Lightweight Two-Party Secure Sampling Protocol for Differential Privacy

Masanobu Kii
NTT Social Information Laboratories
Tokyo, Japan
masanobu.kii@ntt.com

Atsunori Ichikawa
NTT Social Information Laboratories
Tokyo, Japan

Takayuki Miura
NTT Social Information Laboratories
Tokyo, Japan

Abstract

Secure sampling is a secure multiparty computation protocol that allows a receiver to sample random numbers from a specified non-uniform distribution. It is a fundamental tool for privacy-preserving analysis since adding controlled noise is the most basic and frequently used method to achieve differential privacy. The well-known approaches to constructing a two-party secure sampling protocol are transforming uniform random values into non-uniform ones by computations (e.g., logarithm or binary circuits) or table-lookup. However, they require a large computational or communication cost to achieve a strong differential privacy guarantee. This work addresses this problem with our novel lightweight two-party secure sampling protocol. Our protocol consists of random table-lookup from a small table with the 1-out of-n oblivious transfer and only additions. Furthermore, we provide algorithms for making a table to achieve differential privacy. Our method can reduce the communication cost for $(1.0, 2^{-40})$-differential privacy from 183GB (naïve construction) to 7.4MB.

Keywords

multiparty computation, differential privacy, random number generation, secure sampling, private sampling

1 Introduction

Data collaboration is one of the major goals of privacy-preserving technologies. In typical data collaboration, all participants contribute their data, gather it, analyze it, and all receive the results.

In this setting, there are two types of privacy risks for any individual providing information to participants in the data collaboration: leakage during the collaboration, i.e., one may observe another's data directly, and leakage after the collaboration, i.e., one may infer another's data from the analysis result. There are two known techniques for solving this problem: secure multiparty computation (SMPC) and differential privacy (DP). SMPC enables joint data analysis while protecting individual privacy, and DP can prevent inferring inputs from the outputs.

The additive mechanism is a fundamental and effective tool to achieve DP. It adds a noise that follows a non-uniform distribution to an exact analysis result. This approach has been well-studied and well-used in the literature. Thanks to its results, we can achieve DP with small noise in many cases.

When we use the additive mechanism, everybody is prohibited from learning both the noisy value and any information (even very limited information) about the added noise. This is because the original value would be inferred from them acutely more than expected. If this were to happen, the intended privacy guarantee would not be achieved. In our setting, the noisy output is shared with every participant of the SMPC protocol. Therefore, we must sample noise without disclosing any information about the noise to anyone. Secure sampling (private sampling) is an SMPC protocol to realize this.

Prior Works. There are three approaches in prior works of SMPC protocols realizing the additive mechanism, but each has its shortcomings. The as-in-plaintext approach (or computation-centric approach) basically emulates the noise generation algorithm used in plaintext computation. In general, this approach does heavy computations in SMPC. On the other hand, the random table-lookup approach skips the computations by table-lookup. However, it communicates a huge table, even for mild privacy assurance. In a distributed sampling approach, SMPC participants sample the noise in plaintext, and sum it in SMPC. Since each participant knows a part of the generated noise, it requires much larger noise than the minimum. In summary, the prior methods incur a heavy computation cost, a heavy communication cost, or a larger error value.

Our Aim. This paper aims to provide a secure sampling method with a small computational cost, communication cost, and noise magnitude (L^1 error). Moreover, we aim to prove that our method can achieve DP guarantees at an arbitrary level (privacy parameters). We focus on achieving rigorous DP of integer-valued (or valued in finite-precision) queries because high-precision calculations under SMPC are neither possible nor required at present.

Our Approach. To this end, we extend the random table-lookup approach. Our approach picks out some numbers from a table randomly and transforms them into a noise following a given distribution with a little computation. The original random table-lookup method only requires a small amount of computation and amortized communication. The huge communication cost could be reduced with just a little computation.

Our Contribution. In this paper, we provide a lightweight two-party secure sampling protocol. Its computation in the online phase consists of some random table-lookup with a small table and a few additions. To achieve $(1.0, 2^{-40})$-DP, our method only needs

Proceedings on Privacy Enhancing Technologies 2025(1), 23–36
© 2025 Copyright held by the owner/author(s).
https://doi.org/10.56553/popets-2025-0003

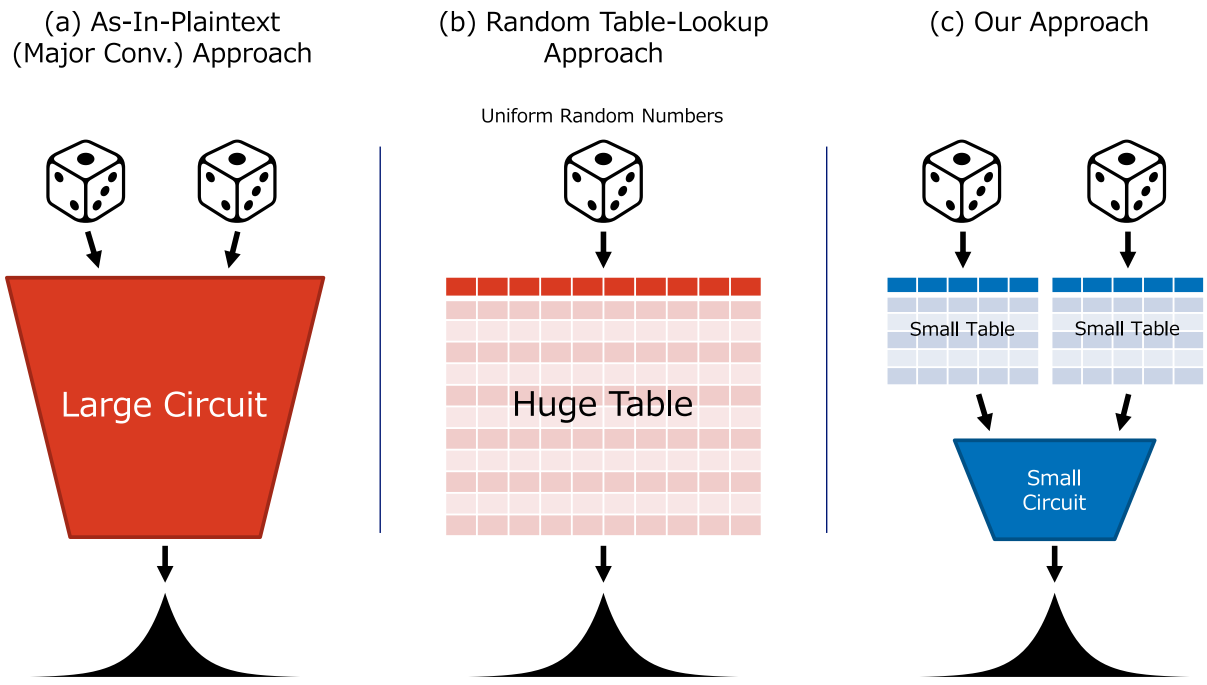

Figure 1: The as-in-plaintext approach (a) transforms some uniform random numbers with arithmetic or binary circuits, and the random table-lookup approach (b) transforms a single uniform random number by table-lookup. Our approach (c) transforms some uniform random numbers by table-lookup and then transforms them to a single noise by a small circuit.

the communication cost of just 7.4 MB, while a naïve construction (Section 5.2) needs a communication cost of 183 GB. Furthermore, we prove that our method can give DP of an arbitrary level with a "sufficiently large" table and verify that the tables are small in experiments. Our method does not require transcendental functions to provide DP. However, the L^1 error of the noise sampled by our method is larger than those of existing methods, especially the distributed sampling method. We believe that the reason for this is the requirement to use the small table in our method. This paper does not look for the optimal distribution (table) for the proposed method, so whether this requirement is an intrinsic limitation of the proposed method is not known. Our proposed approach has the potential for further development, and we believe that the L^1 error can be reduced in future works.

Structure of This Paper. After reviewing some common notations and concepts in Section 3, we formalize and see details of our problem in Section 4. In Section 5, we provide algorithms to generate tables for naïve sampling method with random table-lookup only. These algorithms tell us details of problems in the random table-lookup method and help us understand our proposed method. After that, we propose solutions in Sections 6 and 7. We propose two algorithms to generate tables for our approach in Section 6, and a two-party protocol for secure sampling in Section 7. To show how our method works in practice, we provide an experimentally compare it with several methods in Section 8. We conclude our problem and solutions in Section 9.

2 Related Works

This section reviews prior works related to this study. There are three approaches in prior works of SMPC protocols realizing additive mechanisms.

As-in-plaintext approach. Since the pioneer paper [10], the majority of prior studies on secure sampling [7, 10, 12, 13, 19, 30] is based on the standard sampling method in plaintext. This as-in-plaintext approach transforms uniform random numbers with transcendental functions (e.g., the exponential or logarithm functions) or binary circuits in SMPC. It can sample noise of minimum error (this implies the maximum utility). However, its computation cost is generally very heavy. Among prior studies, methods on DP for real-valued queries [12, 30] must approximate the transcendental function. Moreover, previous studies have not analyzed the effect of this approximation on the DP guarantee. Some studies on DP of integers and finite-precision numbers [13, 19] address this issue. In contrast, this paper provides a two-party SMPC protocol to achieve DP of integer-valued queries.

We refer to a protocol in Keller et al. [19] as the current state-of-the-art in this approach. This protocol computes a binary circuit to generate noise following the discrete Laplace distribution in Yao's garbled circuit scheme. The circuit has 1.86×10^7 (18.7 million) AND gates. According to their experimental results, their method takes 993 milliseconds and 429MB of communication per noise generation. In our experiments, we compared our method with theirs, and found that our method reduces computation costs by combining random table-lookup with a few arithmetic computations.

Distributed sampling approach. Another line of work in secure sampling [1, 8, 9, 17, 27, 28] employs the distributed sampling approach. In this approach, SMPC participants sample random noises in plaintext and sum them in SMPC. This approach gives an information-theoretic DP guarantee, while another gives a computational DP guarantee. However, the resulting error is much larger than the minimum. Since they know the noise generated by itself, the individual noise sampled by each participant must be large enough to achieve DP. For example, Shi et al. [27] provide an aggregation protocol whose output is the sum of the exact aggregate and the two Laplace noises. Our approach also applies i.i.d. noises for a single value, but since the individual noises are hidden, our approach can achieve a smaller error than the methods of the distributed sampling approach.

Random table-lookup approach. Froelicher et al. [14] and subsequent researchers skipped complex calculations with table-lookup. They sample a noise by looking up an encrypted table with a uniformly random index. This approach is simple and easy to implement. However, it requires huge communication cost. In fact, their method requires sending the whole encrypted table for each sample, and Froelicher et al. [14] proved that the number of elements in a table for (ε, δ)-DP is $1/\delta$. It is usually 10^5 to 10^{10}. The communication amount of our method is twice that of a plaintext table, and our method makes the table very small.

3 Preliminary

We introduce definitions of DP and integer-value random variables. First, we define (ε, δ)-DP with hockey–stick divergence. This definition gives the explicit way to compute δ. After that, we introduce the concept of probability generating function to deal with the sum of random variables and some discrete probability distributions and their N-divisibility.

3.1 Differential Privacy

For a discrete probability distribution D in $\mathbb{Z}$, we denote the probability mass functions (PMFs) of D by $P_D(k)$ $(k \in \mathbb{Z})$. We also denote the PMF of a random variable X by $P_X(k)$.

