Almost Instance-optimal Clipping for Summation Problems in the Shuffle Model of Differential Privacy

Wei Dong
Nanyang Technological University
Singapore, Singapore
wei_dong@ntu.edu.sg

Qiyao Luo
OceanBase, Ant Group
Shanghai, China
luoqiyao.lqy@antgroup.com

Giulia Fanti
Carnegie Mellon University
Pittsburgh, United States
gfanti@andrew.cmu.edu

Elaine Shi
Carnegie Mellon University
Pittsburgh, United States
runting@gmail.com

Ke Yi
Hong Kong University of Science and Technology
Hong Kong, Hong Kong
yike@cse.ust.hk

Abstract

Differentially private mechanisms achieving worst-case optimal error bounds (e.g., the classical Laplace mechanism) are well-studied in the literature. However, when typical data are far from the worst case, *instance-specific* error bounds—which depend on the largest value in the dataset—are more meaningful. For example, consider the sum estimation problem, where each user has an integer x_i from the domain $\{0, 1, \ldots, U\}$ and we wish to estimate $\sum_i x_i$. This has a worst-case optimal error of $O(U/\varepsilon)$, while recent work has shown that the clipping mechanism can achieve an instance-optimal error of $O(\max_i x_i \cdot \log\log U/\varepsilon)$. Under the shuffle model, known instance-optimal protocols are less communication-efficient. The clipping mechanism also works in the shuffle model, but requires two rounds: Round one finds the clipping threshold, and round two does the clipping and computes the noisy sum of the clipped data. In this paper, we show how these two seemingly sequential steps can be done simultaneously in one round using just $1 + o(1)$ messages per user, while maintaining the instance-optimal error bound. We also extend our technique to the high-dimensional sum estimation problem and sparse vector aggregation (a.k.a. frequency estimation under user-level differential privacy).

CCS Concepts

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

Keywords

Differential privacy; sum estimation

ACM Reference Format:
Wei Dong, Qiyao Luo, Giulia Fanti, Elaine Shi, and Ke Yi. 2024. Almost Instance-optimal Clipping for Summation Problems in the Shuffle Model of Differential Privacy. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3658644.3690225

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

ACM ISBN 979-8-4007-0636-3/24/10
https://doi.org/10.1145/3658644.3690225

1 Introduction

The *shuffle model* [10, 12, 20] of *differential privacy* (DP) is widely-studied in the context of DP computation over distributed data. The model has 3 steps: (1) Each client uses a randomizer $\mathcal{R}(\cdot)$ to privatize their data. (2) A trusted shuffler randomly permutes the inputs from each client and passes them to an untrusted analyzer, which (3) conducts further analysis. Unlike the *central model* of DP, where a trusted curator has access to all the data, the shuffle model provides stronger privacy protection by removing the dependency on a trusted curator. Unlike the *local model*, where clients send noisy results to the analyzer directly, the addition of the trusted shuffler allows for a significantly improved privacy-accuracy trade-off. For problems like bit counting, shuffle-DP achieves an error of $O(1/\varepsilon)$ with constant probability[1] [23], matching the best error of central-DP, while the smallest error achievable under local-DP is $O(\sqrt{n}/\varepsilon)$ [11, 13].

The *summation* problem, a fundamental problem with applications in statistics [9, 27, 33], data analytics [15, 44], and machine learning such as the training of deep learning models [1, 2, 8, 41] and clustering algorithms [42, 43], has been studied in many works under the shuffle model [6, 7, 12, 24–26]. In this problem, each user $i \in [n] := \{1, \ldots, n\}$ holds an integer $x_i \in \{0, 1, \ldots, U\}$ and the goal is to estimate $\mathrm{Sum}(D) := \sum_i x_i$, where $D = (x_1, \ldots, x_n)$. All of these works for sum estimation under shuffle-DP focus on achieving an error of $O(U/\varepsilon)$. Such an error can be easily achieved under central-DP, where the curator releases the true sum after masking it with a Laplace noise of scale U/ε. In the shuffle-DP model, besides error, another criterion that should be considered is the communication cost. The recent shuffle-DP summation protocol of [24] both matches the error of central-DP and achieves optimal communication. More precisely, it achieves an error that is just $1 + o(1)$ times that of the Laplace mechanism, while each user just sends in expectation $1 + o(1)$ messages, each of a logarithmic number of bits.

