Efficient Noise Generation Protocols for Differentially Private Multiparty Computation

Reo Eriguchi, Atsunori Ichikawa, Noboru Kunihiro, and Koji Nuida

Abstract—To bound information leakage in outputs of protocols, it is important to construct secure multiparty computation protocols which output differentially private values perturbed by the addition of noise. However, previous noise generation protocols have round and communication complexity growing with differential privacy budgets, or require parties to locally generate non-uniform noise, which makes it difficult to guarantee differential privacy against active adversaries. We propose three kinds of protocols for generating noise drawn from certain distributions providing differential privacy. The two of them generate noise from finite-range variants of the discrete Laplace distribution. For \((\epsilon, \delta)\)-differential privacy, they only need constant numbers of rounds independent of \(\epsilon, \delta\) while the previous protocol needs the number of rounds depending on \(\delta\). The two protocols are incomparable as they make a trade-off between round and communication complexity. Our third protocol non-interactively generates shares of noise from the binomial distribution by predistributing keys for a pseudorandom function. It achieves communication complexity independent of \(\epsilon\) or \(\delta\) for the computational analogue of \((\epsilon, \delta)\)-differential privacy while the previous protocols require communication complexity depending on \(\epsilon\). We also prove that our protocols can be extended so that they provide differential privacy in the active setting.

Index Terms—Differential privacy, secure multiparty computation, secret sharing.

1 INTRODUCTION

THERE is an increasing demand for providing statistical analysis of a large number of private data. A motivating example is performing surveys on customer information held by banks [1] or medical data stored by hospitals [2]. Secure multiparty computation (MPC) offers a solution, which enables parties to compute a function without revealing information on their data beyond an output. However, standard MPC protocols output an exact calculation result and cannot prevent an adversary from learning what follows from the result. In some applications, the exact value may contain some sensitive information on individuals. For example, exact statistics or machine learning models can be used to extract personal data [3], [4].

To deal with that problem, differentially private mechanisms [5], [6], [7] add noise drawn from an appropriate distribution to an exact calculation result and make the distributions of outputs for two “similar” inputs approximately the same. Since parties’ inputs are sensitive, it is not appropriate to assume a trusted party who aggregates their data and applies a mechanism to them. We have to realize it in a distributed setting by emulating the trusted party.

Generally, given an MPC protocol generating noise, we can emulate a mechanism based on the noise and obtain a protocol guaranteeing the same level of differential privacy [8], [9]. However, it is not straightforward to efficiently realize distributed noise generation since we have to correctly obtain noise drawn from a non-uniform distribution even if an adversary has access to and manipulates the local randomness of corrupted parties. Particularly, it cannot be solved by a simple protocol where a designated party samples and shares noise among the other parties, or where each party generates non-uniform noise \(r_i\) to perturb an outcome by the addition of \(\sum_{i \in [n]} r_i\). Furthermore, many other differentially private protocols in the local model [10], [11], [12], [13] and in the shuffled model [14], [15], [16] also require parties to locally generate non-uniform noise, which makes it difficult in the presence of active adversaries to efficiently verify that corrupted parties indeed generate their randomness from the correct distribution.

Several suitable protocols for noise generation have been proposed in the literature [5], [9], [17]. Dwork et al. [5] devise secret-sharing based MPC protocols to generate shares of noise drawn from the discrete Laplace distribution and the binomial distribution of parameter \(1/2\). However, their protocol for the discrete Laplace distribution requires the number of rounds proportional to approximately \(\log \log \delta^{-1}\) to achieve \((\epsilon, \delta)\)-differential privacy. Their protocols for the binomial distribution have to securely generate almost the same number of uniform random bits as the size of the support of the distribution. Then, the communication complexity grows linearly in \(\epsilon^{-2}\) and exponentially in the length of the fractional part when a fixed-point data type is dealt with. The protocols [9], [17] assume arithmetic operations for real numbers with infinite precision and do not rigorously analyze achievable levels of differential privacy when they are implemented under finite-precision semantics.
In summary, there is no constant-round protocol for generating discrete Laplace noise with theoretical guarantees of differential privacy, or no protocol achieving communication complexity independent of privacy budgets \( \epsilon, \delta \) for binomial noise.

1.1 Our Results

We propose MPC protocols achieving constant round complexity for the discrete Laplace distribution and achieving communication complexity independent of \( \epsilon, \delta \) for the binomial distribution. We rigorously analyze achievable levels of differential privacy of them. They can also be extended so that they provide differential privacy against active adversaries.

1.1.1 Abstraction of an Output Perturbation Framework for MPC

We first abstract a framework implicitly used in the previous protocols [5], [9], [17]. Specifically, they let parties agree on a uniformly random element \( s = (s_i)_{i \in [n]} \) of a subset \( U \subseteq S^n \) for some set \( S \), e.g., the set of all possible shares of 0 and 1. They then securely compute a deterministic function \( h(s) \), which we term a noise generator function. Assume that the distribution of \( h(s) \) provides \((\epsilon, \delta)\)-differential privacy even conditioned on \( (s_i)_{i \in T} \) being fixed where \( T \subseteq [n] \). Then, we can generally construct an \((\epsilon, \delta)\)-differentially private protocol against corruption of \( T \) from protocols securely sampling \( s \in U \) and computing \( h(s) \). An important advantage of this framework is that distributed noise generation is reduced to secure computation of deterministic functions and generation of uniformly random elements. We can cut down communication cost by devising an instantiation of \( h \) and achieve differential privacy against active adversaries by using the known actively secure protocols [18], [19].

1.1.2 Two Novel Noise Generation Protocols for the Discrete Laplace Distribution

We instantiate \( h \) with two noise generator functions whose outputs on a random input follow finite-range variants of the discrete Laplace distribution. As a result, we obtain constant-round protocols improving the round complexity of [5]. For example, while the previous protocol [5] needs 25 rounds in practical parameter settings \( \epsilon = 0.5 \) and \( \delta = 2^{-60} \) [20], our first protocol only requires 19 rounds and our second one 14 rounds for any choice of \( \epsilon, \delta \). Furthermore, the probability that protocols fail to generate noise is negligible in our first protocol and 0 in our second one while it is non-negligible in [5]. It means that a single protocol execution would be sufficient and our protocols can achieve better utility than [5]. Although our protocols have higher communication complexity than [5], it does not weaken our advantages in round complexity. Actually, our protocols reduce the estimated total running time of [5] by around 20% in the above parameter setting. As for a comparison between our protocols, the first one has lower communication cost but needs more rounds of interaction than the second one. As a result, the first one is faster for small \( \epsilon \) while the second one is faster for large \( \epsilon \). The second one has another advantage of high utility coming from the failure probability being zero and hence is even applicable to \( \epsilon \) such that the first one is not. A more detailed comparison is given in Section 6.1.

1.1.3 A Novel Noise Generation Protocol for the Binomial Distribution

We also instantiate \( h \) with a function whose output on a random input follows the binomial distribution of size \( N \) and parameter \( 1/2 \). Based on it, we propose a protocol that enables parties to locally compute shares of binomial noise by predistributing keys for a pseudorandom function. Technically, it is a modification of pseudorandom secret sharing [21]. Our protocol reduces the previous communication complexity linear in \( N \) [5] at the cost of predistributing keys. Furthermore, once the keys are distributed, parties can non-interactively generate shares of polynomially many binomial samples, which amortizes the communication cost in the setup. As a drawback, our protocol only satisfies the computational analogue of differential privacy [22]. Moreover, the error bound of our protocol is \( O(n^{1/2}) \) times larger than that of [5], where \( t \) is a corruption threshold. Nevertheless, our protocol works well in the client-server model, in which \( n \) corresponds to the number of servers and is typically small, e.g., \( n = 3 \). We compare the estimated total running time of ours with [5], and figure out that it is up to 23 times faster when generating 100 binomial noises for \( \epsilon = 0.5 \) and \( \delta = 2^{-60} \). A more detailed comparison is given in Section 6.2.

1.1.4 Extension to Active Security

To the best of our knowledge, we are first to formalize unconditional security of MPC achieving differential privacy against active adversaries in the ideal-real-world paradigm. Although the authors of [17], [23] extend their protocols to the active setting, their security depends on cryptographic primitives such as computational verifiable secret sharing and zero-knowledge proof, and they do not discuss unconditional security. We prove that our general framework provides unconditionally secure protocols against active adversaries given actively secure protocols computing deterministic functions and generating uniformly random elements. In particular, our above protocols can be extended to the active setting by replacing the underlying MPC primitives accordingly.

For concrete efficiency, we estimate the running times for our actively secure protocols to generate samples from the discrete Laplace and binomial distributions. Calculating additional communication bits and rounds of interaction, we figure out that our actively secure protocols are about 4–16 times slower than the passively secure counterparts. Based on the experimental results [24], we also see that our protocols can generate noises in the presence of active adversaries for a reasonable time even if local computation time is taken into account. A more detailed performance evaluation is given in Section 7.

1.2 Related Work

The authors of [9], [17], [23] construct protocols for generating noise drawn from the Gaussian and Laplace distributions by making black-box use of MPC protocols for
operations over real numbers. However, since secure com-
putation over real numbers is more costly than on integers
(e.g., more than 140 rounds are necessary for computing
logarithms in the floating-point representation [25]), our
protocols outperform the protocols [9], [17] in efficiency
when an integer data type is dealt with. Furthermore, they
lack a rigorous analysis of the impact of the fine-precision
implementations on differential privacy, which is undesir-
able since differentially private mechanisms are vulnerable
to the exact computations [26], [27].

There are many differentially private mechanisms in the
local model [10], [11], [12], [13], in which every party locally
randomizes his private input and sends it to a designated
party, who then computes a function on the noisy data.
However, as mentioned above, local-model mechanisms
require parties to locally generate non-uniform noise and
it is difficult to efficiently verify the correctness of local
randomness of corrupted parties in the presence of active
adversaries. In addition, since a careful analysis of accumu-
lated noise is needed, mechanisms are proposed only for
a limited class of functions such as aggregate-sum queries
\( \sum_{i \in [n]} f(x_i) \). Other mechanisms in the shuffled model [14],
[15], [16] also involve non-uniform noise generation and are
applicable only for simple functions. We refer the reader to
[28] for a survey.

The authors of [29] propose a method to generate many
biased bits improving the efficiency of [5]. However, the
method is based on oblivious data structures and is not
directly applicable to secret-sharing based MPC protocols.

1.3 Publication Note

The preliminary version appeared in the proceedings of
Financial Cryptography and Data Security 2021 [30]. The
current version provides a novel protocol for the discrete
Laplace distribution improving the communication com-
plexity of [30]. In addition, this paper proves that our
protocols can be extended to the active setting.