Definition 3.1 (Hockey–Stick divergence between Integer-Valued Random Variables). *Let X, Y be two integer-valued random variables. The Hockey–Stick divergence between X and Y with parameter e^ε is defined as*

$$D_{e^\varepsilon}(X\|Y) \coloneqq \sum_{k\in\mathbb{Z}} \max\left\{0, P_X(k) - e^\varepsilon P_Y(k)\right\}.$$

Note that it can be seen as f-divergence for $f(t) = \max\{0, t - e^\varepsilon\}$.

If two datasets differ by only a single element, we call them neighboring datasets and denote $X \sim X'$.

Definition 3.2 (Differential Privacy (DP) of Integer-Valued Mechanism [2, 3, 11, 23]). *Consider a randomized function $\mathcal{M}\colon \mathbb{D} \to \mathbb{Z}$. Let $\varepsilon(\geq 0), \delta(\in [0,1))$ be non-negative real values and n be a positive integer. We say that $\mathcal{M}$ achieves (ε, δ)-differential privacy (DP) if*

$$\bar{\delta}_{\mathcal{M}}(\varepsilon) \coloneqq \sup_{X,X'} D_{e^\varepsilon}(\mathcal{M}(X)\|\mathcal{M}(X')) \leq \delta. \tag{1}$$

Here, $\sup_{X,X'}$ *is a supremum over all pairs of neighboring datasets $X, X' \in \mathbb{D}^n$.*

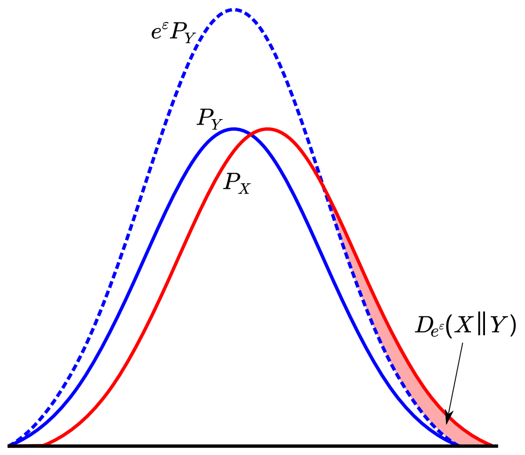

Figure 2: An illustration of the Hockey–Stick divergence $D_\gamma(P\|Q)$ (cf. [23]). The privacy parameter δ is calculated via the maximum value of the Hockey–Stick divergence.

This function $\delta_{\mathcal{M}}$ is called "privacy profile" in several studies (e.g., [2]). The parameters ε, δ are called privacy budgets or privacy loss parameters. The smaller the values of the privacy budgets, the better the privacy guarantee for each individual. For a better privacy guarantee, one of the parameters δ must be a value close to zero, e.g. 10^{-5} to 10^{-10}.

Rigorously, we should consider computational DP [4, 22, 24], an analog of DP for computationally bounded adversaries. This is because our protocol is not information-theoretically secure but is computationally secure. Since any computationally secure SMPC protocol realizing DP mechanism achieves computational DP, we will not deal with it in detail.

The following theorem gives a concrete way to achieve differential privacy. At the same time, it gives conditions that must be satisfied by the noise distribution used. Most of the theory in this paper is devoted to analyzing sufficient conditions for these conditions.

Theorem 3.3. *Let $q\colon \mathbb{D} \to \mathbb{Z}$ be an integer-valued query (function) on a database X, and let D be a discrete probability distribution. We refer to $\Delta \coloneqq \max_{X\sim X'} |q(X) - q(X')|$ as (L^1-)sensitivity of the query q.*

A randomized function

$$\mathcal{M}(X) = q(X) + Z, \quad Z \sim D$$

gives (ε, δ)-DP if the distribution D satisfies

$$\bar{\delta}_D(\varepsilon) \coloneqq \max_{s\in[-\Delta,\Delta]} D_{e^\varepsilon}(Z + s\|Z') \leq \delta, \tag{2}$$

where i.i.d. random variables $Z, Z' \sim D$.

Proof. It follows from Lemma 5 of [29], or Lemma 2.2 in [23]. □

Note that

$$\bar{\delta}_D(\varepsilon) = \max_{s\in[-\Delta,\Delta]} \sum_{k\in\mathbb{Z}} \max\left\{0, P_D(k-s) - e^{\varepsilon} P_D(k)\right\}.$$

If the PMF P_D is unimodal (i.e., it has only one local maximum), the sum on the right-hand side is maximal when $s = \Delta$.

We mainly consider probability distributions D, whose support set is bounded. In this case, the set $A = \{k \mid P_D(k) = \Pr[Z = k] > 0 \wedge \Pr[Z + \Delta = k] = 0\}$ is not empty, and one can see a lower bound

$$\bar{\delta}_D(\infty) \coloneqq \lim_{\varepsilon\to\infty} \bar{\delta}_D(\varepsilon) = \sum_{k\in A} P_D(k) \leq \bar{\delta}_D(\varepsilon) \tag{3}$$

for any ε.

3.2 Probability Generating Function

In this paper, we will treat functions of independent discrete random variables. The concept of probability generating function (PGF) is a convenient tool for dealing with such functions.

Definition 3.4. *For a discrete random variable X taking values in $\mathbb{Z}$ and its PMF $P_X(k) = \Pr[X = k]$, the probability generating function (PGF) of X is*

$$G_X(t) = \sum_{k\in\mathbb{Z}} P_X(k) t^k.$$

We treat PGF as a formal power series i.e., we do not consider its convergence.

The usefulness of PGF lies in the following fact.

Fact 3.5. *Let $X_1, X_2, \ldots, X_n$ be discrete integer-valued random variables, and $a_1, a_2, \ldots, a_n$ be integer constants. Then the PGF of a random variable $S \coloneqq \sum_{i=1}^{n} a_i X_i$ is*

$$G_S(t) = G_{X_1}(t^{a_1}) G_{X_2}(t^{a_2}) \cdots G_{X_n}(t^{a_n}).$$

By the definition of PGF, one can obtain the PMF of S as the coefficients of $G_S(t)$.

For a probability distribution D and a natural number N, we denote by D^{*N} the probability distribution of the sum $\sum_{i=1}^{N} X_i$ ($X_i \sim D$), and we call D^{*N} the N-th autoconvolution of D. The PGF of D^{*N} is $(G_{X_1}(t))^N$, and we obtain the PMF of D^{*N} by the above fact. We also define N-th autoconvolution for an integer sequence $x = \{x_i\}_{i\in\mathbb{Z}}$ or an array (an ordered list) $A = [a_0, a_1, \ldots, a_L]$ in similar way, and denote by x^{*N}.

Conversely, for a given E, we say that E is N-divisible if there exists D such that $E = D^{*N}$. At this time, D is called a component of E.

3.3 Probability Distributions and its Decomposition

In this section, we define several well-known discrete probability distributions and describe their important properties.

Definition 3.6. *The probability mass functions (PMFs) of discrete Laplace distribution* $\mathrm{DLap}(p)$*, geometric distribution* $\mathrm{Geom}(p)$*, and negative binomial distribution* $\mathrm{NB}(\alpha, p)$ *are given by the following.*

Discrete Laplace distribution
$P_{\mathrm{DLap}(p)}(k) = \frac{1-p}{1+p} p^{|k|}$ *where* $p \in (0, 1]$ *and* $k \in \mathbb{Z}$*.*

Geometric distribution
$P_{\mathrm{Geom}(p)}(k) = p^k (1-p)$ *where* $p \in (0, 1]$ *and* $k \in \mathbb{Z}_{\geq 0}$ [1].

Negative binomial distribution
$P_{\mathrm{NB}(\alpha,p)}(k) = \binom{\alpha+k-1}{k} p^k (1-p)^{\alpha}$ *where* $\alpha \in \mathbb{R}_{\geq 0}, p \in [0, 1]$ *and* $k \in \mathbb{Z}_{\geq 0}$ [2].

The discrete Laplace distribution is a discrete analog of the Laplace distribution. Some prior works ([16], for example) refer to it as the "two-sided geometric distribution".

If $p = e^{-\varepsilon/\Delta}$, then $\mathrm{DLap}(p)$ satisfies the conditions of Theorem 3.3. Specifically, the mechanism to add noise following $\mathrm{DLap}(p)$ to the query result satisfies $(\varepsilon, 0)$-DP [10, 16]. Additionally, this mechanism achieves the minimum L^1 error $\mathbb{E}[|Z|]$ and $(\varepsilon, 0)$-DP if the sensitivity $\Delta = 1$ ([15], Section VI-C).

The following theorem states that discrete Laplace and geometric distributions are N-divisible.

Theorem 3.7 ([18]). *Let Y_1, Y_2 be i.i.d. random variables following the geometric distribution* $\mathrm{Geom}(p)$*. Then, random variables $Y_1 - Y_2$ follow the discrete Laplace distribution* $\mathrm{DLap}(p)$*.*

Also, let N be a positive integer and $W_i (i = 1, \ldots, N)$ be i.i.d. random variables following negative binomial distribution $\mathrm{NB}(1/N, p)$*. Then, $\sum_{i=1}^{N} W_i$ follow the geometric distribution* $\mathrm{Geom}(p)$*.*

Hence, the discrete Laplace distribution is decomposable into the geometric distribution $\mathrm{Geom}(p)$ *or the negative binomial distribution* $\mathrm{NB}(1/N, p)$*.*

This theorem implies the following equations of PGFs holds.

$$\begin{aligned} &G_{\mathrm{DLap}(p)}(t) \\ &= G_{\mathrm{Geom}(p)}(t) G_{\mathrm{Geom}(p)}(t^{-1}) \\ &= \left(G_{\mathrm{NB}(1/N,p)}(t) G_{\mathrm{NB}(1/N,p)}(t^{-1})\right)^N. \end{aligned}$$

We can obtain the decomposition of $\mathrm{DLap}(p)$ through expanding the left-hand side of

$$\left(G_{\mathrm{DLap}(p)}(t)\right)^{1/N} = G_{\mathrm{NB}(1/N,p)}(t) G_{\mathrm{NB}(1/N,p)}(t^{-1}).$$

The consequence is

$$P_{\mathrm{DDLap}(N,p)}(k) \coloneqq (1-p)^{2/N} p^k \sum_{\substack{i-j=k \\ i,j\geq 0}} \binom{1/N + i - 1}{i} \binom{1/N + j - 1}{j} p^{2i}. \tag{4}$$

We define a distribution $\mathrm{DDLap}(N, p)$ by this PMF. Note that this is a special case of the skewed generalized discrete Laplace distribution [26].

4 Problem Description

Our approach is secure sampling by emulating a procedure in Figure 4 in two-party SMPC.

Alice and Bob can not learn the query result x nor the internally used noise z.

Our first problem is to find suitable parameters of the $\mathcal{F}_{\mathrm{DAM}}^{q,T_0,\ldots,T_{N-1},\phi}$ that fulfill the following demands:

[1]The standard definition of $P_{\mathrm{Geom}(p)}(k)$ is $(1-p)^k p$, but the notation is changed to fit the discrete Laplace distribution. See Theorem 3.7.

[2]The same as the footnote in definition of $\mathrm{Geom}(p)$.