However, in real applications (as well as in [24]), U must be set independently of the dataset; to account for all possible datasets, it should be conservatively large. For instance, if we only know that the x_i's are 32-bit integers, then $U = 2^{32} - 1$. Then the error

[1] In Section 1, all stated error guarantees hold with constant probability. We will make the dependency on the failure probability β more explicit in later sections.

Mechanism			Error	Average messages sent by each user
1-D Sum	Prior work	[24]	$O(U/\varepsilon)$	$1+o(1)$
		[7]	$O(U/\varepsilon)$	$O(1)$
		[27] + [24] + [22]	$\tilde{O}\big(\mathrm{Max}(D)\cdot\log^{3.5}U\sqrt{\log(1/\delta)}/\varepsilon\big)$	**Round** 1: $\tilde{O}\big(\log^6 U\cdot\log\big(1/\delta\big)/\varepsilon\big)$ **Round** 2: $1+o(1)$
	Our result	Theorem 5.1	$O\big(\mathrm{Max}(D)\cdot\log\log U/\varepsilon\big)$	$1+o(1)$
	Best result under central model [18]		$O\big(\mathrm{Max}(D)\cdot\log\log U/\varepsilon\big)$	
d-D Sum	Prior work	[27]	$\tilde{O}\big(U_{\ell_2}\sqrt{d\log(n)\log(1/\delta)}/\varepsilon\big)$	$d+\tilde{O}(d^{1.5}\log^{1.5}(1/\delta)/(\varepsilon\sqrt{n}))$
		[27] + [24] +[22]	$\tilde{O}\big(\mathrm{Max}_{\ell_2}(D)\cdot(\sqrt{d\log(nd)\log(1/\delta)}+\log^{3.5}U_{\ell_2}\cdot\sqrt{\log(1/\delta)})/\varepsilon\big)$	**Round** 1: $\tilde{O}\big(\log^6 U_{\ell_2}\cdot\log(1/\delta)/\varepsilon\big)$ **Round** 2: $d+\tilde{O}(d^{1.5}\log^{1.5}(1/\delta)/(\varepsilon\sqrt{n}))$
	Our result	Theorem 6.2	$O\big(\mathrm{Max}_{\ell_2}(D)\cdot\log(d\log U_{\ell_2})\cdot\sqrt{d\log(nd)\log(1/\delta)}/\varepsilon\big)$	$d+\tilde{O}(d^{1.5}\log^{1.5}(1/\delta)/(\varepsilon\sqrt{n}))$
	Best result under central model [17]		$O\big(\mathrm{Max}_{\ell_2}(D)\cdot(\sqrt{d\log(1/\delta)}+\log\log U_{\ell_2})/\varepsilon\big)$	
Sparse Vector Aggregation	Our result	Theorem 7.1	$\tilde{O}(\mathrm{Max}_{\ell_2}(D)\cdot\log d\sqrt{\log(1/\delta)}/\varepsilon)$	$\|x_i\|_1+1+O(d^{1.5}\log d\log^{1.5}(1/\delta)/(\varepsilon n))$
	Best result under central model [17]		$O\big(\mathrm{Max}_{\ell_2}(D)\cdot(\sqrt{\log(1/\delta)}+\log\log U_{\ell_2})/\varepsilon\big)$	

Table 1: Comparison between our results and prior works on sum estimation, high-dimensional sum estimation, and sparse vector aggregation under shuffle model, where we use the absolute error, ℓ_2 error, and ℓ_∞ error as the error metrics. Each message contains $O(\log U+\log d+\log n)$ bits. The mechanism without an indicator for the round runs in a single round. Our communication cost for 1-D Sum requires the condition $n=\omega(\log^2 U)$.

of $O(U/\varepsilon)$ could dwarf the true sum for most datasets. Notice that sometimes some prior knowledge is available, and then a smaller U could be used. For example, if we know that the x_i's are people's incomes, then we may set U as that of the richest person in the world. Such a U is still too large for most datasets as such a rich person seldom appears in most datasets.