2 Preliminaries

2.1 Notations

For \( n \in \mathbb{N} \), \([n]\) denotes \{\( z \in \mathbb{Z} : 1 \leq z \leq n \)\} and \([0..n]\) denotes
\{\( z \in \mathbb{Z} : 0 \leq z \leq n - 1 \)\}. We denote by \( e^x \) or
\( \exp(x) \) the exponential function of \( x \in \mathbb{R} \). We assume that \( q \)
is a sufficiently large odd prime and identify the prime field
\( \mathbb{Z}_q \) of size \( q \) with \{\( z \in \mathbb{Z} : q/2 < z < q/2 \)\}. A function
\( f : \mathbb{N} \rightarrow \mathbb{R} \) is negligible in \( \lambda \) and denoted by \( \negl(\lambda) \) if for
any \( c > 0 \), there exists \( N \in \mathbb{N} \) such that \( 0 \leq f(\lambda) < 1/\lambda^c \)
for any \( \lambda \geq N \). For \( T \subseteq [n] \) and a vector \( x \in \mathbb{R}^n \), we denote by
\( x_T \in S_T^{[n]} \) the sub-vector obtained by restricting the indices
to \( T \).

Let \( X \) and \( Y \) be two random variables with range \( U \).
We define the statistical distance \( \text{SD}(X,Y) \) between \( X \)
and \( Y \) as \( \text{SD}(X,Y) = (1/2) \sum_{u \in U} | \Pr[X = u] - \Pr[Y = u] | \).
It holds that \( \text{SD}(X_1,X_2),(Y_1,Y_2) \leq \text{SD}(X_1,Y_1) + \text{SD}(X_2,Y_2) \)
for random variables \( X_1, X_2, Y_1, Y_2 \) and that
\( \text{SD}(F(X),F(Y)) \leq \text{SD}(X,Y) \) for any randomized function
\( F \). We denote by \( X \sim D \) if \( X \) is distributed according to
a probability distribution \( D \). We also write \( s \sim D \) if \( s \) is a
value sampled from \( D \). We denote by \( \text{Uni}(S) \) the uniform
distribution over a finite set \( S \).

2.2 Secure Multiparty Computation

Assume that there are \( n \) parties holding their private inputs
\( x_i, i \in [n] \) from a finite set \( D \). Let \( g \) be a deterministic
function to compute on their inputs. Let \( \Pi \) be a protocol.
For a subset \( T \subseteq [n] \), we define \( \text{View}_{\Pi}^T(x) \) as the joint view of
the parties in \( T \) during the execution of \( \Pi \) with inputs
\( x \). We say that \( \Pi \) is an MPC protocol \( t \)-securely computing
\( g \) if (1) for any subset \( T \subseteq [n] \) of size \( t \) and any pair of
inputs \( x = (x_i)_{i \in [n]} \) and \( y = (y_i)_{i \in [n]} \), the distributions
of \( \text{View}_{\Pi}^T(x) \) and \( \text{View}_{\Pi}^T(y) \) are identical as long as \( \psi_T = y_T \)
and \( g(x) = g(y) \) and (2) for every \( x \) and \( i \in [n] \), \( g(x) \) is
included in \( \text{View}_{\Pi}^I(x) \). See [31] for a more general case of \( n \)-
input \( n \)-output randomized functionalities in the presence
of an active adversary.

2.2.1 Secret Sharing

Given a secret \( a \in \mathbb{Z}_q \), the \((t,n)\)-Shamir secret sharing
scheme [32] generates a random polynomial \( p \) of degree
at most \( t \) such that \( p(0) = a \) and outputs \( [a]_i = p(i) \)
as the \( i \)-th share. We simply write \( [a] \) if the index \( i \) is clear
from the context. If \( t < n/2 \), there exists a protocol \( \text{MULT} \)
t-securely computes \([ab] \) from \([a] \) and \([b] \) [33]. If \( t < n/3 \), it is
possible to \( t \)-securely realize it even in the presence of active
adversaries. We measure the communication complexity of
an MPC protocol by the number of invocations of \( \text{MULT} \)
and its round complexity by the number of sequential
invocations of \( \text{MULT} \).

2.2.2 Pseudorandom Secret Sharing

Pseudorandom secret sharing [21] allows parties to non-
interactively generate shares of a pseudorandom number by
predistributing keys for a pseudorandom function. Techni-
cally, a pseudorandom function [34] with length parameters
\( s, \ell : \mathbb{N} \rightarrow \mathbb{N} \) is a collection of functions \( \{\psi_r : \{0,1\}^s \rightarrow \{0,1\}^{\ell(\lambda)}\}_{r \in \{0,1\}^*} \), where \( \{0,1\}^* \)
denotes the set of all the
bit strings of arbitrary length and \( \lambda \) is the bit length of \( r \),
such that (efficient evaluation) \( \psi_r(a) \) can be computed in
polynomial time from \( r \in \{0,1\}^\lambda \) and \( a \in \{0,1\}^{s(\lambda)} \)
and (pseudorandomness) for every probabilistic polynomial-
time (PPT) oracle machine \( M \) which has access to out-
puts of a function on inputs of its choice, it holds that
\( |\Pr[M^{\psi_{r_0}(1^{\lambda})} = 1] - \Pr[M^{\psi_{r_1}(1^{\lambda})} = 1]| = \negl(\lambda) \),
where \( U_\lambda \sim \text{Unif}([0,1]^{\lambda}) \) and \( F_\lambda \) is a uniformly
selected map from \( \{0,1\}^{s(\lambda)} \) to \( \{0,1\}^{\ell(\lambda)} \). There is a provably secure pseudorandom function
\( \psi_r : \{0,1\}^s \rightarrow \{0,1\}^\ell \) \( r \in \{0,1\}^\lambda \) with
\( \lambda = 161 \times 160 \) and \( s = \ell = 128 \) [35]. One can also use the
AES encryption with \( \lambda = 128 \) and \( s = \ell = 128 \).

Let \( A = \{A \subseteq [n] : |A| = n - t\} \) and \( A_i = \{A \in A : i \in A\} \) for \( i \in [n] \). For \( A \in A \), let \( f_A \in \mathbb{Z}_q[X] \) be the unique polynomial such that
\( f_A(0) = 1 \). The party \( A \) locally computes \( v_i = \sum_{A \in A_i} \psi_{r_i}(a) f_A(i) \)
from his keys \( (r_A)_{A \in A_i} \) and a public input \( a \in \{0,1\}^{s(\lambda)} \)
by embedding \( \{0,1\}^{\ell(\lambda)} \) into \( \mathbb{Z}_q \). It can be verified that \( (v_i)_{i \in [n]} \)
are consistent shares of the \((t,n)\)-Shamir secret sharing
scheme for a pseudorandom number \( \sum_{A \in A_i} \psi_{r_i}(a) \) \( a \in \mathbb{Z}_q \).
Note that the setup assumption can be removed by using
any protocol for generating random shares of the replicated
secret sharing scheme (e.g. [36]) since \( (r_A)_{A \in A_i} \) is the \( i \)-th
suppose the following definition of MPC protocols satisfy:

adversaries who see internal states of corrupted parties but compute (x, y), the distributions of View_\Pi^t(x) and View_\Pi^t(y) are (\epsilon,\delta)-DP close.

- (\alpha,\beta)-Utility: for every input x, it holds that Pr[[\Pi(x) = g(x)] \leq \alpha] \geq 1 - \beta, where \Pi(x) is an output of \Pi on input x.

We can define a computational analogue of (t,\epsilon,\delta)-differentially private protocols by considering (\epsilon,\delta)-DP closeness in the computational setting. Note that we do not relax t-privacy in the computational setting.

In the above definition, we require that differential privacy holds for pairs of T-neighboring inputs for a set T of corrupted parties, instead of \emptyset-neighboring ones. This is because what we need to guarantee is that an adversary cannot tell if at most one honest party i \notin T changes his input in the sense of differential privacy. For neighboring inputs differing on an entry whose index is in T, the adversary can trivially break differential privacy by viewing the corrupted parties’ inputs. We note that this definition is adopted in the literature [8], [9].

2.4 Additive Noise Mechanisms

A typical technique to achieve differential privacy is adding controlled noise to an outcome. Let g : D^n \rightarrow \mathbb{R}. We define the sensitivity \Delta of g as \Delta = \max\{|\{g(x) - g(y)\} : x \neq y\} and the range R of g as R = \max_{x \in D^n} |g(x)|.

For 0 < p < 1, the discrete Laplace distribution DL(p) is the distribution of L defined as Pr[L = k] = p^{k}(1-p)/(1+p) for k \in Z [40]. The functionality g(\cdot) + L satisfies (\epsilon,0)-differential privacy and (\alpha,\beta)-utility if p = exp(-\epsilon/\Delta) and \alpha = (\Delta/\epsilon)\ln(1/\beta) [41]. Since we focus on integers on a finite interval, we consider finite-range variants of DL(p), which are also shown to provide differential privacy in Section 4.

The binomial distribution is also used in the literature [7]. For \ell, M \in \mathbb{N}, let Y ~ Bin(N,1/2), i.e., Pr[Y = k] = \binom{N}{k}2^{-N} for k = 0,1,\ldots,N and define NBin(N,M) as the distribution of (1/M)(Y - N/2). Suppose that N,M satisfy N/4 \geq \max\{23\log(10/\delta),\Delta M\} for some \delta > 0. For Z ~ NBin(N,M), the functionality M_y^{N,M}(\cdot) = g(\cdot) + Z is (\epsilon,\delta)-differentially private if

\[\epsilon \geq \epsilon(\delta,N,M,\Delta) := \Delta \left( c_1(\delta) \frac{M}{\sqrt{N}} + c_2(\delta) \frac{M}{N} \right), \] (1)

where c_1(\delta) = O(\sqrt{\log \delta^{-1}}) and c_2(\delta) = O((\log \delta^{-1})^2). For any input x \in D^n, Hoeffding’s inequality implies that M^{N,M}_y satisfies (\alpha,\beta)-utility for g if \beta = 2\exp(-2M^2\alpha^2/N), i.e., \alpha = \sqrt{(N/2M^2)\ln(2/\beta)}.

3 Abstraction of an Output Perturbation Framework for MPC

We abstract out the framework implicitly used in the previous protocols for generating shares of noise drawn from non-uniform distributions [5], [9], [17]. They first get parties agree on an element uniformly selected from a certain set and then securely compute a deterministic function h on the element. Technically, let S be a finite set and U be a subset of S^n. For T \subseteq [n], we define U_T = \{a \in S^{|T|} : a = u_T for some u \in U\}.

2.3 Differential Privacy

Two random variables X, Y are said to be (\epsilon,\delta)-DP close if for every distinguisher D, it holds that

Pr[D(X) = 1] \leq e^\epsilon \Pr[D(Y) = 1] + \delta.