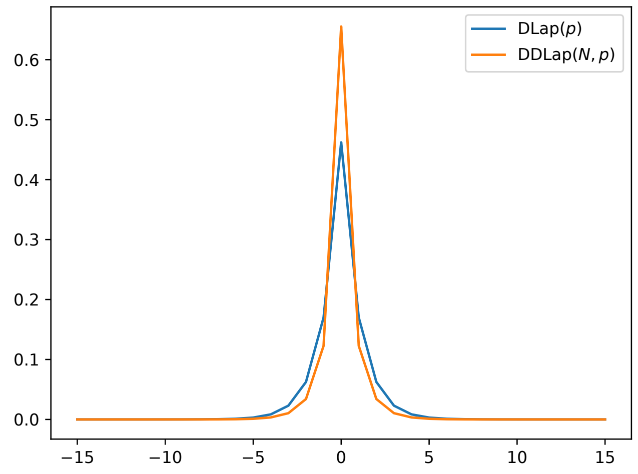

Figure 3: The PMF of the discrete Laplace distribution DLap(p) **and the decomposed discrete Laplace distribution** DDLap(N, p)**. In this example, we set parameters $p = 1/e$ and $N = 2$.**

Parameters: Let $q\colon \mathbb{D}\times\mathbb{D}\to\mathbb{Y}$ be a query function on two databases, $T_0,\dots,T_{N-1}$ be tables (arrays), and $\phi\colon \mathbb{Z}^N\to\mathbb{Z}$ be an arithmetic function.
Functionality:
1. Receive values A from Alice.
2. Receive values B from Bob.
3. Evaluate the query q and $x \longleftarrow q(A,B)$
4. Draw elements $z_0,\dots,z_{N-1}$ uniform-randomly and independently from $T_0,\dots,T_{N-1}$ respectively.
5. Compute $z \longleftarrow \phi(z_0,\dots,z_{N-1})$ and $y \longleftarrow x+z$.
6. Output y to both Alice and Bob.

Figure 4: Ideal functionality $\mathcal{F}_{\text{DAM}}^{q,T_0,\dots,T_{N-1},\phi}$ of distributing additive mechanism

(G1) the randomized function $(A,B)\mapsto y = q(A,B)+z$ achieves DP in view of both Alice and Bob,
(G2) the total size of tables $T_0,\dots,T_{N-1}$ is small (this implies small communication cost),
(G3) the arithmetic function ϕ is easy to compute (this implies a small computation cost), and
(G4) the L^1 error of the output $\mathbb{E}[|Z|]$ is acceptably small

Note that the condition (G1) is equivalent to the distribution of the noise z satisfying the condition of Theorem 3.3.

Our second problem is to construct a two-party SMPC protocol that emulates the ideal functionality $\mathcal{F}_{\text{DAM}}^{q,T_0,\dots,T_{N-1},\phi}$. The main ingredient in this problem is hiding which elements in the table are picked from Alice and Bob.

We will describe the distributions that can be sampled with random table-lookup (the step 4. in Figure 4). We define a counting function $C(k)$ for T as the number of integers $k\in\mathbb{Z}$ contained in the table T. In other words, we define by $C(k) \coloneqq \#\{i \mid T[i]=k\}$. Then the PMF of the sampled integer is written by $f_C(k) = \frac{C(k)}{\sum_{n\in\mathbb{Z}} C(n)}$. Note that the denominator is the total number of elements in the table. Furthermore, the support set of the $f_C(k)$ is a finite subset of the integers $\mathbb{Z}$ since the table is finite. Therefore, if a probability distribution D is realizable by random table-lookup, then its PMF P_D satisfies the following conditions:

(R1) the PMF is proportional to a function $C(k)$ that takes on non-negative integer values, and
(R2) it has finite support, i.e., there exists an integer w and $P_D(k) = 0$ if $|k| > w$.

The converse is also true. Therefore, instead of PMFs, we may primarily consider functions $C(k)$, which have a finite support set and take non-negative integer values.

After sampling noises by random table-lookup independently, our approach transforms them into a noise with an arithmetic function $\phi(T_1, T_2, ..., T_N)$ (the line 5. in Figure 4). If ϕ is a linear sum, the above condition holds since PGF of random variable $\sum a_i T_i$ is a product of PGFs (cf. Section 3.2). The converse does not hold in general. That is, for any distribution D satisfying the conditions and any linear sum ϕ, there does not necessarily exist a decomposition $T_1,\dots,T_N$ such that $\phi(T_1,\dots,T_N)$ follows a distribution D. A counterexample is the uniform distribution and the sum $\phi(T_1,\dots,T_N) = \sum_{i=1}^N T_i$ [3]. Therefore, we should fix ϕ first and then look for a suitable PMF.

5 Naïve Algorithms for Random Table-Lookup Approach

In this section, we look at a simple random table-lookup method to understand our first problem. In other words, we consider the case when the function ϕ in Figure 4 is the identity $\phi(e) = e$. The algorithms in this section are not practical because they generate large tables. On the other hand, they can be seen as an archetype of our proposed method and will help the reader to understand it.

5.1 Method I : Adjusting Known PMFs

If a distribution D does not satisfy the conditions (R1) and (R2) in Section 4, we can adjust it by truncation and rounding. That is, we define a PMF $P_{\tilde{D}}$ of an adjusted distribution $\tilde{D}$ as one that is proportional to a counting function

$$C_{\tilde{D}}(k;s) \coloneqq \begin{cases} \lfloor sP_D(k)\rceil & \text{if } k\in\{-w,\dots,w\} \\ 0 & \text{otherwise} \end{cases}, \tag{5}$$

where $\lfloor - \rceil$ means the function rounding to integers. Here, the scale factor $s\in\mathbb{R}_{\geq 0}$ and the width $w\in\mathbb{Z}_{\geq 0}$ are free parameters.

Let us explore how we can satisfy the condition $\bar{\delta}_{\tilde{D}}(\varepsilon)\leq\delta$ of Theorem 3.3. To investigate the effect of rounding, we extract a component from $\bar{\delta}_{\tilde{D}}(\varepsilon)$. We assume that $\tilde{D}$ is a unimodal distribution, e.g., let $\tilde{D}$ be a discrete Laplace distribution. Then, a

[3]This example cannot even be decomposed into an arbitrary discrete probability distribution.

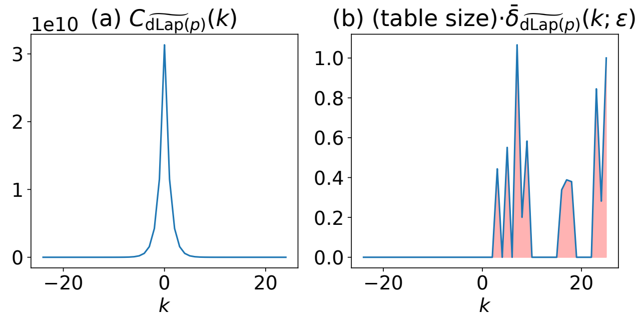

Figure 5: (a) The adjusted PMF $P_{\widetilde{\mathrm{DLap}(p)}}(k)$ of the discrete Laplace distribution $\mathrm{DLap}(p)$. (b) The values of (table size) $\cdot \bar{\delta}_{\widetilde{\mathrm{DLap}(p)}}(k;\varepsilon)$. In this example, the parameters were set to generate the smallest table that achieves $(1.0, 10^{-10})$-DP. The width was $w = 24$, the scale factor was $\approx 1.44 \times 10^{10}$, and the number of elements in the table size was $\approx 6.78 \times 10^{10}$.

point-wise version of $\bar{\delta}_{\tilde{D}}$ can be defined below.

$$\bar{\delta}_{\tilde{D}}(k;\varepsilon) := \max\left\{0, P_{\tilde{D}}(k-\Delta) - e^{\varepsilon}P_{\tilde{D}}(k)\right\} \tag{6}$$

$$= \left(\sum_{-w\le k\le w} C_{\tilde{D}}(k;s)\right)^{-1} \max\left\{0, C_{\tilde{D}}(k-\Delta;s) - e^{\varepsilon}C_{\tilde{D}}(k;s)\right\} \tag{7}$$

We have $\bar{\delta}_{\tilde{D}}(\varepsilon) = \sum_{k\in\mathbb{Z}} \bar{\delta}_{\tilde{D}}(k;\varepsilon)$. Thus, if

$$\frac{P_{\tilde{D}}(k-\Delta)}{P_{\tilde{D}}(k)} = \frac{C_{\tilde{D}}(k-\Delta;s)}{C_{\tilde{D}}(k;s)} \le e^{\varepsilon} \tag{8}$$

holds, then $\bar{\delta}_{\tilde{D}}(k;\varepsilon) = 0$ and this makes $\bar{\delta}_{\tilde{D}}(\varepsilon)$ is smaller. The left-hand side is close to $\frac{P_{\tilde{D}}(k-\Delta)}{P_{\tilde{D}}(k)}$ when $P_{\tilde{D}}(k)$ is large or when s is large.

With these tools, we can see that it is difficult to create a distribution $\tilde{D}$ with small $\bar{\delta}_{\tilde{D}}(\varepsilon)$ from an analytical PMF using this naïve method. An example of applying this method to $\mathrm{DLap}(p)$ is shown in Figure 5. This figure displays the values of a counting function of $\tilde{D} = \widetilde{\mathrm{DLap}(p)}$ (the adjusted $\mathrm{DLap}(p)$), and the values of a function

$$\text{(table size)} \cdot \bar{\delta}_{\widetilde{\mathrm{DLap}(p)}}(k;\varepsilon)$$
$$\left(= \max\left\{0, C_{\widetilde{\mathrm{DLap}(p)}}(k-\Delta;s) - e^{\varepsilon}C_{\widetilde{\mathrm{DLap}(p)}}(k;s)\right\}\right).$$

This function shows exactly the effect of rounding error since $P_{\mathrm{DLap}(p)}(k-\Delta;s) - e^{\varepsilon}P_{\mathrm{DLap}(p)}(k;s) = 0$. As this example shows, this function exceeds one at many points k, and this implies

$$\text{(table size)} > \left(\bar{\delta}_{\widetilde{\mathrm{DLap}(p)}}(k;\varepsilon)\right)^{-1} \ge \frac{1}{\delta}.$$

The $\frac{1}{\delta}$ is usually required to be larger than 10^5 or 10^{10} for privacy. Moreover, it is difficult to predict the random behavior of the function. Thus, it is also difficult to generate a small table with small $\bar{\delta}_{\tilde{D}}(\varepsilon)$ from an analytical PMF using this naïve method.

5.2 Method II : Iterative Algorithm

We consider the reduction of $\bar{\delta}_D(\varepsilon)$ from the sufficient condition that it is small. As in Section 4, we construct a function $C_D(k)$ whose values are non-negative integers and obtain a PMF by normalizing it. From the discussion in the previous section, it is sufficient to create an unimodal function $C_D(k)$ satisfying $C_D(k-\Delta) \le e^{\varepsilon}C_D(k)$ at each k. If this condition is satisfied, then $\bar{\delta}_D(k)$ coincides with the lower bound $\bar{\delta}_D(\infty)$. It is easy to determine iteratively the value of the function $C_D(k)$ so that it satisfies this inequality. The algorithm is described in Algorithm 1.

```
Input: Privacy budgets to achieve $\varepsilon(>0), \delta(\in(0,1))$, the
       sensitivity of a query $\Delta(\in\mathbb{Z}_{\ge 1})$
r ← $e^{\varepsilon/\Delta}$

/* Set the initial value so that $\lfloor A_0 r\rfloor > A_0$ */
if r ≥ 2 then $A_0$ ← 1
else $A_0$ ← $\lceil \frac{1}{r-1} \rceil$ // Remark that 1 < r < 2 in here.


/* Main loop */
A ← [ $A_0$ ]
for w ≥ 0 do
    $m$ ← ⌊rA[w]⌋
    A ← InsertCenter(A, $m$)

    /* Define a distribution $D_{\mathrm{Alg1}}$ */
    Define a count function $C_{D_{\mathrm{Alg1}}}(k) := A[k + (w+1)]$ for
      $k \in \{-(w+1), \dots, (w+1)\}$
    Define a distribution by a PMF $P_{D_{\mathrm{Alg1}}}(k) := \frac{C_{D_{\mathrm{Alg1}}}(k)}{\sum C_{D_{\mathrm{Alg1}}}(k)}$

    /* Check if $\bar{\delta}_{D_{\mathrm{Alg1}}}$ is small enough */
    if (w > Δ) and $\bar{\delta}_{D_{Alg1}}(\varepsilon) \le \delta$ then
        return $P_{D_{Alg1}}(k)$
    end
end
```

Algorithm 1: An algorithm to generate scaled PMF which can be used to achieve (ε,δ)-DP

The function $\mathtt{InsertCenter}(A, x)$ in Algorithm 1 takes an array $A = [A[0], \dots, A[2w]]$ of odd length and a value x and gives a new array $[A[0], \dots, A[w], x, A[w], \dots, A[2w]]$ of length $2(w+1)+1$. Note that the middle element $A[w]$ of A is duplicated.