Instance-Awareness. The earlier error bound of $O(U/\varepsilon)$ can be shown to be optimal, but only in the worst case. When typical input data are much smaller than U, an *instance-specific* mechanism (and error bound) can be obtained—i.e., a mechanism whose error depends on the largest element of the dataset. In the example of incomes above, an instance-aware mechanism would achieve an error proportional to the actual maximum income in the dataset. This insight has recently been explored under the central model of DP [4, 15, 21, 27, 36, 39].

A widely used technique for achieving instance-specific error bounds under central-DP is the *clipping mechanism* [4, 27, 36, 39]. For some τ, each x_i is clipped to $\mathrm{Clip}(x_i,\tau):=\min(x_i,\tau)$. Then we compute the sum of $\mathrm{Clip}(D,\tau):=\big(\mathrm{Clip}(x_i,\tau)\mid i=1,\ldots,n\big)$ and add a Laplace noise of scale $O(\tau/\varepsilon)$. Note that the clipping introduces a (negative) bias of magnitude up to $\mathrm{Max}(D)\cdot|\{i\in[n]\mid x_i>\tau\}|$, where $\mathrm{Max}(D):=\max_i x_i$. Thus, one should choose a good clipping threshold τ that balances the DP noise and bias. Importantly, this must be done in a DP fashion, and this is where all past investigations on the clipping mechanism have been devoted. In the central model, the best error bound achievable is $O\big(\mathrm{Max}(D)\cdot\log\log U/\varepsilon\big)$ [18].[2] For the real summation problem, such an error bound is considered (nearly) instance-optimal, since any DP mechanism has to incur an error of $\Omega(\mathrm{Max}(D))$ on either D or $D-\{\mathrm{Max}(D)\}$ [45]. The factor of $\log\log U/\varepsilon$ is known as the "optimality ratio". It has been shown that the optimality ratio $O(\log\log U/\varepsilon)$ is the best possible in the case of $\delta=0$ (δ is a privacy parameter, see Section 3.1

[2] [18] achieves an error of $O(\mathrm{Max}(D)\log\log(\mathrm{Max}(D)))$ rather than the cited $O(\mathrm{Max}(D)\log\log U)$. The $\log\log(\mathrm{Max}(D))$ result is more meaningful for the unbounded domain setting where $U=\infty$, but in the shuffle-DP model, there is currently no known method can handle the unbounded domain case for any problem, including sum estimation. Our proposed mechanism also only supports the bounded domain case. Therefore, for simplicity, we ignored this minor difference and just cited the $\log\log U$ result.

for more details) [18]. Under the case of $\delta > 0$, it is still an open question whether that ratio is optimal or not. Notably, in the literature [5, 15, 27], a polylog optimality ratio is often considered satisfactory enough and is named as "instance-optimal" and so far no known DP mechanism has a better optimality ratio.

As suggested in [27], the clipping mechanism can be easily implemented in the shuffle model as well, but requiring two rounds. The first round finds τ. Then we broadcast τ to all users. In the second round, we invoke a summation protocol, e.g., the one in [24], on $\mathrm{Clip}(D, \tau)$. Two-round protocols are generally undesirable, not only because of the extra latency and coordination overhead, but also because they leak some information to the users (τ in this case, which is an approximation of $\mathrm{Max}(D)$). Note that the shuffle model, in its strict definition, only allows one-way messages from users to the analyzer (through the shuffler), so the users should learn nothing from each other. Moreover, the central-DP mechanism for finding the optimal τ [18] does not work in the shuffle model. Instead, [27] uses the complicated range-counting protocol of [22]. This results in a sum estimation protocol that runs in two rounds, having an error of $\tilde{O}(\mathrm{Max}(D) \cdot \log^{3.5} U \sqrt{\log(1/\delta)})$[3], and sends $\tilde{O}\big(\log^6 U \log(1/\delta)/\varepsilon\big)$ messages per user. Thus, this protocol is of only theoretical interest; indeed, no experimental results are provided in [27].