Analogously, we say that they are computationally (\epsilon,\delta)-DP close [22] if for every PPT distinguisher D, it holds that

Pr[D(X) = 1] \leq e^\epsilon \Pr[D(Y) = 1] + \delta + \negl(\lambda),

where \lambda is a security parameter. Two vectors x = (x_1,\ldots,x_n), y = (y_1,\ldots,y_n) \in D^n are called T-neighboring if there is exactly one index i \in [n] \setminus T such that x_i \neq y_i and simply called neighboring if they are \emptyset-neighboring. We say that a randomized functionality M with domain D^n is (resp. computationally) (\epsilon,\delta)-differentially private if M(x) and M(y) are (resp. computationally) (\epsilon,\delta)-DP close for all neighboring vectors x, y \in D^n. When the output domain of M is included in \mathbb{R}, we say that M satisfies (\alpha,\beta)-utility for a function g : D^n \rightarrow \mathbb{R} if for every input x \in D^n, it holds that Pr[|M(x) - g(x)| \leq \alpha] \geq 1 - \beta. We only consider passive adversaries who see internal states of corrupted parties but do not deviate from protocols until Section 7. For now, we suppose the following definition of MPC protocols satisfying differential privacy against passive adversaries.

Definition 1. An MPC protocol \Pi is called a (t,\epsilon,\delta)-differentially private protocol for computing g with (\alpha,\beta)-utility if the following holds:

- t-Privacy: for every subset T of size at most t and for any pair of inputs x and y with x_T = y_T and g(x) = g(y), the distributions of View_\Pi^t(x) and View_\Pi^t(y) are identical.

- (\epsilon,\delta)-Differential privacy: for every subset T of size at most t and for any pair of T-neighboring inputs x and y, the distributions of View_\Pi^t(x) and View_\Pi^t(y) are (\epsilon,\delta)-DP close.
Assumption. The parties receive correlated randomness $s \sim \text{Uni}(U)$.

Input. $x \in D^n$.

Output. $z = g(x) + h(s) \in \mathbb{Z}_q$.

Protocol.
1) $[g(x)] = \Pi_y(x)$.
2) $[h(s)] = \Pi_b(s)$.
3) $[z] = [g(x)] + [h(s)]$.
4) Reconstruct and output $z$.

Fig. 1: A $(t; \epsilon, \delta)$-differentially private protocol for computing $g$.

**Proposition 1.** Let $g : D^n \rightarrow \mathbb{Z}_q$ and $h : U \rightarrow \mathbb{Z}_q$ be deterministic functions. Let $\Pi_y$ (resp. $\Pi_b$) be a protocol which takes $x \in D^n$ (resp. $s \in U$) as input and $t$-securely computes $([g(x)])_{i \in [n]}$ (resp. $([h(s)])_{i \in [n]}$). Assume that for any subset $T$ of size $t$, any pair of $T$-neighboring vectors $x, y \in D^n$, and any $a \in U_T$, two distributions $(x_T, s_T, g(x) + h(s)), (y_T, s_T, g(y) + h(s))$ conditioned on $s_T = a$ are $(\epsilon, \delta)$-DP close over the randomness of $s \sim \text{Uni}(U)$. Furthermore, assume that $\Pr_{x \sim \text{Uni}(U)}([h(s)] \leq a) \geq 1 - \beta$ for $\alpha$ and $\beta$. Then, the protocol $\Pi$ described in Fig. 1 (assuming certain correlated randomness) is a $(t; \epsilon, \delta)$-differentially private protocol for computing $g$ with $(\alpha, \beta)$-utility.

We call $h$ in Proposition 1 a noise generator function. We emphasize that the level of differential privacy and utility that the resultant protocol achieves only depends on the property of $h$. Therefore, a single noise generator function $h$ can give the same level of differential privacy and utility to MPC protocols for many different functions.

**Proof of Proposition 1:** Let $T \subseteq [n]$ with $|T| \leq t$. For $x \in D^n$ and fixed $s \in U$, let $v_T^H(x; s)$ denote the joint view of $T$ when $\Pi$ is executed on $x$ and $s$. Note that the distribution of $\text{View}_T^H(x)$ is the same as that of $v_T^H(x; s)$ induced by $s \sim \text{Uni}(U)$. From the security of $\Pi_y$ and $\Pi_b$, we have a simulator $\text{Sim}_T$ such that the distribution of $\text{View}_T^H(x)$ is the same as that of $\text{Sim}_T(x_T, s_T, g(x) + h(s))$ for fixed $x$ and $s$.

As for $t$-privacy, let $x, y$ be two inputs such that $x_T = y_T$ and $g(x) = g(y)$. Then, it holds that for any set of transcripts $O$,

$$\Pr[\text{View}_T^H(x) \in O] = \Pr_{s \sim \text{Uni}(U)}[v_T^H(x; s) \in O] = \sum_{u \in U} \Pr[s = u] \cdot \Pr[v_T^H(x; u) \in O] = \sum_{u \in U} \Pr[s = u] \cdot \Pr[\text{Sim}_T(x_T, u_T, g(x) + h(u)) \in O] = \sum_{u \in U} \Pr[s = u] \cdot \Pr[\text{Sim}_T(y_T, u_T, g(y) + h(u)) \in O] = \Pr[\text{View}_T^H(y) \in O].$$

As for $(\epsilon, \delta)$-differential privacy, let $x, y$ be $T$-neighboring inputs. From the assumption, we have that for any set of outcomes $O'$ and any $a \in U_T$,

$$\Pr[(x_T, s_T, g(x) + h(s)) \in O' \mid s_T = a] \leq e^\epsilon \Pr[(y_T, s_T, g(y) + h(s)) \in O' \mid s_T = a] + \delta.$$  

Since differential privacy is immune to post-processing, it also holds that for any set of transcripts $O$ and any $a \in U_T$,

$$\Pr[\text{Sim}_T(x_T, s_T, g(x) + h(s)) \in O \mid s_T = a] \leq e^\epsilon \Pr[\text{Sim}_T(y_T, s_T, g(y) + h(s)) \in O \mid s_T = a] + \delta.$$  

Since $v_T^H(x; s)$ and $\text{Sim}_T(x_T, s_T, g(x) + h(s))$ are identically distributed for fixed $x$ and $s$, we obtain that for any set of transcripts $O$ and any $a \in U_T$,

$$\Pr[v_T^H(x; s) \in O \mid s_T = a] \leq e^\epsilon \Pr[v_T^H(y; s) \in O \mid s_T = a] + \delta.$$  

Thus, taking the probability-weighted sum over all $a \in U_T$ gives $\Pr[\text{View}_T^H(x) \in O] \leq e^\epsilon \Pr[\text{View}_T^H(y) \in O] + \delta$ for any set of transcripts $O$.

Finally, $(\alpha, \beta)$-utility easily follows.

It can be seen that $\Pi$ is a computationally $(t; \epsilon, \delta)$-differentially private protocol if $h$ only provides $(\epsilon, \delta)$-differential privacy in the computational setting.

If generating a random input to $h$ is equivalent to obtaining shares for uniform random values or bits, we can construct differentially private protocols without the setup assumption by running protocols for generating random shares. This is actually the case in all of the previous protocols [5], [9], [17] and our instantiations given later.

As a corollary of Proposition 1, if the input domain of $h$ is the set of all the possible shares of $m$ secret bits and the output of $h$ essentially depends on the $m$ bits, then it becomes easier to find out the level of differential privacy that $h$ can provide.

**Corollary 1.** Let $g$ and $\Pi_3$ be as in Proposition 1. Let $h_0 : \{0, 1\}^m \rightarrow \mathbb{Z}_q$ be a deterministic function and $\Pi_0$ be a protocol which takes shares of $m$ bits $b = (b_1, \ldots, b_m)$ as input and $t$-securely computes $([h_0(b)])_{i \in [n]}$. Assume that, for any subset $T$ of size $t$ and any pair of $T$-neighboring vectors $x, y \in D^n$, two distributions $(x_T, g(x) + h_0(b)); (y_T, g(y) + h_0(b))$ are $(\epsilon, \delta)$-DP close over the randomness of $s \sim \text{Uni}(\{0, 1\}^m)$. Furthermore, assume that $\Pr_{b \sim \text{Uni}(\{0, 1\}^m)}[h_0(b)] \leq \alpha \geq 1 - \beta$ for $\alpha$ and $\beta$. Then, there exists a $(t; \epsilon, \delta)$-differentially private protocol for computing $g$ with $(\alpha, \beta)$-utility.

**Proof:** Define $U$ as

$$U = \{(b_1, \ldots, b_m) \in \{0, 1\}^m : (b_1, \ldots, b_m) \in \{0, 1\}^m \}$$

and $h : U \rightarrow \mathbb{Z}_q$ as $h([b_1], \ldots, [b_m]) = h_0(b_1, \ldots, b_m)$. Then, $\Pi_0$ $t$-securely computes $h(s)$ from $s \in U$. Furthermore, the setup assumption of $\Pi$ can be removed since generating correlated randomness $s \sim \text{Uni}(U)$ can be done by $\Sigma_{\text{bit}}$. Due to the security of $\Sigma_{\text{bit}}$, the conditional distribution of $h(s)$ given $s_T$ is the same as $h_0(b)$ for $b \sim \text{Uni}(\{0, 1\}^m)$ as long as $|T| \leq t$. Therefore, for any subset $T$ of size $t$ and any pair of $T$-neighboring vectors $x, y \in D^n$, two conditional distributions $(x_T, s_T, g(x) + h(s)); (y_T, s_T, g(y) + h(s))$ given $s_T$ are $(\epsilon, \delta)$-DP close for $s \sim \text{Uni}(U)$. The statements then follow from Proposition 1.

\end{proof}
The protocol provided in the above proof can also be viewed as a protocol securely realizing the randomized functionality which, on input \( x \in D^n \), outputs \( g(x) + h_0(b) \) for \( b \sim \text{Unif}(\{0, 1\}^m) \). Thus, the same statement is obtained by applying the general result [8], [9] which states that protocols securely realizing differentially private mechanisms achieve the same level of differential privacy. Nevertheless, an important feature of Corollary 1 is that it reduces generation of noise for differential privacy to generation of uniformly random bits and secure computation of the deterministic function \( h_0 \).

4 Two Novel Noise Generation Protocols for the Discrete Laplace Distribution

To begin with, we present simple sufficient conditions for a probability distribution over a finite interval of \( \mathbb{Z} \) to provide differential privacy.

Lemma 1. Let \( N \in \mathbb{N} \) and \( g : D^n \rightarrow \mathbb{Z}_q \) be a function with sensitivity \( \Delta \) and range \( R \). Assume that \( g/2 > N + R \). Let \( X \) be a random variable on \( \mathbb{Z}_q \) such that

1) its support is \( \{ z \in \mathbb{Z} : -N < z < N \} \);
2) \( \Pr [X = z] = \Pr [X = -z] \) for \( 0 \leq z < N \);
3) \( \Pr [X = z] \leq \epsilon \Pr [X = z'] \) for \( 0 \leq z' \leq z + \Delta \);
4) \( \Pr [N - \Delta < X < N] \leq \delta \).

Then, \( M(\cdot) = g(\cdot) + X \) is \((\epsilon, \delta)\)-differentially private.