Figure 6 shows an example of the output.

While the algorithm grows the array A, it follows a recurrence relation

$$A[k+1] = \lfloor rA[k] \rfloor = A[2w-(k+1)] \quad \text{for } k = 0, \dots, w-1$$

where the length of A is $2w+1$ and $r = e^{\varepsilon/\Delta}$. This implies $\bar{\delta}_{D_{\mathrm{Alg1}}}(k;\varepsilon) = 0$ for $k = 0, \dots, 2w-\Delta$. This always holds until $\bar{\delta}_{D_{\mathrm{Alg1}}}(\varepsilon) \le \delta$ is satisfied and the algorithm finishes. Note that Algorithm 1 becomes one that computes the PMF of the truncated discrete Laplace distribution numerically if we do not use the floor function. Additionally, we can see $A[k+1] = A[2w-(k+1)] > A[k]$ from the definition of the

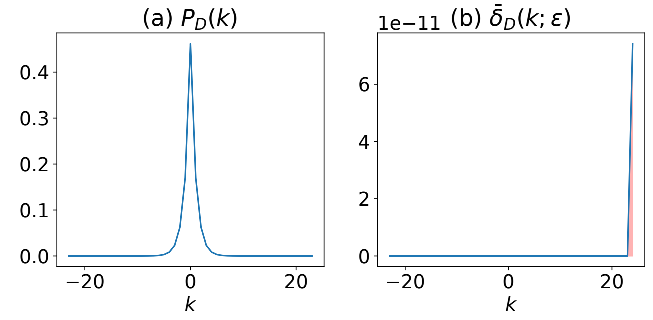

Figure 6: An example of discrete probability distribution D_{Alg1} generated by Algorithm 1. The input parameters were $\varepsilon = 1.0, \delta = 10^{-10}, \Delta = 1$. The resulting table size was 100.5 GiB and $\bar{\delta}_{D_{\mathrm{Alg1}}}(\varepsilon) \sim 10^{-10.1}$. In (b), the scale of the vertical axis is 10^{-11}, and the pink area is $\bar{\delta}_D(\varepsilon)$.

initial value A_0. Especially if $1 < r < 2$, we have

$$rA_0 = A_0 + (r-1)A_0 \geq A_0 + (r-1)\frac{1}{r-1} = A_0 + 1,$$

since $A_0 := \lceil \frac{1}{r-1} \rceil \geq \frac{1}{r-1}$.

In this way, we obtain a distribution D_{Alg1} with $\bar{\delta}_{D_{\mathrm{Alg1}}}(\varepsilon)$ equal to the lower bound $\bar{\delta}_{D_{\mathrm{Alg1}}}(\infty)$. Moreover, the above algorithm allows us to achieve (ε,δ)-DP with near-optimal L^1 error $\mathbb{E}[|X|]$. Despite these efforts, the algorithm makes large tables for strong privacy guarantees. From the construction, $\bar{\delta}_D(\varepsilon)$ can be evaluated from below as follows.

$$(\varepsilon =)\bar{\delta}_{D_{\mathrm{Alg1}}}(\infty) = \frac{\sum_{k=w-\Delta}^{w} C_{D_{\mathrm{Alg1}}}(k)}{\sum_{-w\leq k\leq w} C_{D_{\mathrm{Alg1}}}(k)} \tag{9}$$

$$\geq \frac{1}{\sum_{-w\leq k\leq w} C_{D_{\mathrm{Alg1}}}(k)} = \frac{1}{\text{the table size}} \tag{10}$$

Therefore, the table must be larger than $1/\delta (\geq 10^5)$ in this method, for exactly the same reasons as in the previous section. This is why the table for a simple random table-lookup method is huge.

6 Our Algorithm to Make Small Table for Our Approach

6.1 Overview of Our Method

For tables made for the naïve random table-lookup method, we considered the condition $\bar{\delta}_D(\varepsilon) \leq \delta$ of Theorem 3.3 in the previous section. We observed that a lower bound of $\bar{\delta}_D(\varepsilon)$:

$$\bar{\delta}_D(\varepsilon) \geq \bar{\delta}_D(\infty) = \frac{\sum_{k=w-\Delta}^{w} C_D(k)}{\sum_{-w\leq k\leq w} C_D(k)} \geq \frac{1}{\text{table size}}. \tag{11}$$

Thus, we concluded that the table must be larger than $1/\delta$ (usually $\geq 10^5$).

We address this problem with two ideas. Our first idea is to decompose the noise into sums of random values (our second idea is described in Section 6.3). In other words, the first idea is to use the summation function $\sum_N(t_0,\ldots,t_{N-1}) = \sum_{i=0}^{N-1} t_i$ as a transformation function in the functionality (Figure 4). This idea reduces $\bar{\delta}_D(\infty)$ that is the lower bound of $\bar{\delta}_D(\varepsilon)$ drastically. For simplicity, we use a single table instead of N different tables.

We use notations as in Section 4 and denote the distribution of values sampled from the table T by D. The distribution of the output of the functionality $\mathcal{F}_{\mathrm{DAM}}^{T,\sum_N}$ is written by D^{*N}. Again, the similar formula

$$\bar{\delta}_D(\varepsilon) \geq \bar{\delta}_D(\infty) = \frac{\sum_{k=w-\Delta}^{w} C_D(k)}{\sum_{-w\leq k\leq w} C_D(k)}$$

holds as well as the expression (11). However, unlike the case where the sum is not considered, we can see $\bar{\delta}_{D^{*N}}(\infty) = \frac{NC_D(w)}{(\sum_{-w\leq k\leq w} C_D(k))^N}$ if $\Delta = 1$. It decreases much more rapidly than $\bar{\delta}_D(\infty) = \frac{C_D(w)}{\sum_{-w\leq k\leq w} C_D(k)}$. This suggests that a small table can achieve the condition $\bar{\delta}_{D^{*N}}(\infty) \leq \delta$.

Below, we consider a distribution D that is feasible with random table-lookup (see Section 4), and its N-th autoconvolution D^{*N} gives DP of integer-valued queries.

Although the idea above is simple, it is not yet easy to make a small table that satisfies the above conditions. A quick idea would be to apply Method I (Section 5.1) to a distribution component that achieves DP, e.g., the decomposed discrete Laplace distribution DDLap(N,p). However, as we saw in Section 5.1, $\bar{\delta}_{\tilde{D}}(k;\varepsilon)$ becomes large in many points, resulting in the need for a large table. The numerical behavior of $\bar{\delta}_{\tilde{D}}(k;\varepsilon)$ is shown in the experiment in Section 8.

In this case, the major reason is the truncation (approximating of the tail value by 0), rather than the rounding error. Since it takes real numbers as values, adjusting as in Section 5.1 is still required for actual use. To address this problem, we propose a PMF having a finite support set in this section. After that, we propose a method in Section 6.3 that approximates the PMF constructed in the next section. This is similar to the relationship between Method II (Section 5.2) and the discrete Laplace distribution.

Let us consider a sufficient condition for $\bar{\delta}_{D^{*N}}(\varepsilon) = \bar{\delta}_{D^{*N}}(\infty)$, based on Section 5.2. The only difference in settings from Section 5.2 is that D is changed to D^{*N}, thus we can obtain the sufficient condition by just replacing the symbols in the formula in Section 5.2. The obtained sufficient condition is that the PMF of the autoconvolution $P_{D^{*N}}$ is a constant multiple of a unimodal integer-valued function $C_{D^{*N}}(k)$ that satisfies the following condition:

$$e^{\varepsilon} C_{D^{*N}}(k) - C_{D^{*N}}(k-\Delta) \geq 0 \tag{12}$$

for $k \in \{-Nw,\ldots,\Delta\}$ and $C_D(-k) = C_D(k)$ for $k \in \{-w,\ldots,w\}$. If this difference on the left-hand side is small, the L^1 error of the distribution becomes small. This is a system of polynomial inequalities with the value of $C_D(k)$ as a variable. As such, it is generally a very difficult problem for which to find even a single solution.

Another important idea (or observation) for solving this next problem is that a solution is easy to find for the "tail" part of the system but quite hard to solve for the whole system. For $k \in \{(N-1)w,\ldots,Nw\}$, each inequality system (12) can be written in the form:

$$e^{\varepsilon}N \cdot C_{D^{*N}}(w)^{N-1} \cdot C_{D^{*N}}(k-(N-1)w) + \begin{pmatrix}\text{a degree } N \text{ polynomial of } C_{D^{*N}}(k-(N-1)w+ \\ 1), \dots, C_{D^{*N}}(w-1)\end{pmatrix}.$$

Thus, by setting the value of the initial term $C_{D^{*N}}(w)$ and iteratively solving the linear inequality for $C_{D^{*N}}(k-(N-1)w)$, one can find a solution for a part of the system (12) for $k \in \{(N-1)w, \dots, Nw\}$.

In the next section, we analyze the distribution obtained by solving the “tail" part of the equations system induced by the inequality system (12). This shows that the solution of the “tail" part satisfies the whole inequality system. Section 6.3 gives an integer-valued function $C_{D^{*N}}(k)$ by rounding the solution to integers while solving the same linear inequality sequentially. Without this rounding, the algorithm in Section 6.3 becomes an algorithm to compute the analytical solution given in Section 6.2. Since what is obtained by this algorithm is an approximate solution of the inequality system, it cannot be proved that all of the inequality system (12) is satisfied. However, the results for the analytical solutions in the previous section and the checks in the algorithm guarantee that the output satisfies the system of inequalities.

6.2 Approximate Decomposition of DLap(p)

On the basis of this observation, we discovered that it is better to consider a distribution $D(p, N, w)$ with the following PMF. This PMF is related to the PMF of the negative binomial distribution.

Definition 6.1. *Let $p \in (0, 1], N \in \mathbb{Z}_{>0}$ and $w \in \mathbb{Z}_{\geq 0}$. The PMF of a distribution $D(p, N, w)$ is defined as*

$$P_{D(p,N,w)}(k) = C \cdot \binom{1/N + w - |k| - 1}{w - |k|} r^{w-|k|}, \quad \text{where } k \in \{-w, \dots, +w\}. \tag{13}$$

Here, $r := \frac{1}{p} (\geq 1)$ and C is the normalization constant.