Problem Statement. Does there exist a practical, *single-round* protocol for sum estimation under shuffle-DP that simultaneously: (1) matches the optimal central-DP error of $O(\mathrm{Max}(D) \cdot \log\log U/\varepsilon)$, and (2) requires $1 + o(1)$ messages per user?

1.1 Our results

We answer this question in the affirmative, by presenting a new single-round shuffle-DP clipping protocol for the sum estimation problem. At the core of our protocol is a technique that finds the optimal τ and computes the noisy clipped sum using τ at the same time. This appears impossible, as the second step relies on the information obtained from the first. Our idea is to divide the data into a set of disjoint parts and do the estimations for each part independently. This ensures we only pay the privacy and communication cost of one since each element will only be involved in one estimation. Based on these estimations, we can compute the noisy clipped sums for all the clipping thresholds $\tau = 1, 2, 4, \ldots, U$. Meanwhile, we show that these noisy estimations already contain enough information to allow us to decide which τ is the best. Besides solving sum aggregation, we show that using this protocol as a building block or deriving a variant of this idea can achieve state-of-the-art privacy-utility-communication tradeoffs for two other important summation problems.

1.1.1 Contributions. Our contributions are threefold, summarized in Table 1 and below:

(1) Sum estimation. For the vanilla sum estimation problem[4], we present a single-round protocol (Section 4 and 5) that achieves the optimal error of $O(\mathrm{Max}(D) \cdot \log\log U/\varepsilon)$, which improves the error rate of $O(U/\varepsilon)$ from [24] exponentially in U. More importantly, we have $1 + o(1)$ messages per client when $n = \omega(\log^2 U)$ (see Theorem 5.1 for more details), a criterion typically met in most common regimes.

(2) High-dimensional sum estimation. Next, we consider the sum estimation problem in high dimensions, which has been extensively studied in the machine learning literature under central DP [9, 27, 33]. Here, each x_i is a vector with integer coordinates taken from the d-dimensional ball of radius U_{ℓ_2} centered at the origin, and we wish to estimate $\mathrm{Sum}(D)$ with small ℓ_2 error.

The literature for this problem exhibits similar patterns to the 1D summation problem. Under central-DP, the state-of-the-art mechanism achieves an error proportional to $\sqrt{d} \cdot \mathrm{Max}_{\ell_2}(D)$, where $\mathrm{Max}_{\ell_2}(D) := \max_i \|x_i\|_2$ [17]. Generalizing the argument in the 1D case, $\mathrm{Max}_{\ell_2}(D)$ is an instance-specific lower bound for d-dimensional sum estimation, and the factor $\sqrt{d}$ is also optimal [27]. For shuffle-DP, [27] presented a one-round protocol achieving an error proportional to $\sqrt{d} \cdot U_{\ell_2}$ (i.e., not instance-specific) with $d + \tilde{O}(d^{1.5} \log^{1.5}(1/\delta)/(\varepsilon\sqrt{n}))$ message complexity. [27] observed that a two-round clipping mechanism can be used to achieve an instance-specific error, but as in the 1D case, this incurs high polylogarithmic factors in both the optimality ratio and the message complexity.

In Section 6, we propose our single-round protocol for high-dimensional summation by treating our 1D summation protocol as a black box: we first do a rotation over the space, and invoke our 1D protocol in each dimension. This approach has the same instance-optimal error as the central-DP up to polylogarithmic factors, and achieves the same message complexity as the existing worst-case error protocol.

(3) Sparse vector aggregation. As the third application of our technique, we study the sparse vector aggregation problem. This problem is the same as the high-dimensional sum estimation problem, except that (1) each x_i is now a binary vector in $\{0, 1\}^d$, (2) the x_i's are sparse, i.e., $\|x_i\|_1 = \|x_i\|_2^2 \ll d$, and (3) we aim at an ℓ_∞ error. This problem is also known as the *frequency estimation* problem under *user-level DP*, where each user contributes a set of elements from $[d]$, and we wish to estimate the frequency of each element. For this problem, people are more interested in the ℓ_∞ error since we would like each frequency estimate to be accurate.

