New York, NY, USA, 218–229.

[47] Ilya Mironov. 2017. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*. IEEE, 263–275.

[48] Sikha Pentyala, Davis Railsback, Ricardo Maia, Rafael Dowsley, David Melanson, Anderson Nascimento, and Martine De Cock. 2022. Training differentially private models with secure multiparty computation. *arXiv preprint arXiv:2202.02625* (2022).

[49] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2013. Understanding hierarchical methods for differentially private histograms. *Proceedings of the VLDB Endowment* 6, 14 (2013), 1954–1965.

[50] Xuebin Ren, Liang Shi, Weiren Yu, Shusen Yang, Cong Zhao, and Zongben Xu. 2022. LDP-IDS: Local differential privacy for infinite data streams. In *Proceedings of the 2022 international conference on management of data*. 1064–1077.

[51] Edo Roth, Karan Newatia, Yiping Ma, Ke Zhong, Sebastian Angel, and Andreas Haeberlen. 2021. Mycelium: Large-scale distributed graph queries with differential privacy. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*. 327–343.

[52] Amrita Roy Chowdhury, Chuan Guo, Somesh Jha, and Laurens van der Maaten. 2022. Eiffel: Ensuring integrity for federated learning. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 2535–2549.

[53] W. Ruan, M. Xu, W. Fang, L. Wang, L. Wang, and W. Han. 2023. Private, Efficient, and Accurate: Protecting Models Trained by Multi-party Learning with Differential Privacy. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 1926–1943. https://doi.org/10.1109/SP46215.2023.10179422

[54] Yuchao Tao, Ryan McKenna, Michael Hay, Ashwin Machanavajjhala, and Gerome Miklau. 2021. Benchmarking differentially private synthetic data generation algorithms. *arXiv preprint arXiv:2112.09238* (2021).

[55] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. 2017. Locally differentially private protocols for frequency estimation. In *26th USENIX Security Symposium (USENIX Security 17)*. 729–745.

[56] Tianhao Wang, Joann Qiongna Chen, Zhikun Zhang, Dong Su, Yueqiang Cheng, Zhou Li, Ninghui Li, and Somesh Jha. 2021. Continuous release of data streams under both centralized and local differential privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 1237–1253.

[57] Tianhao Wang, Bolin Ding, Min Xu, Zhicong Huang, Cheng Hong, Jingren Zhou, Ninghui Li, and Somesh Jha. 2020. Improving utility and security of the shuffler-based differential privacy. *Proceedings of the VLDB Endowment* 13, 13 (2020), 3545–3558.

[58] Tianhao Wang, Milan Lopuhaä-Zwakenberg, Zitao Li, Boris Skoric, and Ninghui Li. 2019. Locally differentially private frequency estimation with consistency. *arXiv preprint arXiv:1905.08320* (2019).

[59] Chengkun Wei, Ruijing Yu, Yuan Fan, Wenzhi Chen, and Tianhao Wang. 2023. Securely Sampling Discrete Gaussian Noise for Multi-Party Differential Privacy. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*.

[60] Chengkun Wei, Minghu Zhao, Zhikun Zhang, Min Chen, Wenlong Meng, Bo Liu, Yuan Fan, and Wenzhi Chen. 2023. DPMLBench: Holistic Evaluation of Differentially Private Machine Learning. *arXiv preprint arXiv:2305.05900* (2023).

[61] Zihang Xiang, Tianhao Wang, Wanyu Lin, and Di Wang. 2023. Practical Differentially Private and Byzantine-resilient Federated Learning. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–26.

[62] Xingxing Xiong, Shubo Liu, Dan Li, Zhaohui Cai, and Xiaoguang Niu. 2020. A comprehensive survey on local differential privacy. *Security and Communication Networks* 2020 (2020), 1–29.

[63] Andrew C Yao. 1982. Protocols for secure computations. In *23rd annual symposium on foundations of computer science (sfcs 1982)*. IEEE, 160–164.

[64] Zhikun Zhang, Tianhao Wang, Ninghui Li, Shibo He, and Jiming Chen. 2018. CALM: Consistent adaptive local marginal for marginal release under local differential privacy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 212–229.

[65] Zhikun Zhang, Tianhao Wang, Ninghui Li, Jean Honorio, Michael Backes, Shibo He, Jiming Chen, and Yang Zhang. 2021. PrivSyn: Differentially Private Data Synthesis. In *30th USENIX Security Symposium (USENIX Security 21)*. 929–946.