Proof: Let \( S \subseteq \mathbb{Z}_q \) and \( x, y \in D^n \) be neighboring vectors. Assume that \( g(y) \leq g(x) \). Define \( S_1, S_2 \subseteq S \) as \( S_1 = \{ z \in S : g(x) - N \leq z \leq g(y) + N \} \) and \( S_2 = S \setminus S_1 \). Since \( g(x) \leq g(y) + \Delta \), we have

\[
\Pr[M(x) \in S] = \Pr[M(x) \in S_1] + \Pr[M(x) \in S_2] \\
\leq \sum_{z \in S_1} \Pr[X = z - g(x)] + \Pr[N - \Delta \leq X < N] \\
\leq \sum_{z \in S_1} e^\epsilon \Pr[X = z - g(y)] + \delta \\
\leq e^\epsilon \Pr[M(y) \in S] + \delta.
\]

A similar argument works when \( g(y) \geq g(x) \).

Furthermore, \((\alpha, \beta + \delta_0)\)-utility follows from

\[
\Pr[|M'(x) - g(x)| \leq \alpha] = \Pr[|X'| \leq \alpha] \\
\geq \Pr[|X| \leq \alpha] - \delta_0 \\
\geq 1 - \beta - \delta_0.
\]

\[\square\]

Lemma 3. Let \( \perp \) be a special symbol not \in \( \mathbb{Z}_q \). Let \( X \) be a random variable on \( \mathbb{Z}_q \cup \{ \perp \} \) and \( \delta_0 = \Pr[X = \perp] \). Let \( X' \) be the random variable associated with the conditional distribution of \( X \) given \( X \neq \perp \). Assume that \( M'(\cdot) = g(\cdot) + X' \) satisfies \((\epsilon, \delta)\)-differential privacy and \((\alpha, \beta)\)-utility for \( g \). Then, \( M(\cdot) = g(\cdot) + X \) satisfies \((\epsilon, \delta)\)-differential privacy and \((\alpha, \beta + \delta_0)\)-utility for \( g \), where for any input \( x \), we define \( M(x) = \perp \) if \( X = \perp \).

Proof: Let \( S \subseteq \mathbb{Z}_q \cup \{ \perp \} \) and \( x, y \) be neighboring vectors. Then, it holds that

\[
\Pr[M(x) \in S] = \Pr[M(x) \neq \perp] \cdot \Pr[M(x) \in S | M(x) \neq \perp] \\
+ \Pr[M(x) = \perp] \cdot \Pr[M(x) \in S | M(x) = \perp] \\
\leq (1 - \delta_0)(e^\epsilon \Pr[M(y) \in S | M(y) \neq \perp] + \delta_0)I_S(\perp) \\
= e^\epsilon \Pr[M(y) \in S] - \delta_0I_S(\perp) + (1 - \delta_0)\delta_0I_S(\perp) \\
= e^\epsilon \Pr[M(y) \in S] + (1 - \delta_0)\delta_0I_S(\perp) \\
\leq e^\epsilon \Pr[M(y) \in S] + \delta,
\]

where \( I_S(\perp) \) is 1 if \( \perp \in S \) and 0 otherwise.

4.1 The First Protocol

As a building block, we present a constant-round protocol for generating noise according to the Bernoulli distribution from many uniformly random bits. Let \( \text{Ber}(\alpha) \) be the distribution over \( \{0, 1\} \) such that \( \Pr[X = b] = \alpha^b(1 - \alpha)^{1-b} \). For \( \ell \in [d], \) let \( \alpha_\ell \) be the \( \ell \)-th most significant bit in the binary expansion of \( \alpha \). Define \( B_{\alpha,d} : \{0, 1\}^d \rightarrow \mathbb{Z}_q \) as \( B_{\alpha,d}(b_1, \ldots, b_d) = 1 - b_j \) for \( j = \min(\ell \in [d] : b_\ell \neq \alpha_\ell) \) (we set \( j = d+1 \) if there is no such index). Equivalently, it outputs 1 if \( \sum_{\ell \in [d]} b_\ell 2^{-\ell} \leq \alpha \) and otherwise, outputs 0. The statistical distance between \( \text{Ber}(\alpha) \) and \( B_{\alpha,d}(b) \) for \( b \sim \text{Unif}(\{0, 1\}^d) \) is at most \( 2^{-d} \). The protocol \( \text{BER}_{\alpha,d} \) described in Fig. 2 \( \ell \)-securely computes \( b_{\alpha,d}(b) \) from \( b \in \{0, 1\}^d \). Note that at Step 1, \( [b_\ell \oplus \alpha_\ell] \) can be locally computed from \( [b_\ell] \) since \( \alpha_\ell \) is a public parameter.

Now, we construct a protocol for generating noise drawn from a finite-range variant of \( DL(p) \). Let \( 0 < p < 1 \) and \( N \in \mathbb{N} \). Let \( G \sim \text{Geo}(p) \) be the truncated geometric distribution defined by \( \Pr[G = x] = C(1-p)^{x-1}p^x \) for \( x \in [0, N] \), where \( C = (\sum_{x \in [0..N]}(1-p)p^x) \) is a normalizing constant. Define \( \text{FDL}_1(p, N) \) as the distribution
Input. \([b_i]_\ell \in [d]\), where \(b_i \in \{0, 1\}\).
Output. \([B_{\alpha,d}(b_1, \ldots, b_d)]\).

Protocol.
1) \([c_\ell] = [b_\ell + 4\alpha]\) for \(\ell \in [d]\).
2) \(([c_\ell])_{\ell \in [d]} = \text{Pr}^N(\{c_k\}_{k \in [d]}),
3) \([f_\ell] = [c_{\ell-1}] - [c_{\ell-1}]\) for \(\ell \in [d]\), where
4) \([g] = \text{IP}_d([f_\ell])_{\ell \in [d]}([b_\ell])_{\ell \in [d]}\).
5) Output \(1 - [g]\).

Fig. 2: The protocol BER(\(\alpha,d\)) computing \(B_{\alpha,d}\)

of \(X_1 - X_2\) for independent \(X_1, X_2 \sim \text{Geo}(p, N)\). Explicitly, the distribution of \(X' \sim \text{FDL}_1(p, N)\) is

\[
\Pr[X' = x] = \frac{(1 - p)p^{x}}{\left(1 + p\right)(1 - p^{N})}, \quad N < x < N.
\]

The motivation behind this definition is a mathematical fact that the distribution of \(g_1 - g_2\) is DL(p) for two independent samples \(g_1, g_2\) drawn from the geometric distribution [40].

To make it applicable to a wide range of privacy budgets, for \(M \in \mathbb{N}\) with \(M < N\), we define \(\text{FDL}_1(p, N, M)\) as the probability distribution of \(X' \sim \text{FDL}_1(p, N)\) conditioned on \(|X'| \leq M\). That is, \(X \sim \text{FDL}_1(p, N, M)\) has the probability distribution

\[
\Pr[X = x] = \frac{\Pr[X' = x]}{\Pr[|X'| \leq M]}, \quad -M \leq x \leq M,
\]

where \(X' \sim \text{FDL}_1(p, N)\).

**Proposition 2.** Let \(X \sim \text{FDL}_1(p, N, M)\). The functionality \(\mathcal{M}(\cdot) = g(\cdot) + X\) is \((\epsilon, \delta)\)-differentially private for a function \(g\) with sensitivity \(\Delta\) and range \(R\) if \(p, N,\) and \(M\) satisfies that \(N + R < q/2\),

\[
p - \frac{2M(N - M)}{1 - p^{2(N - M)}} \leq \epsilon',
\]

and

\[
\frac{M - \epsilon}{1 - p} \leq \delta.
\]

Furthermore, \(\mathcal{M}\) satisfies \((\alpha, \beta)\)-utility for \(g\) if \(\alpha = (\Delta/\epsilon) \ln(2/((\beta(1 - p)(1 - p^{2N})))).\)

**Proof:** The support of \(\text{FDL}_1(p, N, M)\) is \(\{z \in \mathbb{Z} : |z| < M\}\). For any \(z, z'\) with \(0 \leq z \leq z' < z + \Delta\), we have

\[
\Pr[X = z] \leq \frac{1 - p}{1 - p^{2(N - z)}} \leq \frac{1 - p^{2(N - z)}}{1 - p^{2(N - z - \Delta)}}
\]

since \((1 - p^{2(N - z)})/(1 - p^{2(N - z - \Delta)})\) is monotonically increasing with respect to \(z\).

To show \(\Pr[|X'| \leq \Delta] \leq \delta\), we observe that for \(X' \sim \text{FDL}_1(p, N)\) and any \(0 \leq m \leq M\),

\[
\Pr[|X'| \leq M] = \frac{1 - p}{(1 + p)(1 - p^{N})} \left(\sum_{|z| \leq m} p^{2(N - z)} - \sum_{|z| \leq m} p^{2N(z)}\right)
\]

since \((1 - p^{2(N - z)})/(1 - p^{2(N - z - \Delta)})\) is monotonically increasing with respect to \(z\).

To show \(\Pr[|X'| \leq \Delta] \leq \delta\), we observe that for \(X' \sim \text{FDL}_1(p, N)\) and any \(0 \leq m \leq M\),

\[
\Pr[|X'| \leq M] = \frac{1 - p}{(1 + p)(1 - p^{N})} \left(\sum_{|z| \leq m} p^{2(N - z)} - \sum_{|z| \leq m} p^{2N(z)}\right)
\]

\[
= \frac{1 + p + p^{2N} + p^{2N + 1} - 2p^{m + 1} - 2p^{2N - m}}{(1 + p)(1 - p^{N})}.
\]

Thus, we have that

\[
\Pr[|X'| \leq M] = \frac{1}{2} \left(1 - \frac{\Pr[|X'| \leq M - \Delta - 1]}{\Pr[|X'| \leq M]}\right)
\]

\[
= \left(p^{M - \Delta} - (1 - p^{\Delta + 1})(1 - p^{2N - 2M + \Delta})\right)
\]

\[
= \frac{1 + p + p^{2N} + p^{2N + 1} - 2p^{m + 1} - 2p^{2N - m}}{(1 + p)(1 - p^{N})^2} \Pr[|X'| \leq M]
\]

\[
\leq \left(1 - (1 - p^N)\right) p^M - \Delta
\]

\[
\leq \delta.
\]

Differential privacy then follows from Lemma 1. For \(0 \leq \alpha \leq M = \lfloor \alpha \rfloor\), we have

\[
\Pr[|X| > \alpha] = 1 - \Pr[|X'| \leq \alpha] \leq \frac{1}{2} \left(1 - \frac{\Pr[|X'| \leq M]}{\Pr[|X'| \leq M]}\right)
\]

\[
= \frac{2^{k+1}(1 - p^{M - k})(1 - p^{2N - M - k})}{(1 + p)(1 - p^{N})^2} \Pr[|X'| \leq M]
\]

\[
\leq \frac{1}{2} \left(1 - p^N\right) \Pr[|X'| = 0]
\]

\[
\leq \delta.
\]

**Remark.** We give an explicit way to choose parameters \(p, N, M\) satisfying the conditions (2) and (3) given \(\epsilon, \delta, \Delta\). We first show a more simple sufficient condition for (2) that

\[
\epsilon = \frac{2(1 - \zeta)}{\zeta^2} + 1
\]

and

\[
p = \exp\left(-\frac{\epsilon - \theta(K)}{\Delta}\right),
\]

where \(\zeta = \epsilon/2\Delta\) and \(\theta(K) = \ln(1 + K^{-1})\). Given \(\epsilon, \delta, \Delta\), choose \(K\) as any even number satisfying (4) and set \(p\) according to (5). Then, choose \(M\) as a sufficiently large number satisfying

\[
\frac{M - \Delta}{(1 + p)(1 - p^{K + 2M + 2\Delta})} \leq \delta
\]

and finally set \(N = K/2 + M + \Delta\).