One can see the following equations about geometric series:

$$\begin{aligned}&\left(t^0 + (rt)^1 + (rt)^2 + \dots\right)^{1/N} \\ &= \left(1 - rt^{-1}\right)^{-1/N} \\ &= \sum_{k=0}^{\infty} \binom{-1/N}{k} (-r)^k t^k \\ &= \sum_{k=0}^{\infty} \binom{1/N + k - 1}{k} r^k t^k.\end{aligned}$$

The PGF corresponding to $P_{D(p,N,w)}(k)$ can be made by cutting and pasting this series. Thus the PMF of N-th autoconvolution $P_{D(p,N,w)^{*N}}(k)$ coincides with $P_{\mathrm{DLap}(p)}(k)$ in the tail part.

$$\begin{aligned}P_{D(p,N,w)^{*N}}(Nw-0) &= C^N 1 \\ P_{D(p,N,w)^{*N}}(Nw-1) &= C^N r \\ &\dots \\ P_{D(p,N,w)^{*N}}(Nw-w) &= C^N r^w \\ &\dots \\ P_{D(p,N,w)^{*N}}(-Nw+w) &= C^N r^w \\ &\dots \\ P_{D(p,N,w)^{*N}}(-Nw+1) &= C^N r \\ P_{D(p,N,w)^{*N}}(-Nw) &= C^N 1\end{aligned}$$

Therefore, $\delta(x; \varepsilon) = 0$ at $|k| \geq (N-1)w$ for $r = e^{\varepsilon/\Delta}$. Furthermore, as shown below, $\delta(x; \varepsilon) = 0$ in $|k| < (N-1)w$ as well.

Theorem 6.2. *Let N, w be a positive integer and $p \in (0, 1)$. For sufficiently large $w \in \mathbb{Z}_{>0}$ and $k \in \{0, \dots, Nw-1\}$,*

$$P_{D(p,N,w)^{*N}}(k) \leq r P_{D(p,N,w)^{*N}}(k+1).$$

Proof. See Appendix A. □

This fact implies $\bar{\delta}_{D(p,N,w)}(\varepsilon)$ coincides with the lower bound $\bar{\delta}_{D(p,N,w)}(\infty)$ when $p \geq e^{\varepsilon/\Delta}$.

Theorem 6.3. *Let $q \colon \mathbb{D} \to \mathbb{Z}$ be a function with L^1 sensitivity Δ, $\varepsilon (> 0)$ be a positive real value, and $D = D(e^{-\varepsilon/\Delta}, N, w)$. Let δ be a positive real number that satisfies*

$$\sum_{k=Nw}^{Nw-\Delta+1} P_{D^{*N}}(k) \leq \delta < 1.$$

Then, a randomized function

$$\mathcal{M}(X) = q(X) + \sum_{i=1}^{N} Z_i \quad , Z_1, \dots, Z_N \sim D \tag{14}$$

achieves (ε, δ)-DP.

Proof. According to the previous theorem,

$$P_{D^{*N}}(k) \leq r P_{D^{*N}}(k+1),$$

for $k \in \{0, \dots, Nw-1\}$. By using it recursively

$$P_{D^{*N}}(k) \leq r^s P_{D^{*N}}(k+s),$$

for $s = 0, \dots, \Delta$. Here, $r^s = e^{\varepsilon \frac{s}{\Delta}}$. As a consequence, we have $\bar{\delta}_D(k; \varepsilon) = 0$ for $k \in \{0, \dots, Nw\}$. From this and the symmetry $P_{D^{*N}}(-k) = P_{D^{*N}}(k)$, the equality $\bar{\delta}_{D^{*N}}(\varepsilon) = \bar{\delta}_{D^{*N}}(\infty)$ holds. Since the assumption of this theorem is

$$\bar{\delta}_{D^{*N}}(\infty) = \sum_{k=Nw}^{Nw-\Delta+1} P_{D^{*N}}(k) \leq \delta,$$

this theorem follows from Theorem 3.3. □

6.3 Iterative Algorithm

We have a distribution $D(p, N, w)$ that has a finite support set, and its N-th autoconvolution gives DP to integer-valued queries. We can make it a constant multiple of an integer function as in Section 5.1. Since the PMF $P_D(k)$ values are small in the tail $k \in \{(N-1)w, \dots, Nw\}$, the rounding error is larger than the original value. This inaccuracy causes $\bar{\delta}_{D(e^{-\varepsilon/\Delta}, N, w)}(k; \varepsilon) > 0$ in the tail.

From the observation discussed in Section 6.1, we can iteratively determine the value of PMF that enjoys $\delta(x; \varepsilon) = 0$ in the tail part. Algorithm 2 is the detailed procedure.

```
Input: Privacy budgets to achieve $\varepsilon(>0), \delta(\in(0,1))$,
       sensitivity $\Delta(\in \mathbb{Z}_{\geq 1})$, number of decomposition
       $N(\geq 1)$, and initial value $A_0$ (if not specified, = 1)
A ← [ A_0 ]
for w ≥ 0 do
    $A_{+x}$ ← InsertCenter(A, $x$)
        // The length of array A is $2(w+1)+1$
    $A_{+x}^{*N}$ ← $N$-th autoconvolution of the array $A_{+x}$
        // See section 3.2 for the definition
    $s$ ← the solution of a linear equation of $x$ :
      $A_{+x}^{*N}[w] == e^{\varepsilon/\Delta}(A_{+x}^{*N}[w-1])$
        // Left-hand side is a constant
    A ← InsertCenter(A, $\lfloor s \rfloor$)
    $A^{*N}$ ← $N$-th autoconvolution of the array A
      // The length of array $A^{*N}$ is $2N(w+1)+1$

    /* Define a distribution $D_{\mathrm{Alg2}}$ */
    Define a count function $C_{D_{\mathrm{Alg2}}}(k) := A[k + N(w+1)]$
      for $k \in \{-N(w+1), \dots, N(w+1)\}$
    Define a distribution by a PMF $P_{D_{\mathrm{Alg2}}}(k) := \frac{C_{D_{\mathrm{Alg2}}}(k)}{\sum C_{D_{\mathrm{Alg2}}}(k)}$

    /* Check if $D_{\mathrm{Alg2}}$ is unimodal and
       $\bar{\delta}_{D_{\mathrm{Alg2}}}(k;\varepsilon) = 0$ */
    if there exists $i \in [0, N(w+1))$ that satisfies neither
      $A^{*N}[i] > 0$ nor $A^{*N}[i+1] \leq e^{\varepsilon/\Delta} A^{*N}[i]$ then
        $A_0$ ← $A_0 + 1$
        Restart this algorithm
    end

    /* Check if $\bar{\delta}_{D_{\mathrm{Alg2}}}(\varepsilon) \leq \delta$ */
    if $(w > \Delta)$ and $\bar{\delta}_{D_{\mathrm{Alg2}}}(\varepsilon) \leq \delta$ then
        return $P_{D_{\mathrm{Alg2}}}(k)$
    end
end
```

Algorithm 2: An algorithm to generate scaled PMF which can be used to achieve (ε, δ)-DP

For the definition of the function `InsertCenter`, see Section 5.2.

The algorithm generates a distribution that is symmetric and satisfies $\bar{\delta}_{A^{*N}}(k, \varepsilon) = 0$ in the tail $k \in \{0, \dots, w-1\} \cup \{2Nw - w + 1, \dots, 2Nw\}$. The shape of the generated distribution is almost the same as in Figure 6. As in the discussion above, the element $A_{+x}^{*N}[k]$ is a constant in the tail $k \in \{0, \dots, w-1\} \cup \{2Nw - w + 1, \dots, 2Nw\}$, and a polynomial of x elsewhere. In particular, $A_{+x}^{*N}[k]$ is a linear formula of x for $k = w$ or $k = 2Nw - w$, the one step inside of the tail. This means that the inequality $C_A(k-1) \leq e^{\varepsilon/\Delta} C_A(k)$ is a linear one at the two point. Thus, we can determine x easily, such that the inequality $\bar{\delta}_{A^{*N}}(k, \varepsilon) = 0$ holds in the larger set $\{0, \dots, w\} \cup \{2Nw - w, \dots, 2Nw\}$. By repeating this, the table can be made larger and larger while satisfying the inequality.

The distribution generated by Algorithm 2 approximates the PMF defined in the previous section. In fact, if we do not approximate in line 9, Algorithm 2 becomes one that computes the PMF $P_{D(N,p,w)}(k)$ numerically. Also note that when $N = 1$, it coincides with Method II (Section 5.2). However, this observation alone does not guarantee that Algorithm 2 gives DP. Strictly speaking, we must check that $\bar{\delta}_D(k; \varepsilon) = 0$ holds in the loop. The following is the proof of this.

Theorem 6.4. *Let $q \colon \mathbb{D} \to \mathbb{Z}$ be a function with sensitivity Δ, and D be a distribution whose PMF is produced by Algorithm 2. Then,*

$$\mathcal{M}(X) = q(X) + \sum_{i=1}^{N} Z_i \quad , Z_1, \dots, Z_N \sim D \tag{15}$$

achieves (ε, δ)-DP.

Proof. Let P_{Alg2} be the output of Algorithm 2. By its construction, P_{Alg2} has the following properties.

(1) it is symmetric about zero i.e. $P_{\mathrm{Alg2}}(-k) = P_{\mathrm{Alg2}}(k)$.
(2) its support set is $\{-w, \dots, +w\}$.
(3) P_{Alg2}^{*N} is monotonically increasing for $i \in \{-w, \dots, 0\}$.
(4) $e^{\varepsilon/\Delta} P_{\mathrm{Alg2}}^{*N}[i] \leq P_{\mathrm{Alg2}}^{*N}[i+1]$ for $i \in \{-w, \dots, 0\}$.

Since the properties (1), (2) and (3), D_{Alg2} is a unimodal distribution, so

$$\bar{\delta}_{P_{\mathrm{Alg2}}^{*N}}(\varepsilon) = \sum_{k=-w,\dots,+w} \max\{0, P_{\mathrm{Alg2}}^{*N}(k) - e^{\varepsilon} P_{\mathrm{Alg2}}^{*N}(k + \Delta)\}.$$

Also, from properties (1) and (3), $\bar{\delta}_{P_{\mathrm{Alg2}}}(\varepsilon) = \bar{\delta}_{P_{\mathrm{Alg2}}}(\infty)$. The termination condition of the algorithm means that it is less than δ. Therefore, the theorem is shown by Theorem 3.3. □

Our goal is to generate a small table, and our idea is toward it. A theoretical evaluation of the table size is given below.

Theorem 6.5. *Let P_{Alg2} be the PMF generated by Algorithm 2 and δ' be a positive real value. If*

$$\delta' \leq \bar{\delta}_{P_{\mathrm{Alg2}}}(\varepsilon) (\leq \delta)$$

holds, then the table size $S := \sum_{-w \leq k \leq w} C(k)$ has an upper bound

$$S \leq \left(\frac{\sum_{k'=0}^{\Delta-1} e^{k'\varepsilon/\Delta} c}{\delta'} \right)^{1/N},$$

where $c := C(w)$.

In most cases $\bar{\delta}_{P_{\mathrm{Alg2}}}(\varepsilon)$ is very slightly smaller than δ. Assuming this, both sides are approximately equal.