Under central-DP, the state-of-the-art algorithm already achieves ℓ_∞ error with $\mathrm{Max}_{\ell_2}(D)$ [17]. Under shuffle-DP, there is no known prior work on this problem. Our high-dimensional sum estimation protocol can solve the problem, but it does not yield a good ℓ_∞ error and incurs a message complexity of at least d per user, even if the user has only a few elements.

In Section 7, we present our one-round sparse vector aggregation protocol. This protocol can be regarded as a variant of our 1D sum protocol, where we divide the data per its sparsity. This has error of $\mathrm{Max}_{\ell_2}(D)$, and sends $\|x_i\|_1 + 1 + O(d^{1.5} \log d \log^{1.5}(1/\delta)/(\varepsilon n))$ messages for user i. Note that this ℓ_∞ error implies an ℓ_2 error that is $\sqrt{d}$ times larger (but not vice versa), so it also matches the high-dimensional sum estimation protocol in terms of ℓ_2 error. Furthermore, it exploits the sparsity of each x_i in the message complexity. It remains an interesting open problem if the extra $O(d^{1.5} \log d \log^{1.5}(1/\delta)/(\varepsilon n))$ term can be reduced.

[3] For any function f, $\tilde{O}(f) := f \cdot \mathrm{polylog}(f)$.

[4] Note that although we focus on the integer domain $\{0, \ldots, U\}$, our protocol easily extends to the real summation problem, where each value x_i is a real number from $[0, 1]$, by discretizing $[0, 1]$ into U buckets of width $1/U$. This incurs an extra additive error of $O(n/U)$. Thanks to the double logarithmic dependency on U, we could set U sufficiently large (e.g., $U = n^{\log n}$) to make this additive error negligible while keeping the $O(\mathrm{Max}(D) \cdot \log\log n/\varepsilon)$ error bound.

2 Related Work

For sum estimation under central-DP, the worst-case optimal error $O(U/\varepsilon)$ can be easily achieved by the *Laplace mechanism*. Many papers have studied how to obtain instance-specific error, i.e., an error depending on $\mathrm{Max}(D)$ [4, 5, 14, 15, 18, 21, 27, 36, 39]. Most of these works rely on the clipping mechanism [4, 5, 14, 15, 18, 27, 36, 39]. Similarly, for the high-dimensional sum aggregation problem, existing approaches have achieved instance-specific error by using the clipping mechanism [9, 17, 27, 33, 34]; such mechanisms also yield an ℓ_∞ error of $\mathrm{Max}_{\ell_2}(D)$ for sparse vector aggregation.

In the shuffle-DP setting, for sum estimation, two settings are used. In the *single-message* setting, each user sends one message. Here, [6] achieve an error of $O(Un^{1/6})$ and further show that this is worst-case optimal. In the *multi-message* setting, where each user is allowed to send multiple messages, most prior works try to achieve the worst-case optimal error while minimizing communication costs. Cheu et al. [12] first achieved an error of $O(U\sqrt{\log(1/\delta)}/\varepsilon)$ with $O(\sqrt{n})$ messages sent per user. Then, [26] achieved the same error but reduced the number of messages per user to $O(\log(n))$. [25] and [7] further improved the error to $O(U/\varepsilon)$ with constant messages. Recently, [24] reduced that communication to $1 + o(1)$ messages per user. We aim to obtain instance-optimal error, while keeping the $1 + o(1)$ per-client message complexity of [24].