To see (4) and (5) imply (2), let \(\eta = \exp(\zeta)\). Since \(\eta - 1 \geq \zeta\), we have that

\[
\eta^K = (1 + (\eta - 1))^K
\]

\[
\geq 1 + K\zeta + \frac{K(K - 1)}{2}\zeta^2
\]

\[
\geq K + 1.
\]

The last inequality follows from \(K \geq 2(1 - \zeta)/\zeta^2 + 1\). Since \(K \geq 2/\epsilon\), we have \(\epsilon - \theta(K) \geq \epsilon/2\) and

\[
\exp\left(K(\epsilon - \theta(K))\right) \geq \eta^K \geq K + 1
\]
and hence
\[
1 - \exp \left(\frac{K(\epsilon - \theta(K))}{\Delta}\right) \geq \frac{K}{K + 1}.
\]
Since \(\ln(K/(K + 1)) = -\theta(K)\), we then have that
\[
0 \leq \theta(K) + \ln \left(1 - \exp \left(-\frac{K(\epsilon - \theta(K))}{\Delta}\right)\right) = \epsilon + \Delta \ln p + \ln(1 - p^K).
\]
Hence, it holds that \(\epsilon \geq -\Delta \ln p - \ln(1 - p^K) + \ln(1 - p^{2N})\) and \(e^\epsilon \geq p^{-\Delta(1 - 1/p^{2N})}/(1 - p^{2N-\Delta})\).

To obtain a noise generator function for FDL\((p,N,M)\), we show the following lemmas. Lemma 4 readily follows from the definitions of FDL\((p,N)\) and FDL\((p,N,M)\). Lemma 5 is implicitly shown in [5].

**Lemma 4.** Let \(0 < p < 1\) and \(N, M \in \mathbb{N}\) be such that \(N > M\).

Let \(X_1, X_2\) be independent random variables following Geo\((p,N)\). Define \(X\) over \(\mathbb{Z}_q \cup \{\bot\}\) as
\[
X = \begin{cases} X_1 - X_2, & \text{if } |X_1 - X_2| \leq M, \\ \bot, & \text{otherwise}. \end{cases}
\]
Then, the distribution of \(X\) conditioned on \(X \neq \bot\) is FDL\((p,N,M)\). Furthermore, \(\Pr[X = \bot] \leq 2p^{M+1}/((1 + p)(1 - p^{N}))\).

**Lemma 5 ([5]).** Let \(N = 2^c\) for \(c \in \mathbb{N}, 0 < p < 1, \) and \(\beta_i = (1 + p^{2^{-c}})^{-1}\) for \(i \in [0..c]\). Define \(G_{p,N} : \{0,1\}^c \rightarrow \mathbb{Z}_q \times \mathbb{Z}_q\) as \(G_{p,N}(b_0,\ldots,b_{c-1}) = \sum_{i=0}^{c-1} b_i 2^i\). If \(X_i \sim \text{Ber}(\beta_i)\) for each \(i \in [0..c]\), then \(G_{p,N}(X_0,\ldots,X_{c-1})\) follows Geo\((p,N)\).

**Proof:** Define \(Z_{\geq c} = \{z \in \mathbb{Z} : z \geq c\}\) for \(c \in \mathbb{Z}\). Let \(\beta_i = (1 + p^{-2i})^{-1}\) also for \(i \in \mathbb{Z}_{\geq c}\). It is shown in [5] that \(G_{p,N}(b_0,\ldots,b_{c-1})\) has the geometric distribution Geo\((p)\), i.e., \(\Pr[G = g] = (1-p)p^g\) for \(g \geq 0\). Therefore, for every finite set \(I \subseteq \mathbb{Z}_{\geq 0}\),
\[
\Pr[X_i = 1 (\forall i \in I) \land X_i = 0 (\forall i \in (\mathbb{Z}_{\geq 0} \setminus I))] = (1-p)p^{g(I)},
\]
where we define \(g(I) = \sum_{i \in I} 2^i\). Let \(g \in [0..N]\) and \(I \subseteq [0..c]\) be such that \(g = g(I)\).
\[
\Pr\left[\sum_{i \in [0..c]} X_i 2^i = g\right] = \Pr\left[\begin{array}{c} X_i = 1 (\forall i \in I) \\ X_i = 0 (\forall i \in ([0..c] \setminus I)) \end{array}\right] = \sum_{J \subseteq \mathbb{Z}_{\geq c}} (1-p)p^{g(I,J)} = (1-p)p^g \sum_{J \subseteq \mathbb{Z}_{\geq c}} p^{g(J)}.
\]
Since \(\sum_{g \in [0..N]} \Pr\left[\sum_{i \in [0..c]} X_i 2^i = g\right] = 1\), we have
\[
\sum_{g \in [0..N]} p^{g(J)} = \left(\sum_{g \in [0..N]} (1-p)p^g\right)^{-1} = C.
\]

Now, let \(c,d \in \mathbb{N}, N = 2^c, M < N, m = 2cd,\) and \(0 < p < 1\). Define \(\beta_i = (1 + p^{2^{-c}})^{-1}\) for \(i \in [0..c]\). Define \(F_1 : \{0,1\}^m \rightarrow \mathbb{Z}_q \cup \{\bot\}\) as follows: For \(u = ((u_1,\ldots,u_d)_{i \in [0..c]},(v_1,\ldots,v_d)_{i \in [0..c]}) \in [0,1]^m\),
1) for \(i \in [0..c]\) and \(j \in [d]\), let \(a_i = B_{\beta_i,d}(u_1,\ldots,u_d)\) and \(b_i = B_{\beta_i,d}(v_1,\ldots,v_d)\);

![Fig. 3: The protocol \(\Pi_{FDL_1}\) for computing \(F_1\)](image)

2) let \(x_1 = G_{p,N}(a_0,\ldots,a_{c-1})\) and \(x_2 = G_{p,N}(b_0,\ldots,b_{c-1})\);
3) let
\[
F_1(u) = \begin{cases} x_1 - x_2, & \text{if } |x_1 - x_2| \leq M, \\ \bot, & \text{otherwise}. \end{cases}
\]
Since the statistical distance between \(a_i\) (or \(b_i\)) and \(\text{Ber}(\beta_i)\) is at most \(2^{-t}\), we have that \(SD(F_1(u),X) \leq c2^{-d+1}\) for \(u \sim \text{Uni}([0,1]^m)\) and \(X\) defined in Lemma 4.

In view of Lemmas 2, 3, and Proposition 3, we can estimate differential privacy and utility provided by \(F_1(u)\) for \(u \sim \text{Uni}([0,1]^m)\). We show an MPC protocol for \(F_1\) in Fig. 3. Combining it with Corollary 1, we have the following theorem. Note that since \(G_{p,N}\) is a linear function of inputs, it is straightforward to construct a non-interactive protocol Geo\(_{p,N}\) securely computing \(G_{p,N}\).

**Theorem 1.** Let \(g : D^n \rightarrow \mathbb{Z}_q\) be a function with sensitivity \(\Delta\) and range \(R\). For \(\epsilon > 0\) and \(\delta > 0\), let \(0 < p < 1\), \(N = 2^c\) for \(c \in \mathbb{N}\), and \(M \in \mathbb{N}\) be the ones satisfying the conditions (2), (3), and that \(q/2 > N + R\). Let \(\alpha, \beta \in \mathbb{R}\) be such that \(\alpha = (\Delta/\epsilon) \ln(2/\beta(1-p)(1-p^{2N}))\). For any \(t < n/2\) and any \(d \in \mathbb{N}\), there is a \((t,\epsilon,\delta')\)-differentially private protocol computing \(g\) with \((\alpha,\beta,\delta^0+\delta_1,\delta_2)\)-utility, where \(\delta_0 = c2^{-d+1}\), \(\delta_1 = 2p^{M+1}/((1 + p)(1 - p^{N})^2\), and \(\delta' = \delta + \delta_0(\epsilon^\gamma + 1)\).

### 4.2 The Second Protocol

Next, we construct a protocol for generating noise drawn from another finite-range variant of DL\((p)\). Let \(0 < p < 1\) and \(N \in \mathbb{N}\). Let FDL\(_2(p,N)\) denote the probability distribution over \(\{z \in \mathbb{Z} : -N \leq z \leq N\}\) defined as
\[
\Pr[X = x] = \begin{cases} p^{\lfloor x(1-p)/(1+p)\rfloor}, & \text{if } |x| < N, \\ p^{\lfloor n/(1+p)\rfloor}, & \text{if } |x| = N, \\ 0, & \text{otherwise}. \end{cases}
\]

**Proposition 3.** Let \(X \sim \text{FDL}_2(p,N)\). The functionality \(M(\cdot) = g(\cdot) + X\) is \((\epsilon,\delta)\)-differentially private for a function \(g : D^n \rightarrow \mathbb{Z}_q\) with sensitivity \(\Delta\) and range \(R\) if
\[
p = \exp \left(-\frac{\epsilon}{\Delta}\right) \cdot p^{\frac{N}{1+p} + \frac{\Delta}{1+p} \leq \delta, \text{ and } q/2 > N + R.\]

(6)
Observe that $\gamma$ is defined as $\frac{(\Delta_\epsilon) \ln (2/\beta)}{\epsilon}$. 

**Proof:** Differential privacy roughly follows by applying Lemma 1 to FDL$_2(p, N)$ although we have to slightly modify Lemma 1 so that $Pr[N - \Delta \leq X \leq N]$ (instead of $Pr[N - \Delta \leq X < N]$) is upper bounded by $\delta$. For $0 \leq \alpha < N$ and $k = \lfloor \alpha \rfloor$, we have

$$Pr[|X| \leq \alpha] = \sum_{i=-k}^{k} \frac{1-p^i}{1+p} = 1 - \frac{2p^{k+1}}{1+p} > 1 - \frac{2p^\alpha}{1+p}.$$ 

Therefore, $M$ satisfies $(\alpha, \beta)$-utility if $\beta = 2p^\alpha/(1 + p)$, i.e.,

$$\alpha = \frac{\Delta \ln 2}{\beta (1 + p)} \leq \frac{\Delta \ln 2}{\beta^2}.$$

We show that sampling a value from FDL$_2(p, N)$ can be reduced to sampling sufficiently many bits from Ber($\alpha$).