Proof. As the previous theorem, $\bar{\delta}_{D_{\mathrm{Alg2}}}(\varepsilon) = \bar{\delta}_{P_{\mathrm{Alg2}}}(\infty)$. We can see $C_{D_{\mathrm{Alg2}}}(k) \leq r^{k+Nw} C_{D_{\mathrm{Alg2}}}(w)$ for $k \in \{-Nw, \dots, -Nw+w\}$ and for $r = e^{\varepsilon/\Delta}$by the construction. Therefore,

$$\bar{\delta}_{P_{\mathrm{Alg2}}}(\infty) = \frac{\sum_{k=w-\Delta}^{w} C(k)}{\sum_{-w \leq k \leq w} C(k)} \leq \frac{\sum_{i=0}^{\Delta-1} r^i C(w)}{\sum_{-w \leq k \leq w} C(k)}.$$

A straightforward calculation shows the desired result. □

7 Our Secure Sampling Protocol

So far, we can sample noises for DP with a small table T and summation $\sum_N : (t_0, \dots, t_{N-1}) \mapsto \sum_{i=0}^{N-1} t_i$. This section provides a two-party secure computation protocol that realizes distributing additive mechanism functionality $\mathcal{F}_{\mathrm{DAM}}^{q,T,\Sigma}$. The main challenge in realizing the functionality $\mathcal{F}_{\mathrm{DAM}}^{q,T,\Sigma}$ is implementing the random table lookup in SMPC. The selection of table elements must not be learned by either party.

A combination of oblivious transfer and shuffle can solve this problem. We use the additive secret sharing on the finite cyclic group $\mathbb{Z}_\ell (\ell \in \mathbb{Z}_{>0})$ as the secret computation scheme. Firstly, Alice samples a random mask $m \in \mathbb{Z}_\ell$ and a permutation π on the indices $\{0, \dots, \#T-1\}$, and makes a "doubly" masked table T' as $T'[i] = T[\pi(i)] - m \in \mathbb{Z}_\ell$. Next, Alice inputs T' as a sender, and Bob inputs uniformly-random index c as a receiver, to 1-out of-($\#T$) oblivious transfer. Then Alice's random mask m and $T'[c] = T[\pi(c)] - m$ received by Bob are additive shares of $T[\pi(c)]$. Obviously, neither Alice nor Bob can know the chosen element $\pi(c)$. The protocol is described fully in Protocol 3.

In our protocol, we use the two-party secure function evaluation functionality $\mathcal{F}_{\mathrm{SFE}}^{q}$ for a function $q : \mathbb{D} \times \mathbb{D} \to \mathbb{Z}_\ell$. It receives databases A, B from each party and outputs additive shares of the evaluation result in $q(A, B)$ to each party, and neither can obtain any other information. Note that the function evaluated is public in our situation. This functionality can be realized by secret shared function [5, 6], for example.

Furthermore, we use the 1-out-of-n oblivious transfer functionality, as we mentioned above. We denote this functionality $\mathcal{F}_{n\text{-OT}}$.

Note that operations within the For-loop can be executed in parallel. One can construct a batched version with batched OT, but this paper does not prove its security.

The protocol can be scaled up with more participants by using private information retrieval instead of oblivious transfer. Again, in this case, the participants who generate the permutations and the participants who choose the elements must be different. We think this change may reduce the communication cost, but we do not know about the computation cost.

7.1 Security

We prove the two-party protocol $\Pi_{\mathrm{DAM}}^{q,T,\Sigma_N}$ (Protocol 3) is secure in the presence of a semi-honest adversary. We define the summation function as $\sum_N : \mathbb{Z}^N \to \mathbb{Z}; (y_0, \dots, y_{N-1}) \mapsto \sum_{i=0}^{N-1} y_i$.

Theorem 7.1. *Protocol $\Pi_{\mathrm{DAM}}^{q,T,\Sigma_N}$ (Protocol 3) securely realizes ideal functionality $\mathcal{F}_{\mathrm{DAM}}^{q,T,\Sigma_N}$ against a semi-honest adversary in the $(\mathcal{F}_{\mathrm{SFE}}^{q}, \mathcal{F}_{\mathsf{L}\text{-OT}})$-hybrid model.*

```
Parameters: a query function $q : \mathbb{D} \times \mathbb{D} \to \mathbb{Z}_\ell$ on two
            databases, a table (an array) of integers
            $\mathsf{T} \in \mathbb{Z}_\ell^{\mathsf{L}}$ of length L, number of times to draw
            from the table $N (> 0)$
Input: Alice's databases $A$ and Bob's databases $B$
Output: Both's output $y$
1  /* Evaluate the query $q(A,B)$ securely */
2  Alice and Bob invoke the secure function evaluation
     functionality $\mathcal{F}_{\mathrm{SFE}}^{q}$
3  Alice and Bob input $A$ and $B$ respectively, and obtain
     additive shares $[\![x]\!]_A$ and $[\![x]\!]_B$ respectively
4  for $k = 1$ to $N$ do
5      /* Sample an element of the table
         uniform-randomly */
6      Alice samples a uniformly-random number
         $[\![z_k]\!]_A \xleftarrow{\$} \mathbb{Z}_\ell$
7      Alice generates random permutation $\pi_k$ on
         $\{0, \dots, \mathsf{L}-1\}$
8      Alice makes a table (an array) $\mathsf{T}'_k$ such that
         $\mathsf{T}'_k[i] = \mathsf{T}[\pi_k(i)] - [\![z_k]\!]_A$ for $i = 0, \dots, \mathsf{L}-1$
9      Bob samples a random index $c_k \xleftarrow{\$} \{0, \dots, \mathsf{L}-1\}$
10
11     Alice and Bob invoke the 1-out of-L oblivious transfer
         functionality $\mathcal{F}_{\mathsf{L}\text{-OT}}$
12     Alice inputs $\mathsf{T}'$ and Bob inputs $c_k$ to $\mathcal{F}_{\mathsf{L}\text{-OT}}$
13     Bob receives $[\![z_k]\!]_B$ from $\mathcal{F}_{\mathsf{L}\text{-OT}}$
14 end
15 Alice computes $[\![y]\!]_A \leftarrow [\![x]\!]_A + \sum_{k=1}^{N} [\![z_k]\!]_A$ locally
16 Bob computes $[\![y]\!]_B \leftarrow [\![x]\!]_B + \sum_{k=1}^{N} [\![z_k]\!]_B$ locally
17
18 Alice and Bob reveal their share $[\![y]\!]_A$ and $[\![y]\!]_B$ to each
     other
19 Alice and Bob obtain the result $y \leftarrow [\![y]\!]_A + [\![y]\!]_B$
```

Protocol 3: A two-party protocol $\Pi_{\mathrm{DAM}}^{q,T,\Sigma_N}$ for functionality $\mathcal{F}_{\mathrm{DAM}}^{q,T,\Sigma_N}$

Proof. We will construct simulators of each party's view, and prove that they are computationally indistinguishable from the real view.

Corrupt Alice. Alice's view consists of additive share $[\![x]\!]_A, [\![y]\!]_B$, and the result y ($k = 1, \dots, N$, $i = 0, \dots, L-1$). We can simulate her view by doing the following:

- Sample $[\![x]\!]_A$ from the uniform distribution on $\mathbb{Z}_\ell$.
- For each $k = 1, \dots, N$:
  - Sample $[\![z_k]\!]_A$ from the uniform distribution on $\mathbb{Z}_\ell$.
  - Compute $[\![y]\!]_A \leftarrow [\![x]\!]_A + \sum_{k=1}^{N} [\![z_k]\!]_A$.
- Obtain y from the ideal functionality $\mathcal{F}_{\mathrm{DAM}}^{q,T,\Sigma_N}$.
- Compute $[\![y]\!]_B = y - [\![y]\!]_A$.

Since the share $[\![x]\!]_A$ is the output of the ideal functionality $\mathcal{F}_{\mathrm{SFE}}^{q}$, it can be simulated as a uniformly random element. $[\![y]\!]_A$ only depends on $[\![x]\!]_A$ and $[\![z_1]\!]_A$, and the latter is generated by Alice independently. Thus, we can simulate these data from Alice's point of view.

We will prove that we can simulate the result y of the real protocol with the ideal functionality $\mathcal{F}_{\mathrm{DAM}}^{q,T,\Sigma_N}$. Alice and Bob separately choose π_k and c_k respectively, where $k - 1, \ldots, N$. Furthermore, the random variables $\pi_1, \ldots, \pi_N$ and $c_1, \ldots, c_N$ are independent. This results in $\pi_1(c_k), \ldots, \pi_N(c_N)$ being random variables that follow a uniform distribution on $\{0, \ldots, L-1\}$. Note that the two pairs of random variables c_k and $\pi_k(c_k)$, and π_k and $\pi_k(c_k)$ are independent. This means we can simulate $\pi_k(c_k)$ without random values π_k or c_k generated by real participants. From the above, we conclude that the distribution of $y = x + \sum_{k=1}^{N} T[\pi_k(c_k)]$ is equal to that of the output of the ideal functionality $\mathcal{F}_{\mathrm{DAM}}^{q,T,\Sigma_N}$.

Corrupt Bob. Bob's view consists of the additive share $[\![x]\!]_B, [\![z_k]\!]_B$, $[\![y]\!]_A, [\![y]\!]_B$, and the result y $(k = 1, \ldots, N,\ i = 0, \ldots, L-1)$. The simulator for his view is the same as for Alice's. Since Bob's view contains $[\![z_k]\!]_B$ in addition to Alice's view, let us show that this can also be simulated. $[\![z_k]\!]_B = T[\pi_k(c_k)] - m$ can be computed from a uniformly random element of the public table $T[\pi_k(c_k)]$ and a uniformly random value in $\mathbb{Z}_\ell$. Thus we can simulate $[\![z_k]\!]_B$ by sampling from the uniform distribution on $\mathbb{Z}_\ell$. □

8 Experimental Evaluation

In this section, we experiment with our method (Algorithm 2 and Protocol 3) and compare it with the prior methods in terms of runtimes, communication cost, and L^1 (additive) error. For the comparison, we take a representative from each of the 3 approaches.

8.1 Experiment Setting

This section describes the environment, implementation, and parameters we used in our experiments.

8.1.1 Environments. We run our implementation on a desktop computer with Intel Core i9-9900K (3.60 GHz × 8) CPU and 16 GiB RAM. Only LAN was used as the network environment. Each measurement is performed 10 times, and we report the averages.

8.1.2 Implementation Details. We implement our protocol using the 1-out-of-L oblivious transfer protocol in [21] implemented in libOTe [25]. The table was implemented as a 16-bit signed integer array since the table elements never exceeded $\pm 2^{15}$. We did not implement the optimization using batched OT.

8.1.3 Parameters. For simplicity, we fixed the sensitivity of queries $\Delta = 1$. As mentioned in Section 3.3, the discrete Laplace distribution $\mathrm{DLap}(p)$ achieves the minimum L^1 error under $(\varepsilon, 0)$-DP if $\Delta = 1$ and there were no other technical constraints.

8.2 Compared Methods

We refer to the following protocols as examples of a secure sampling protocol of other approaches for the discrete Laplace distribution.

As an example of protocols in the as-in-plaintext approach, we refer to a protocol in Keller et al. [19], which is considered to be the current state-of-the-art in this approach. This protocol computes a binary circuit to transform random bits into noise following the discrete Laplace distribution in Yao's garbled circuit scheme. Their circuit has 1.86×10^7 (18.7 million) AND gates. We have taken measurements from their paper because their protocol implementation is not publicly available. Note that their protocols perform the same for many ε, while our method performs differently depending on the privacy parameters we want to achieve.

As an example of methods in the distributed sampling approach, we refer to a simple protocol in [27]. In this example, two participants sample i.i.d. noises following the discrete Laplace distribution separately in the plaintext computation, and they add them to the exact analysis result in a secure multiparty computation. this example method gives statistical ε-DP, while our method and methods in the as-in-plaintext approach give computational DP.

As examples of methods in the random table-lookup approach, we refer to a method using our protocol with $N = 1$ and the table generated by the method in Sections 5.1 and 5.2. The details of the table generation algorithm used in these examples are as follows.