**Proposition 4.** Assume that $q/2 > N$. Let $B_0$ be a random variable with Ber((1 - $p$)/$1 + p$) and $B_1, \ldots, B_N$ be independent random variables with Ber($1 - p$). Define a random variable $Y$ on Z as

$$Y = \begin{cases} \min \{ i \in \{0..N \} : B_i = 1 \}, & \text{if } B_i = 1 \text{ for some } i, \\ N, & \text{otherwise.} \end{cases}$$

Then, the distribution of $X = \sigma Y$ for $\sigma \sim \text{Unif}\{-1, +1\}$ is FDL$_2(p, N)$.

**Proof:** Let $p_0 = (1 - \varepsilon)/(1 + \varepsilon)$ and $p_1 = 1 - p$. Observe that $Pr[\sigma Y = 0] = p_0$, $Pr[\sigma Y = k] = (1/2)(1 - p_0)(1 - p_1)^{|k| - 1}p_1$ if $0 < |k| < N$, and $Pr[\sigma Y = k] = (1/2)(1 - p_0)(1 - p_1)^N$ if $|k| = N$. We then obtain a noise generator function for FDL$_2(p, N)$. Let $N, d \in \mathbb{N}$, $m = Nd + 1$, and $0 < p < 1$. Define $\gamma_i$ for $i \in \{0..N\}$ as $\gamma_0 = (1 - p)/(1 + p)$ and $\gamma_1 = 1 - p$ for $i \neq 0$. Define $F_2 : \{0, 1\}^m \rightarrow \mathbb{Z}_q$ as follows:

1. let $b_i = B_{\gamma_i}(b_{i,1}, \ldots, b_{i,d})$ for $i \in \{0..N\}$ and $\sigma = 1 - 2b_i$
2. let $y = \min \{ i \in \{0..N\} : b_i = 1 \}$ if $b_i = 1$ for some $i$ and otherwise $y = N$
3. let $F_2(b) = \sigma y$

Since the statistical distance between $b_i$ and Ber($\gamma_i$) is at most $2^{-d}$, we have that $SD(F_2(b), \text{FDL}_2(p, N)) \leq N2^{-d}$ for $\sigma \sim \text{Unif}\{0, 1\}^m$). We can also directly obtain an MPC protocol $\Pi_{\text{FDD}_2}$ for $F_2$. (Fig. 4). Combining it with Corollary 1, we have the following theorem.

**Theorem 2.** Let $g : D^a \rightarrow \mathbb{Z}_q$ be a function with sensitivity $\Delta$ and range $R$. For $\epsilon > 0$ and $\delta > 0$, let $0 < p < 1$ and $N \in \mathbb{N}$ be the ones satisfying the condition (6). Let $\alpha, \beta \in \mathbb{R}$ be such that $\alpha = (\Delta_\epsilon) \ln (2/\beta)$. For any $t < n/2$ and any $d \in \mathbb{N}$, there is a $((t, \epsilon, \delta'), \beta$)-differentially private protocol computing $g$ with $(\alpha, \beta + \delta_0)$-utility, where $\delta_0 = N2^{-d}$ and $\delta' = \delta + \delta_0(\epsilon^2 + 1)$.

---

**Input.** $(\lfloor b_{i,1} \rfloor, \ldots, \lfloor b_{i,d} \rfloor)_{i \in \{0..N\}}$, $[b]$, where $b_{ij}, b \in \{0, 1\}$.

**Output.** $[F_2(b_{i,1}, \ldots, b_{i,d})_{i \in \{0..N\}}, y]$. 

**Protocol.**

1. $[b_i] = \text{Ber}_\gamma, d((b_{ij})_{j \in \{d\}})$ for $i \in \{0..N\}$.
2. $(\lfloor c_j \rfloor)_{j \in \{0..N\}} = \text{Pre}_\gamma((b_{ij})_{j \in \{0..N\}})$.
3. $\lfloor y \rfloor = N - \sum_{i \in \{0..N\}} [c_i]$.
4. $[\sigma] = 1 - 2[\bar{b}]$.
5. Output $[\sigma y] = \text{MULT}([y], [\sigma])$.

---

**Fig. 4:** The protocol $\Pi_{\text{FDD}_2}$ for computing $F_2$

**Fig. 5:** The protocol $\Pi_{\text{Bin}}$ for computing $h$

---

5 A NOVEL NOISE GENERATION PROTOCOL FOR THE BINOMIAL DISTRIBUTION

We provide a protocol which allows parties to non-interactively obtain a share of noise drawn from $\text{NBin}(\ell, M)$ by using predistributed keys for a pseudorandom function.

We define some notations. Let $\Lambda \in \mathbb{N}$ be a security parameter. Let $U$ be the set of all possible tuples of keys for pseudorandom secret sharing. Formally, recall that we have defined $A = \{A \subseteq \{n\} : |A| = n - t\}$ and $A_i = \{A : i \in A\}$ for $i \in \{n\}$. We define $S = \{0, 1\}^{\ell(\lambda - 1)}$ and $U = \{((r_A, i)_{A \in A_i})_{i \in \{n\}} : r_A, i \in A \cap A_i\}$.

We restrict ourselves to the family $\{\psi_r : \{0, 1\}^{s(\lambda)} = \{0, 1\}^{s(\lambda)}\}_{r \in \{0, 1\}^s}$ with key length of $\lambda$ and simply write $s = s(\lambda)$ and $\ell = \ell(\lambda)$. Let $\ell_r(a)$ be the number of 1’s in $\psi_r(a) \in \{0, 1\}^s$ for $r \in \{0, 1\}^s$ and $a \in \{0, 1\}^s$.

Assume that $q > \ell[A]$. Since $s \in U$ can be written as $s = ((r_A, A)_{A \in A_i})_{i \in \{n\}}$ for some $r_A \in \{0, 1\}^\lambda$, it is possible to define a function $h : U \rightarrow \mathbb{Z}_q$ (depending on a public input $a \in \{0, 1\}^s$) as $h(s) = \sum_{A \in A_i} r_A(a)$. Then, the pseudorandomness of $\psi_r$ implies that $(1/M)(h(s) - \ell[A]/2)$ works as a noise generator function for $\text{NBin}(\ell[A], M)$ in the computational setting. The protocol $\Pi_{\text{Bin}}$ described in Fig. 5 non-interactively (and hence t-securely) computes $h(s)$ from $s \in U$. Note that generating a uniformly random element in $U$ is equivalent to generating random shares of the replicated secret sharing scheme, which in turn can be securely done by a known protocol $\Sigma_{\text{Ber}}$ [36].

The following theorem (Theorem 3) mostly follows from the above observation and Proposition 1. One exception is that parties run $\Pi_{Mg}$ rather than $\Pi_p$, where $Mg$ is defined as $\langle Mg(x) \rangle = M \cdot g(x)$, and then compute $(1/M)(y - \ell[A]/2)$ from the recovered secret $y$ as plaintexts. We do that procedure to avoid an arithmetic over operation over $\mathbb{R}$ in secret-shared form. For completeness, we formally describe the
protocol II in Fig. 6. Its utility is discussed after the proof of Theorem 3.

Theorem 3. Let $g : D^n \rightarrow \mathbb{Z}_q$ be a function with sensitivity $\Delta$ and range $R$. Let $\lambda$ be a security parameter and assume a pseudorandom function $\{\psi_r : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda\}_{r \in \{0,1\}^\lambda}$. For $\epsilon \in (0, \log \lambda)$ and $\delta > 0$, assume that there exists $M \in \mathbb{N}$ such that $\epsilon \geq \epsilon(\delta, \ell, M, \Delta)$ and $q > MR + \ell |A|$. For any $\ell < n/2$, the protocol II described in Fig. 6 is a $(t; \ell, \delta)$-computationally differentially private protocol for computing $g$. 

Proof: Fix a set $T \subseteq [n]$ of $t$ corrupted parties and let $J = [n] \setminus T \in A$. The output of II has the form of $M_{g,J,A,M}(x)$. However, the adversary knows all but one keys $r_A, A \neq J$ and the only noise unknown to him is $\ell_J(a)$. Therefore, an achievable level of differential privacy against the adversary’s view deteriorates to that of $M_{g,M}(x)$.

More formally, let $f(x; s) = g(x) + (1/M)(h(s) - \ell |A|/2)$ for $x \in D^n$ and $s \in U$. In view of Proposition 1, it is sufficient to show that for all $T$-neighboring vectors $x, y$, two distributions $(x_T, s_T, f(x; s_T)), (y_T, s_T, f(y; s_T))$ induced by $s \sim Un(U)$ are computationally $(\epsilon, \delta)$-DP close even when the randomness of $s_T$ is fixed. Recall that we have defined $U_T = \{u \in S^{nT} : u = s_T \text{ for some } s \in U\}$.

First, define a randomized function $f_J$ as $f_J(x; u) = f_J(x; u) + (1/M)(\ell_J - \ell/2)$ for $x \in D^n$ and $u = ((r_A)_{A \in A, i \in T}) \in U_T$, where $f_J(x; u) = g(x) + (1/M)\sum_{A \in A, i \in T} (\ell_A(a) - \ell/2)$ and $\ell_J \sim Bin(\ell, 1/2)$. In other words, if $u = s_T$, then $f_J(x; u)$ is defined by replacing $\ell_J(a)$ in $f(x; s)$ with $\ell_J$ properly sampled from $Bin(\ell, 1/2)$. For fixed $u \in U_T$, $f\ell(x; u)$ is equivalent to the $(\epsilon, \delta)$-differentially private mechanism $M_{g,M}(x)$ and hence for all $T$-neighboring vectors $x, y$ and for any $u \in U_T$, two distributions $(x_T, u, f_J(x; u)), (y_T, u, f_J(y; u))$ are $(\epsilon, \delta)$-DP close, where the randomness is $\ell_J \sim Bin(\ell, 1/2)$.

Next, we show that for any but fixed $x \in D^n$ and $u = ((r_A)_{A \in A, i \in T}) \in U_T$, the distribution of $f(x; u)$ induced by $\ell_J \sim Bin(\ell, 1/2)$ and that of $f(x; ((r_A)_{A \in A, i \in [n]})) = f_J(x; u) + (1/M)(\ell_A(a) - \ell/2)$ induced by $r_J \sim Un(U)$ are computationally indistinguishable. Assume otherwise that there are a PPT distinguisher $D$ which can distinguish $(x_T, u, f(x; ((r_A)_{A \in A, i \in [n]})))$ from $(x_T, u, f_J(x; u))$ with non-negligible advantage for some $x \in D^n$ and $u = ((r_A)_{A \in A, i \in T}) \in U_T$. We construct a PPT oracle machine $Alg$ for $\{\psi_r : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda\}_{r \in \{0,1\}^\lambda}$ as follows. First, $Alg$ invokes the oracle to receive $\xi^b \in \{0,1\}^\ell$, and sets $\ell^b$ as the number of $1$’s in $\xi^b$. Here, the oracle flips a bit $b \sim Un(\{0,1\})$ and sets $\xi^b = \psi_{r_J}(a)$ for $r_J \sim Un(\{0,1\}^\lambda)$ if $b = 1$ or else $\xi^b = F(a)$ for a uniformly selected map $F : \{0,1\}^\lambda \rightarrow \{0,1\}^\ell$. Next, $Alg$ computes $z^b = f_J(x; u) + (1/M)(\ell^b - \ell/2)$. Note that $z^b$ gives $(x_T, u, z^b)$ to $D$ and receives a guess $b' \in \{0,1\}$ from $D$. Finally, it outputs $b'$. We would have that $Pr[b' = b] = 1/2$ is non-negligible, which contradicts the indistinguishability of $z$.

For any PPT distinguisher $D$, for all $T$-neighboring vectors $x, y$, and for any $u \in U_T$, we have that

$$Pr_{s \sim Un(U)}[D(x_T, s_T, f(x; s)) = 1 | s_T = u] = Pr_{r_J \sim Bin(\ell, 1/2)}[D(y_T, s_T, f(y; s)) = 1 | s_T = u] + \text{negl}(\lambda)$$

$$\leq c\epsilon \Pr_{r_J \sim Bin(\ell, 1/2)}[D(y_T, s_T, f_J(y; u)) = 1] + \delta + \text{negl}(\lambda) = c\epsilon \Pr[D(y_T, s_T, f(y; s)) = 1 | s_T = u] + \delta + \text{negl}(\lambda).$$

We use the assumption that $\epsilon \in O(\log \lambda)$.

As for the utility, we assume that the statistical distance between the uniform distribution over $\{0,1\}^\ell |A|$ and $(r_A)_{A \in A, i \in [n]}$ for $r_A \sim Un(\{0,1\}^\lambda)$ is at most $\delta$. Then, the statistical distance between $Bin(\ell |A|, 1/2)$ and $\sum_{A \in A} e_{r_A}(a)$ for $r_A \sim Un(\{0,1\}^\lambda)$ is also upper bounded by $\delta$. Based on the utility of $M_{g,M}(x)$, we can estimate that the functionality $M(x) = g(x) + (1/M)\sum_{A \in A} e_{r_A}(a) - \ell |A|/2$ and hence the protocol II satisfies $(\alpha, \beta, \delta')$-utility for $g$ if $\alpha = \sqrt{(\ell |A|/2M^2)\ln(2/\beta)}$.

6 COMPARISON

6.1 The Discrete Laplace Distribution

We compare our protocols in Section 4 with the one in [5]. The following comparison is based on the MPC primitives in Section 2.2.3. To describe their protocol [5], let $N$ be a power of 2 and $L \sim TDL(p, N)$ be the truncated discrete Laplace distribution, i.e., $Pr[L = k] = Cp_{N}(1/p)/(1+p)$, $-N < k < N$, where $C = (1 + p)/(1 + p - 2p^N)$ is a normalizing constant. Technically, their protocol generates shares of $g$ drawn from Geo(p, N) using log $N$ independent biased bits. Then, according to [29], $\sigma g$ conditioned on $(g, s) \neq (0, -1)$ follows $TDL(p, N)$ if $\sigma \sim Un\{-1, +1\}$. Since the statistical distance between the output distribution and $TDL(p, N)$ is $2^{-d} \log N$ for a parameter $d$, it follows from Lemma 1 that to achieve $(\epsilon, \delta)$-differential privacy for a function with sensitivity $\Delta$, it is necessary to choose $p, N, d$ such that $p = \exp(-\epsilon/\Delta)$ and

$$e^c + 1 \leq \frac{\log N}{1 + p - 2p^N} \leq \delta.$$
TABLE 1: Comparison of protocols sampling noise from finite-range discrete Laplace distributions

<table>
<thead>
<tr>
<th>Reference</th>
<th>Distribution</th>
<th>Round</th>
<th>Communication</th>
<th>Probability of failure</th>
<th>Statistical distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>[5]</td>
<td>TDL(p, N)</td>
<td>2⌊log d⌋ + 11</td>
<td>O(d log N + log q)</td>
<td>(1 − p)/2</td>
<td>2−d log N</td>
</tr>
<tr>
<td>Theorem 1</td>
<td>FDL(p, N, M)</td>
<td>19</td>
<td>O(d log N + log q)</td>
<td>2p^{3d+1}/((1 + p)(1 − p^N)^2)</td>
<td>2−d log N</td>
</tr>
<tr>
<td>Theorem 2</td>
<td>FDL_2(p, N)</td>
<td>14</td>
<td>O(dN)</td>
<td>0</td>
<td>2−dN</td>
</tr>
</tbody>
</table>

Fig. 7: Our protocols for the discrete Laplace distributions performing the sub-protocols in parallel as much as possible.

Boolean circuit for generating biased bits. The authors of [42] propose a circuit of size 7d − 3 and depth 2⌊log d⌋ + 2 to sample biased bits with statistical difference at most 2−d. The total round complexity of [5] is therefore given as 2⌊log d⌋ + 11, which depends on δ due to the condition (7).

We note that it is possible to further reduce the above round complexity by using pseudorandom secret sharing, which comes at the cost of satisfying only computational differential privacy. Specifically, our protocols can be done in 8 rounds for FDL_1(p, N, M) and 7 rounds for FDL_2(p, N) while [5] needs 2⌊log d⌋ + 7 rounds by replacing the primitives in Section 2.2.3 with the ones based on pseudorandom secret sharing [39].

6.1.4 Concrete Comparison

We estimate the running time required by the protocol of [5] and ours (Theorems 1 and 2). Consider the client-server model and set n = 3 and t = 1. Let g be any function with sensitivity Δ = 1 and q = 2^61 − 1. We assume the WAN setting, and set the network speed to B = 100 Mbits/sec and the latency to 0.1 sec [43]. We calculate the running time as (# comm. bits)/B + (# rounds) × (latency), assuming that the local computation time is negligible compared with the communication time. Times to generate one Laplace noise are shown in Fig. 8, where the privacy budget ε ranges from 0.1 to 1, for

- δ = 2−100, 2−80 and 2−60, with β = 0.2 and α = 50;
- α = 10, 20 and 30, with β = 0.2 and δ = 2−60.

We do not plot the cases where protocols cannot achieve that privacy budget ε under the constraint of the (α, β)-utility.

We figure out that our protocols are up to around 1.3 times faster than the protocol [5]; e.g., our first protocol is at least 1.27 times faster in Fig. 8 (a). This is because the dominant cost in the running time comes from the round complexity in the WAN setting. Since the protocol [5] has non-negligible probability of failure (1 − p)/2 = (1 − exp(−ε/δ))/2, it cannot be applied to some ranges of ε under the constraint of utility. On the other hand, our protocols are applicable to wider ranges of ε; see Figs. 8 (d–f).

As a comparison between our protocols, we note that the first protocol (Theorem 1) is faster than the second one (Theorem 2) if ε gets close to 0 in Figs. 8 (a–c). This is because the communication complexity increases as ε → 0 and then it becomes a dominant factor of the running time. An advantage of the second protocol is that under the strong constraint of utility, it is even applicable to ε such that the first one is not; see Figs. 8 (d–f). It comes from the fact that it has the zero probability of failure and as a result, satisfies higher utility with the same communication cost.

6.1.2 Utility

In view of Lemmas 2 and 3, a protocol achieves (α, β + δ_0 + δ_1)-utility if its target noise distribution provides (α, β)-utility, the statistical distance from its output distribution is δ_0, and the probability of failure is δ_1. Since the probability of failure of [5] is δ_1 = (1 − p)/2 depending only on ε and Δ, it is impossible to make it negligible no matter how we choose other parameters d, N, and q. On the other hand, our protocols always generate an appropriate value or fail only with negligible probability. Hence, it is possible to make the utility of our protocols arbitrarily close to that of the target noise distribution.

6.1.3 Communication Complexity

It follows from the size of the Boolean circuit [42] that the protocol [5] requires 9d log N − 3 log N + 162 log q + 4 = O(d log N + log q) invocations. Our protocol for FDL_1(p, N, M) needs 38d log N + 2log N + 110 log q + 1 = O(d log N + log q) invocations and the one for FDL_2(p, N) needs 19dN + 18N + 3 = O(dN) invocations. Although our protocols require asymptotically higher communication complexity than [5], the difference is not significant in practical parameter settings as described below.
exponentially in the length of the fractional part of the data

g(∆ = max_{x \neq y} |g(x) - g(y)|)^{2/k} \quad x, y \text{ are neighboring},

which means that \( N \) grows exponentially in the length of the fractional part of the data type.

6.2 The Binomial Distribution

We compare our protocol in Section 5 with the ones in [5]. The first protocol of [5] generates shares of \( N \) random bits using \( \Sigma_{Bin} \) and then locally computes a share of \( Z \sim NBin(N, M) \). The second protocol of [5] lets each party \( i \) share \( N/n \) (local) random bits among the other parties and then locally compute a share of \( Z \). Since \( tN/n \) out of the \( N \) bits are revealed to the adversary, the latter is \((\epsilon, \delta)\)-differentially private only for \( \epsilon \geq \epsilon_\Delta \). Our protocol non-interactively communicates \( O(N) \) bits independent of \( N \).

6.2.1 Communication Complexity

At the cost of predistributing keys for a pseudorandom function with output length \( \ell \), our protocol non-interactively generates shares of \( Z \sim NBin(N, M) \) by using the keys \( N/\ell \times t \) times (if overflow does not occur). The communication cost to distribute the keys is \( \lambda(n - t)\binom{n}{t} = O(\lambda n^{t+1}) \), independent of \( N \) or \( M \). Furthermore, the communication cost in the setup can be amortized by reusing the keys. However, both of the protocols [5] require communication complexity proportional to \( N \) when generating \( Z \sim NBin(N, M) \). Approximately, \( N \) grows proportionally to \((\Delta/\epsilon)^2\) in view of the condition (1). In addition, when a fixed-point data type with resolution \( 2^{-k} \) is dealt with, \( N \) should satisfy the condition (1) for \( \Delta = \max_{x \neq y} |g(x) - g(y)| \). This means that \( N \) grows exponentially in the length of the fractional part of the data type.

6.2.2 Utility

To achieve the same level of differential privacy \( \epsilon \) and \( \delta \), the error bound \( \alpha \) of our protocol is \((n/\ell)^{1/2} = O(n^{1/2})\) times larger than [5]. Nevertheless, we should also consider the client-server model for practical applications. Our protocol is available and even suitable for this model since \( n \) corresponds to the number of servers and is typically small, e.g., \( n = 3 \).