- **Adjusted** $\mathrm{DLap}(p)$, which is made from the discrete Laplace distribution $\mathrm{DLap}(p)$ (Section 3.2) by the method in Section 5.1.
- **Naïve Iterative Algorithm (Naïve Itr.)**, which is described in Section 5.2.

We also compare our method with an adjusted $\mathrm{DDLap}(N, p)$, which is made from the decomposed discrete Laplace distribution $\mathrm{DDLap}(N, p)$ by the method in Section 5.1. This method is essentially different from the proposed method, and it is worthwhile to compare them in the experiment. Note that since the Algorithm 2 is based on the distribution $D(p, N, w)$ (Definition 6.1), we do not compare them in the experiment.

8.3 Experimental Results

8.3.1 Runtime. We compared the computation time of our protocol (Protocol 7) with those of methods in the as-in-plaintext approach and the random table-lookup approach. We do not compare ours with that of the distributed sampling approach since its computation time is as short as sampling in plaintext.

The results of the experiment are shown in Table 1. As ε or N increases, the table length decreases. Therefore, the runtime also tends to decrease in the same manner. When the parameters were $(\varepsilon, \delta) = (0.12, 2^{-40})$ and $N = 2$, the runtime of our method was 983.1 ms, which is almost equal to the runtime of the protocol in [19]. As mentioned in Section 8, the measurements for the protocol in [19] are taken from [19], which is only slightly affected by ε.

8.3.2 Communication Cost. We compared the communication cost of our method with those of methods in the as-in-plaintext approach and the random table-lookup approach. We do not compare ours to that of the distributed sampling approach since its communication cost is as small as sampling in plaintext.

The results of the experiment are shown in Table 2. As ε or N increases, the table length decreases. Therefore, the runtime also tends to decrease in the same manner. The communication cost of our method is approximately $2N$ times the table size. Note that we implemented the table as a 16-bit signed integer array. When the

Table 1: Runtimes [ms] per single sampling average over 10 times protocols runs. N is a parameter of our protocol (Protocol 3) and $p = e^{-\varepsilon}$.

Method	ε	δ	N	Total Runtime [ms]
[19] †1	$\mathbb{Z}_{>0}$	2^{-40} †2	–	993.19
Adjusted DLap(p)	1.0	2^{-40}	1	– †3
Naïve Itr.	1.0	2^{-40}	1	– †3
Adjusted DDLap(N, p)	1.0	2^{-40}	2	– †3
Ours (Alg.2+Prot.3)	2.0	2^{-40}	2	109.7
Ours (Alg.2+Prot.3)	1.0	2^{-40}	2	108.2
Ours (Alg.2+Prot.3)	0.5	2^{-40}	2	255.5
Ours (Alg.2+Prot.3)	0.1	2^{-40}	2	1,263.4
Ours (Alg.2+Prot.3)	2.0	2^{-40}	3	10.2
Ours (Alg.2+Prot.3)	1.0	2^{-40}	3	11.2
Ours (Alg.2+Prot.3)	0.5	2^{-40}	3	12.8
Ours (Alg.2+Prot.3)	0.1	2^{-40}	3	30.6

†1 The measurements were obtained from [19], using a machine with an Intel Core i9-7960X CPU and 128GiB RAM. The values are for the optimal settings for two-party parties.
†2 Note that $2^{-40} = 10^{-12.041\cdots}$ and $10^{-10} = 2^{-33.219\cdots}$.
†3 Our implementation could not run in this setting because the table was too large.

privacy parameter was $(\varepsilon, \delta) = (0.1, 2^{-40})$, the communication cost of ours was smaller than that of the protocol in [19].

Table 2: Communication cost [MB] per single sampling. N is a parameter of our protocol (Protocol 3) and $p = e^{-\varepsilon}$.

Method	ε	δ	N	Comm. Costs [MB]
[19] †1	$\mathbb{Z}_{>0}$	2^{-40}	–	492.72
Adjusted DLap(N, p)	1.0	2^{-40}	1	89,106,061.3 (est.) †4
Naïve Itr.	1.0	2^{-40}	1	7,999,129.2 (est.) †4
Adjusted DDLap(N, p)	1.0	2^{-40}	2	116,028,843.9 (est.) †4
Ours (Alg.2+Prot.3)	2.0	2^{-40}	2	7.3
Ours (Alg.2+Prot.3)	1.0	2^{-40}	2	7.4
Ours (Alg.2+Prot.3)	0.5	2^{-40}	2	19.1
Ours (Alg.2+Prot.3)	0.1	2^{-40}	2	80.1
Ours (Alg.2+Prot.3)	2.0	2^{-40}	3	0.1
Ours (Alg.2+Prot.3)	1.0	2^{-40}	3	0.2
Ours (Alg.2+Prot.3)	0.5	2^{-40}	3	0.3
Ours (Alg.2+Prot.3)	0.1	2^{-40}	3	1.6

†1 The measurements were obtained from [19]. The values are for the optimal settings for two-party parties.
†4 Our implementation could not run in this setting because the table was too large. The communication cost was estimated by $2 \cdot$ (the number of elements in the table) $\cdot$ 16 [bit].

8.3.3 L^1 *Error.*

We compared the L^1 error (additive error, $\mathbb{E}[\,|Z|\,]$) of our method with those of methods in the distributed sampling approach. We do not compare ours to that of the as-in-plaintext approach and random table-lookup approach since its L^1 error is optimal as sampling in plaintext.

We compared the L^1 error of each method with its ratio to Δ/ε. This reference value Δ/ε is the smallest L^1 error that can be achieved with the additive mechanism for ε-DP. Note that the additive mechanism for (ε, δ)-DP ($\delta > 0$) can achieve smaller L^1 error, so the ratios may be smaller than one.

The results of the experiment are shown in Table 3. The distributed sampling method had a slightly smaller ratio of its L^1 error to the reference value than our method in the setting $N = 2$. The maximum difference was 0.08 (when the parameters were $\varepsilon = 0.1, \delta = 2^{-40}, N = 2$). In both methods, the smaller the ε, the larger the ratio of L^1 error to the reference value. In our method, the ratio of L^1 error to the reference value increased rapidly with increasing N. The method using a table generated with adjusted DDLap(N, p) had a very small ratio of its L^1 error to the reference value, but this is very unrealistic since its estimated communication cost is 116 TB (see Table 2).

Table 3: The ratio of the L^1 error ($\mathbb{E}[|Z|]$) of the noise Z sampled by each method to $1/\varepsilon$. The reference value $1/\varepsilon$ is the minimum L^1 error under $(\varepsilon, 0)$-DP and $\Delta = 1$. N is a parameter of our protocol (Protocol 3) and $p = e^{-\varepsilon}$.

Method	ε	δ	N	L^1 error$/(1/\varepsilon)$
Adjusted DDLap(N, p)	1.0	2^{-40}	2	0.8509
Dist. Sampling†5	2.0	10^{-320} †2	–	0.9870
Dist. Sampling	1.0	10^{-130}	–	1.3672
Dist. Sampling	0.5	10^{-65}	–	1.4681
Dist. Sampling	0.1	10^{-14}	–	1.4987
Ours (Alg.2+Prot.3)	2.0	2^{-40}	2	1.0513
Ours (Alg.2+Prot.3)	1.0	2^{-40}	2	1.4230
Ours (Alg.2+Prot.3)	0.5	2^{-40}	2	1.5249
Ours (Alg.2+Prot.3)	0.1	2^{-40}	2	1.5622
Ours (Alg.2+Prot.3)	2.0	2^{-40}	3	1.4991
Ours (Alg.2+Prot.3)	1.0	2^{-40}	3	1.8737
Ours (Alg.2+Prot.3)	0.5	2^{-40}	3	1.9808
Ours (Alg.2+Prot.3)	0.1	2^{-40}	3	2.0429

†5 See Section 8.2 for the detail.
†2 Note that $2^{-40} = 10^{-12.041\cdots}$ and $10^{-10} = 2^{-33.219\cdots}$.

9 Conclusion

In this paper, a secure sampling method is proposed. The proposed approach combines a table look-up with a lightweight transformation function. It is computationally less expensive than the conventional method of transforming uniform random numbers by computation only. As one embodiment of this approach, we considered a method that samples random numbers using a single table and computes their sum. An algorithm with a theoretical background is presented to generate such a table that produces random numbers that can be used to achieve differential privacy (DP) with this approach. The proposed method was compared with the existing methods in experiments, and results showed that its runtimes and communication cost are significantly smaller. On the other hand, the L^1 error (additive error) was found to be $1.4 - 1.6\times$ larger than the minimum error under $(\varepsilon, 0)$-DP. This is slightly larger than that of a simple method in the distributed sampling approach, and so our method is not superior in this aspect.

Our proposed approach is completely new, and the method presented in this paper is only a simple embodiment of it. Potential future work can be divided into two directions: 1) finding the optimal table without modifying the transformation function (in this study, we used the summing) or 2) modifying the transformation function itself. As this paper has shown, finding an analytically optimal solution for the former approach would be very challenging. The latter approach requires considering both the transformation function and the algorithm used to generate the table. We will continue this research in both approaches.

Acknowledgments

The authors used DeepL translation and Grammarly to revise the whole text to correct any typos, grammatical errors, and clearer phrasing. This paper was also corrected by an expert before submission.

References

[1] Abbas Acar, Z. Berkay Celik, Hidayet Aksu, A. Selcuk Uluagac, and Patrick McDaniel. 2017. Achieving Secure and Differentially Private Computations in Multiparty Settings. In *2017 IEEE Symposium on Privacy-Aware Computing (PAC)*. IEEE, Washington, DC, USA, 49–59. https://doi.org/10.1109/PAC.2017.12

[2] Borja Balle, Gilles Barthe, and Marco Gaboardi. 2020. Privacy Profiles and Amplification by Subsampling. *Journal of Privacy and Confidentiality* 10, 1 (Jan. 2020). https://doi.org/10.29012/jpc.726

[3] Gilles Barthe and Federico Olmedo. 2013. Beyond Differential Privacy: Composition Theorems and Relational Logic for f-Divergences between Probabilistic Programs. In *Automata, Languages, and Programming (Lecture Notes in Computer Science)*, Fedor V. Fomin, Rūsiņš Freivalds, Marta Kwiatkowska, and David Peleg (Eds.). Springer, Berlin, Heidelberg, 49–60. https://doi.org/10.1007/978-3-642-39212-2_8

[4] Amos Beimel, Kobbi Nissim, and Eran Omri. 2008. Distributed Private Data Analysis: Simultaneously Solving How and What. In *Advances in Cryptology – CRYPTO 2008 (Lecture Notes in Computer Science)*, David Wagner (Ed.). Springer, Berlin, Heidelberg, 451–468. https://doi.org/10.1007/978-3-540-85174-5_25

[5] Elette Boyle, Niv Gilboa, and Yuval Ishai. 2015. Function Secret Sharing. In *Advances in Cryptology - EUROCRYPT 2015*, Elisabeth Oswald and Marc Fischlin (Eds.). Springer, Berlin, Heidelberg, 337–367. https://doi.org/10.1007/978-3-662-46803-6_12

[6] Elette Boyle, Niv Gilboa, and Yuval Ishai. 2016. Function Secret Sharing: Improvements and Extensions. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. Association for Computing Machinery, New York, NY, USA, 1292–1303. https://doi.org/10.1145/2976749.2978429

[7] 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 (CCS'19)*. Association for Computing Machinery, London United Kingdom, 603–614. https://doi.org/10.1145/3319535.3354256

[8] T-H. Hubert Chan, Elaine Shi, and Dawn Song. 2012. Optimal Lower Bound for Differentially Private Multi-party Aggregation. In *Algorithms – ESA 2012 (Lecture Notes in Computer Science)*, Leah Epstein and Paolo Ferragina (Eds.). Springer, Berlin, Heidelberg, 277–288. https://doi.org/10.1007/978-3-642-33090-2_25

[9] Chris Clifton and Balamurugan Anandan. 2013. Challenges and Opportunities for Security with Differential Privacy. In *Information Systems Security (Lecture Notes in Computer Science)*, Aditya Bagchi and Indrakshi Ray (Eds.). Springer, Berlin, Heidelberg, 1–13. https://doi.org/10.1007/978-3-642-45204-8_1

[10] 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 (Lecture Notes in Computer Science)*, Serge Vaudenay (Ed.). Springer, Berlin, Heidelberg, 486–503. https://doi.org/10.1007/11761679_29

[11] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407. https://doi.org/10.1561/0400000042

[12] 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 (ACSAC '14)*. Association for Computing Machinery, New York, NY, USA, 316–325. https://doi.org/10.1145/2664243.2664263

[13] Reo Eriguchi, Atsunori Ichikawa, Noboru Kunihiro, and Koji Nuida. 2021. Efficient Noise Generation to Achieve Differential Privacy with Applications to Secure Multiparty Computation. In *Financial Cryptography and Data Security (Lecture Notes in Computer Science)*, Nikita Borisov and Claudia Diaz (Eds.). Springer, Berlin, Heidelberg, 271–290. https://doi.org/10.1007/978-3-662-64322-8_13

[14] David Froelicher, Patricia Egger, João Sá Sousa, Jean Louis Raisaro, Zhicong Huang, Christian Mouchet, Bryan Ford, and Jean-Pierre Hubaux. 2017. UnLynx: A Decentralized System for Privacy-Conscious Data Sharing. *Proceedings on Privacy Enhancing Technologies* 2017, 4 (Oct. 2017), 232–250. https://doi.org/10.1515/popets-2017-0047

[15] Quan Geng and Pramod Viswanath. 2013. The Optimal Mechanism in Differential Privacy. *arXiv:1212.1186 [cs]* (Oct. 2013). arXiv:1212.1186 [cs] http://arxiv.org/abs/1212.1186

[16] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. 2009. Universally Utility-Maximizing Privacy Mechanisms. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing (STOC '09)*. Association for Computing Machinery, New York, NY, USA, 351–360. https://doi.org/10.1145/1536414.1536464

[17] Slawomir Goryczka, Li Xiong, and Vaidy Sunderam. 2013. Secure Multiparty Aggregation with Differential Privacy: A Comparative Study. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops (EDBT '13)*. Association for Computing Machinery, New York, NY, USA, 155–163. https://doi.org/10.1145/2457317.2457343

[18] Seidu Inusah and Tomasz J. Kozubowski. 2006. A Discrete Analogue of the Laplace Distribution. *Journal of Statistical Planning and Inference* 136, 3 (March 2006), 1090–1102. https://doi.org/10.1016/j.jspi.2004.08.014

[19] Hannah Keller, Helen Möllering, Thomas Schneider, Oleksandr Tkachenko, and Liang Zhao. 2023. Secure Noise Sampling for DP in MPC with Finite Precision. https://eprint.iacr.org/2023/1594

[20] John B. Kioustelidis. 1986. Bounds for Positive Roots of Polynomials. *J. Comput. Appl. Math.* 16, 2 (Oct. 1986), 241–244. https://doi.org/10.1016/0377-0427(86)90096-8

[21] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. 2016. Efficient Batched Oblivious PRF with Applications to Private Set Intersection. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. Association for Computing Machinery, New York, NY, USA, 818–829. https://doi.org/10.1145/2976749.2978381

[22] Andrew McGregor, Ilya Mironov, Toniann Pitassi, Omer Reingold, Kunal Talwar, and Salil Vadhan. 2010. The Limits of Two-Party Differential Privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE, Las Vegas, NV, USA, 81–90. https://doi.org/10.1109/FOCS.2010.14

[23] Sebastian Meiser and Esfandiar Mohammadi. 2018. Tight on Budget? Tight Bounds for r-Fold Approximate Differential Privacy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*. Association for Computing Machinery, New York, NY, USA, 247–264. https://doi.org/10.1145/3243734.3243765

[24] Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil Vadhan. 2009. Computational Differential Privacy. In *Advances in Cryptology - CRYPTO 2009*, Shai Halevi (Ed.), Vol. 5677. Springer Berlin Heidelberg, Berlin, Heidelberg, 126–142. https://doi.org/10.1007/978-3-642-03356-8_8

[25] Peter Rindal and Lance Roy. 2021. Osu-Crypto/libOTe. Cryptography research at Oregon State University. https://github.com/osu-crypto/libOTe

[26] Seetha Lekshmi, V and Simi Sebastian. 2014. A Skewed Generalized Discrete Laplace Distribution. 2, 3 (March 2014), 8. https://www.ijmsi.org/Papers/Volume.2.Issue.3/K0230950102.pdf

[27] E. Shi, T.-H. Hubert Chan, E. Rieffel, Richard Chow, and D. Song. 2011. Privacy-Preserving Aggregation of Time-Series Data. In *Network and Distributed System Security Symposium*. https://www.semanticscholar.org/paper/Privacy-Preserving-Aggregation-of-Time-Series-Data-Shi-Chan/7cc53ef35f8398181bd09755ecc2fa8f52d0da1d

[28] Elaine Shi, T.-H. Hubert Chan, Eleanor Rieffel, and Dawn Song. 2017. Distributed Private Data Analysis: Lower Bounds and Practical Constructions. *ACM Transactions on Algorithms* 13, 4 (Dec. 2017), 50:1–50:38. https://doi.org/10.1145/3146549

[29] David M. Sommer, Sebastian Meiser, and Esfandiar Mohammadi. 2019. Privacy Loss Classes: The Central Limit Theorem in Differential Privacy. *Proceedings on Privacy Enhancing Technologies* (2019). https://petsymposium.org/popets/2019/popets-2019-0029.php

[30] Genqiang Wu, Yeping He, Jingzheng Wu, and Xianyao Xia. 2016. Inherit Differential Privacy in Distributed Setting: Multiparty Randomized Function

Computation. In *2016 IEEE Trustcom/BigDataSE/ISPA*. 921–928. https://doi.org/10.1109/TrustCom.2016.0157

A Proof of Theorem 6.2

Proof. For simplicity, we prove the case when $N = 2$ only, but N is not substituted for 2 as much as possible. The general case can be proved in the same way.

We will show that there exists a bound $R(w,k)(> 1)$ such that

(1) $rP_{D^{*N}}(k+1) - P_{D^{*N}}(k) \geq 0$ holds for any $r \geq R(w,k)$, and
(2) $\max_k R(w,k)$ converges to 1 as $w \to \infty$.

Here we denote $D(e^{-\varepsilon/\Delta}, N, w)$ by D. Since we already know $P_{D^{*N}}(k) = C^2 r^{w-|k|}$ for $|k| \geq (N-1)w$ as above, we consider $k = 0, \ldots, (N-1)w - 1$.

We will compute $rP_{D^{*N}}(k+1)$ and $P_{D^{*N}}(k)$. In this proof, we repeatedly use the shorthand notation $B(i) \coloneqq \binom{1/N+(w-i)-1}{w-i}$. Note that we can write $P_D(k) = B(k)r^{w-i}$.

$$\begin{aligned}
&rP_{D^{*N}}(k+1)\\
&= r^{Nw-k}B(0)B(k+1) + r^{Nw-k}\sum_{\substack{i+j=k\\0\leq i\leq k}} B(i)B(j+1)\\
&+ \binom{N}{1}\sum_{\substack{i+j=k\\k-w+1\leq i<0}} B(-i)r^{w+i}B(j+1)r^{w-j}.
\end{aligned}$$

Also,

$$\begin{aligned}
&P_{D^{*N}}(k)\\
&= r^{Nw-k}\sum_{\substack{i+j=k\\0\leq i\leq k}} B(i)B(j)\\
&+ \binom{N}{1}\sum_{\substack{i+j=k\\k-w+1\leq i<0}} r^{w+i}B(-i)r^{w-j}B(j) + Nr^k B(w-k)B(w).
\end{aligned}$$

By the definition of the binomial coefficient $B(j)$, $B(j+1) - B(j) = \frac{-1/N+1}{w-(j+1)}B(j) \geq 0$ for $0 \leq j \leq w-1 = (N-1)w-1$. Since we consider $0 \leq k \leq (N-1)w-1$, we can obtain the difference

$$\begin{aligned}
&rP_{D^{*N}}(k+1) - P_{D^{*N}}(k)\\
&= r^{Nw-k}B(0)B(k+1) && (16)\\
&+ r^{Nw-k}\sum_{\substack{0\leq i\leq k,\\i+j=k}} \frac{-1/N+1}{w-(j+1)}B(i)B(j) && (17)\\
&+ N\sum_{\substack{k-w+1\leq i\leq -1,\\i+j=k}} \frac{-1/N+1}{w-(j+1)}B(-i)B(j)r^{2w-k+2i} && (18)\\
&- Nr^w B(w-k)B(w). && (19)
\end{aligned}$$

Thus, $rP_{D^{*N}}(k+1) - P_{D^{*N}}(k)$ is a real-coefficient polynomial of r that has only one term with a negative coefficient. Especially, since the leading term coefficient is positive, the polynomial's value is positive as $r \to \infty$. To be more specific, it is positive when r is greater than any positive roots of the polynomial. According to Kioustelidis's result [20], we can evaluate the upper bound of the positive root of this polynomial:

$$R(w,k) \coloneqq \left|\frac{\text{the least coefficient (19)}}{\text{the leading coefficient (16)+(17)}}\right|^{\frac{1}{(Nw-k)-w}}.$$

Hence $rP_{D^{*N}}(k+1) - P_{D^{*N}}(k)$ holds for any $k \in \{0, \ldots, Nw-1\}$ and any $r > \max_k R(w,k)$.

Since $R(w,k)$ is monotonically decreasing in k, the maximal upper bound $R(w) \coloneqq \max_k R(w,k)$ is equal to $R(w,0)$. Specifically,

$$\begin{aligned}
R(w) &= R(w,0)\\
&= \left(\frac{NB(w)B(w)}{2B(0)B(1)}\right)^{\frac{1}{(N-1)w}}\\
&\leq \left(\frac{NB(w)B(w)}{2B(0)B(0)}\right)^{\frac{1}{(N-1)w}}\\
&= \left(\frac{N}{2}\right)^{\frac{1}{(N-1)w}}\binom{1/N+w-1}{w}^{-\frac{2}{(N-1)w}}. && (20)
\end{aligned}$$

A well-known asymptotic formula for the generalized binomial coefficient is

$$\left(\binom{1/N+w-1}{w} =\right)(-1)^w\binom{-1/N}{w} \approx \frac{w^{1/N-1}}{\Gamma(1/N)} \quad (w \to \infty)$$

gives the asymptotic evaluation of the last formula (20):

$$R(w) \leq (20) \approx \left(\frac{N\cdot\Gamma^2(1/N)}{2}\right)^{\frac{1}{(N-1)w}} w^{2/Nw} \quad (w \to \infty).$$

From the above, we conclude that $rP_{D^{*N}}(k+1) - P_{D^{*N}}(k) \geq 0$ holds for any $k \in \{0, \ldots, Nw-1\}$ and $r > 1 (= \lim_{w\to\infty}\max_k R(w,k))$ when w goes to ∞. □