6.2.3 Concrete Comparison

We estimate the running time required by the protocols of [5] and ours (Theorem 3). Consider the same setting as Section 6.1.4. That is, we set \( n = 3 \), \( t = 1 \), \( \Delta = 1 \) and \( q = 2^{51} - 1 \). We again assume the WAN setting, and set the network speed to \( B = 100 \text{ Mbits/sec} \) and the latency to \( 0.1 \text{ sec} \). We use the pseudorandom function based on AES with \( \lambda = 128 \) and \( s = \ell = 128 \), assuming \( \delta_\psi = 0 \). The local computation is considered to be dominated by the execution of AES. We thus calculate the running time as \((\# \text{ comm. bits})/B + (\# \text{ rounds}) \times (\text{latency})\) for the protocols [5] and as \((\# \text{ comm. bits})/B + (\# \text{ rounds}) \times (\text{latency}) + 100t_{AES}N/\ell\) for our protocol, where \( t_{AES} \) is the running time of one AES execution. We use the value \( t_{AES} = 3.5 \text{ nsec} \). Amortized times to generate 100 binomial noises are shown in Fig. 9, where the privacy budget values \( \beta \) range from 2 to 4, for \( \alpha = 100, 150 \) and \( 200 \), with \( \beta = 0.2 \) and \( \delta = 2^{-20} \). Our protocol only communicates \( \lambda(n - t)\binom{n}{t} = 768 \text{ bits independent of } \epsilon \) and \( \delta \). We figure...
out that the running times of ours lie between 0.1 sec and 0.22 sec for all privacy budgets. On the other hand, those of [5] become higher if ε gets close to 0, i.e., higher differential privacy is required. Our protocol is up to around $2^{10}$ times for $\alpha = 100$, 23 times for $\alpha = 150$ and 14 times for $\alpha = 200$.

7 Extension to Active Security

We demonstrate that MPC protocols in our framework in Section 3 can achieve active security if actively secure protocols for evaluating deterministic functions and for jointly generating uniformly random elements are given. In our protocols in Sections 4 and 5, the task of generating uniformly random elements is equivalent to giving parties random shares of Shamir and replicated secret sharing schemes, respectively. Therefore, our protocols can provide differential privacy even in the presence of active adversaries if the corruption threshold $t$ is less than $n/3$.

To be precise, we first extend Definition 1 to the active setting. We define the ideal process and the real process. Consider an MPC protocol $\Pi$ associated with an $n$-party randomized functionality $F$.

**Ideal Process:** This process is defined with respect to a trusted party. A subset of parties $T$ can be corrupted by a PPT ideal process adversary $B$. The process proceeds in the following steps:

1. **Inputs:** The $i$-th party obtains an input $x_i$.

2. **Sending inputs to the trusted party:** An honest party $i \notin T$ always sends $x_i$ to the trusted party. A malicious party $i \in T$ may, depending on $x_i$, either abort or send some $x'_i$ to the trusted party.

3. **Trusted party answers i-th party:** Suppose that the trusted party receives inputs $x'_1$ from the $i$-th party. It sends the $i$-th output $y_i$ to the $i$-th party, where $F(x'_1, \ldots, x'_n) = (y_1, \ldots, y_n)$.

4. **Outputs:** If the $i$-th party is honest, it outputs $y_i$. The adversary $B$ outputs an arbitrary (polynomial-time computable) function of $x'_T$ and the message it has obtained from the trusted party.

We define $\text{Ideal}_B^\Pi(x)$ as the distribution defined over the view of $B$.

**Real Process:** The $i$-th party receives the input $x_i$. All the parties then execute the protocol $\Pi$. A subset of parties $T$ is controlled by an adversary $A$, who can deviate arbitrarily from the rules of the protocol. We define $\text{View}_A^\Pi(x)$ as the distribution over the view of $A$.

We denote by $F_g$ a functionality of computing a deterministic function $g$, that is, $F_g$ outputs $g(x)$ if it receives $x$ from the parties.

**Definition 2.** We call $\Pi$ a $(t; \epsilon, \delta)$-differentially private protocol for computing $g$ with $(\alpha, \beta)$-utility in the presence of active adversaries if for any adversary $A$ in the real process corrupting a subset of $t$ parties $T$, there exists a PPT adversary $B$ in the ideal process for $F_g$ such that:

- **$t$-Privacy:** for any input $x$, the distributions of $\text{View}_A^\Pi(x)$ and $\text{Ideal}_B^\Pi(x)$ are identical.

- **$(\epsilon, \delta)$-Differential Privacy:** for any pair of $T$-neighboring inputs $x$ and $y$, the distributions of $\text{View}_A^\Pi(x)$ and $\text{View}_A^\Pi(y)$ are $(\epsilon, \delta)$-DP close.

- **$(\alpha, \beta)$-Utility:** for any input $x_i$ and $i \notin T$, it holds that

$$\Pr \left[ \left| \text{Output}_A^\Pi(x_i) - g(x_i^B) \right| \leq \alpha \right] \geq 1 - \beta,$$

where $\text{Output}_A^\Pi(x_i)$ is a part of the output the $i$-th party receives at the end of the real process and $x_i^B$ is a tuple of inputs $(x_{[n] \setminus T}, x'_T)$ submitted to the trusted party in the ideal process.

As in Proposition 1, let $g : D^n \rightarrow \mathbb{Z}_q$ be a deterministic function to compute and $h : U \rightarrow \mathbb{Z}_q$ be a noise generator function. Let $t$ be a corruption threshold with $t < n/3$. Then, there is a protocol $\Pi_g$ (resp. $\Pi_h$) which takes $x \in D^n$ (resp. $s \in U$) as input and $t$-securely computes $(\|g(x)\|, i \in [n])$ (resp. $(\|h(s)\|, i \in [n])$) in the presence of active adversaries. Suppose that we are also given an actively secure protocol $\Sigma_{\text{ran}}$ realizing a functionality $F_{\text{ran}}$, which takes no input, samples $s = (s_i)_{i \in [n]} \sim \text{Uni}(U)$, and gives $s_i$ to the $i$-th party. Also, suppose that, for any subset $T$ of size $t$, any pair of $T$-neighboring vectors $x, y \in D^n$, and any $a \in U_T$, two distributions $(x_T, s_T, g(x) + h(s))$ and $(y_T, s_T, g(y) + h(s))$ conditioned on $s_T = a$ are $(\epsilon, \delta)$-DP close, where the randomness is $s \sim \text{Uni}(U)$. Finally, assume that $\Pr_{s \sim \text{Uni}(U)}[\|h(s)\| \leq \alpha] \geq 1 - \beta$ for $\alpha$ and $\beta$. Now, we construct a protocol $\Pi$ based on $\Pi_g$, $\Pi_h$, and $\Sigma_{\text{ran}}$ (Fig. 10).

**Proposition 5.** Using the above notations, the protocol $\Pi$ described in Fig. 10 is a $(t; \epsilon, \delta)$-differentially private protocol for computing $g$ with $(\alpha, \beta)$-utility in the presence of active adversaries.
**Input.** \( x \in D^n \).

**Output.** The \( i \)-th party receives \( (s_i, z) \), where \( s = \sum_{i \in [n]} s_i \sim \text{Uni}(U) \) and \( z = g(x) + h(s) \).

**Protocol.**
1. \( \Pi_{g}(x) = [g(x)] \)
2. \( s = \sum_{i \in [n]} s_i \)
3. \( h(s) = \Pi_h(s) \)
4. \( z = [g(x)] + [h(s)] \)
5. The \( i \)-th party receives the shares \( [z] \) and outputs \( (s_i, z) \).

Fig. 10: A \((t, \epsilon, \delta)\)-differentially private protocol for computing \( g \) in the presence of active adversaries.

**Proof:** Define \( \mathcal{F} \) as a functionality which takes \( x \in D^n \) as input, samples \( s \sim \text{Uni}(U) \), and gives \( (s, f(x; s)) \) to the \( i \)-th party, where \( f(x; s) : = g(x) + h(s) \). Then, the protocol \( \Pi \) \( t \)-securely realizes \( \mathcal{F} \). Indeed, \( \Pi \) simply invokes actively secure protocols \( \Pi_g, \Pi_h, \) and \( \sum_{\text{ran}} \) sequentially and opens \( z \) to every party. We apply the composition theorem [31] and use the robustness of Shamir secret sharing scheme since \( t < n/3 \).

Then, \( \Pi \) satisfies \((t, \epsilon, \delta)\)-differential privacy for computing \( g \) with \((\alpha, \beta)\)-utility. To see this, let \( A \) be an adversary in the real process. From the security of \( \Pi \) for \( \mathcal{F} \), we have an adversary \( A' \) in the ideal process for \( \mathcal{F} \) corresponding to \( A \). In particular, there exists a simulator \( \text{Sim} \) such that \( \text{View}^A_{\Pi}(x) \) is identical to \( \text{Sim}(\mathcal{F}(x; f(x; s))_{i \in T}) \) for any \( x \), where \( x_{i'\in T} \) is a tuple of inputs \( (x_{[n] \setminus T}; x_{i'\in T}) \) submitted by \( B \) and \( \mathcal{F}(x_{i'\in T})_{i \in T} = (s, f(x_{i'\in T}; s))_{i \in T} \) for \( s \sim \text{Uni}(U) \). The \( t \)-privacy of \( \Pi \) follows since we can obtain an adversary \( B \) in the ideal process for \( \mathcal{F}_g \) from \( B' \) and from a simulator \( \text{Sim} \) which on input \( x_{i'\in T} \) and \( g(x_{i'\in T}) \), locally samples \( s \sim \text{Uni}(U) \) and runs \( \text{Sim}' \) on \( (x_{i'\in T}, (s, f(x_{i'\in T}; s) + h(s))_{i \in T}) \). For differential privacy, observe that if \( x \) and \( y \) are \( T \)-neighboring, then so are \( x_{i'\in T} \) and \( y_{i'\in T} \) since \( B' \) substitutes corrupted parties’ inputs depending only on \( x_{i \in T} = y_{i \in T} \). Thus, the distributions \( (x_{i'\in T}, s_{i \in T}, f(x_{i'\in T}; s)) \) and \( (y_{i'\in T}, s_{i \in T}, f(y_{i'\in T}; s)) \) are \((\epsilon, \delta)\)-DP close if \( s \sim \text{Uni}(U) \). The existence of \( \text{Sim}' \) and the post-processing property of differential privacy imply that \( \text{View}^A_{\Pi}(x) \) and \( \text{View}^A_{\Pi}(y) \) are \((\epsilon, \delta)\)-DP close. The \((\alpha, \beta)\)-utility of \( \Pi \) can be derived from that of \( g(s) + h(s) \), \( s \sim \text{Uni}(U) \) and from the fact that Output\( A, i(x) \) is guaranteed to contain \( f(x_{i'\in T}; s) \) for \( s \sim \text{Uni}(U) \) by the security of \( \Pi \).

8 Conclusion

In this paper, we propose three efficient MPC protocols for generating shares of noise drawn from certain distributions capable of providing differential privacy. Our protocols for the discrete Laplace distribution improves the round complexity of the previous one [5]. Our third protocol enables parties to non-interactively compute shares of noise drawn from the binomial distribution by predistributing keys for a pseudorandom function. It reduces the communication complexity of [5] while it loses utility to some extent. Our protocols can be extended so that they provide differential privacy even in the presence of active adversaries.

**Acknowledgments**

This research was partially supported by JSPS KAKENHI Grant Numbers JP20J20797 and JP19K22838, JST CREST Grant Numbers JPMJCR14D6 and JPMJCR19F6, Japan, and the Ministry of Internal Affairs and Communications SCOPE Grant Number 182103105.

**References**


---

**Fig. 11:** Ratio between running times of our active and passive protocols.