3 Preliminaries

We use the following notation: $\mathbb{Z}$ is the domain of all integers, $\mathbb{Z}_{\geq 0}$ non-negative integers, and $\mathbb{Z}_+$ positive integers. Let $D = (x_1, x_2, \ldots, x_n)$, where user i holds an integer x_i from $\{0\} \cup [U]$. For simplicity, we assume that U is a power of 2. We would like to estimate $\mathrm{Sum}(D) = \sum_i x_i$. For brevity, we often interpret D as a multiset, and $D \cap [a, b]$ denotes the multiset of elements of D that fall into $[a, b]$. We introduce two auxiliary functions: $\mathrm{Count}(D)$ is the cardinality of D (duplicates are counted); $\mathrm{Max}(D, k)$ is the kth largest value of D, or more precisely,

$$\mathrm{Max}(D, k) := \max\left\{t : \mathrm{Count}(D \cap [t, U]) \geq k\right\}.$$

3.1 Differential Privacy

Definition 1 (Differential privacy). *For $\varepsilon, \delta > 0$, an algorithm $\mathcal{M} : \mathcal{X}^n \to \mathcal{Y}$ is (ε, δ)-differentially private (DP) if for any neighboring instances $D \sim D'$ (i.e., D and D' differ by a single element), $\mathcal{M}(D)$ and $\mathcal{M}(D')$ are (ε, δ)-indistinguishable, i.e., for any subset of outputs $Y \subseteq \mathcal{Y}$,*

$$\Pr[\mathcal{M}(D) \in Y] \leq e^\varepsilon \cdot \Pr[\mathcal{M}(D') \in Y] + \delta.$$

The privacy parameter ε is typically between 0.1 and 10, while δ should be much smaller than $1/n$.

All DP models can be captured by the definition above by appropriately defining $\mathcal{M}(D)$. In *central*-DP, $\mathcal{M}(D)$ is just the output of data curator; in *local*-DP, the local randomizer $\mathcal{R} : \mathcal{X} \to \mathcal{Z}$ outputs a message in $\mathcal{Z}$, and $\mathcal{M}(D)$ is defined as the vector $(\mathcal{R}(x_1), \mathcal{R}(x_2), \ldots, \mathcal{R}(x_n))$; in *shuffle*-DP, $\mathcal{R} : \mathcal{X} \to \mathbb{N}^{\mathcal{Z}}$ outputs a multiset of messages and $\mathcal{M}(D)$ is the (multiset) union of the $\mathcal{R}(x_i)$'s.

DP enjoys the following properties regardless of the specific model:

Lemma 3.1 (Post Processing [19]). *If $\mathcal{M}$ satisfies (ε, δ)-DP and $\mathcal{M}'$ is any randomized mechanism, then $\mathcal{M}'(\mathcal{M}(D))$ satisfies (ε, δ)-DP.*

Lemma 3.2 (Sequential Composition [19]). *If $\mathcal{M}$ is a (possibly adaptive) composition of differentially private mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_k$, where each $\mathcal{M}_i$ satisfies (ε, δ)-DP, then $\mathcal{M}$ satisfies (ε', δ')-DP, where*

(1) $\varepsilon' = k\varepsilon$ and $\delta' = k\delta$; [Basic Composition]

(2) $\varepsilon' = \varepsilon\sqrt{2k\log\frac{1}{\delta''}} + k\varepsilon(e^\varepsilon - 1)$ and $\delta' = k\delta + \delta''$ for any $\delta'' > 0$. [Advanced Composition]

Lemma 3.3 (Parallel Composition [37]). *Let $\mathcal{X}_1, \ldots, \mathcal{X}_k$ each be a subdomain of $\mathcal{X}$ that are pairwise disjoint, and let each $\mathcal{M}_i : \mathcal{X}_i^n \to \mathcal{Y}$ be an (ε, δ)-DP mechanism. Then $\mathcal{M}(D) := (\mathcal{M}_1(D \cap \mathcal{X}_1), \ldots, \mathcal{M}_k(D \cap \mathcal{X}_k))$ also satisfies (ε, δ)-DP.*

3.2 Sum Estimation in Central-DP

In central-DP, one of the most widely used DP mechanisms is the Laplace mechanism:

Lemma 3.4 (Laplace Mechanism). *Given any query $Q : \{0, 1, \ldots, U\}^n \to \mathbb{R}$, the global sensitivity is defined as $\mathrm{GS}_Q = \max_{D\sim D'} \left|Q(D) - Q(D')\right|$. The mechanism $\mathcal{M}(D) = Q(D) + \mathrm{GS}_Q/\varepsilon \cdot \mathrm{Lap}(1)$ preserves $(\varepsilon, 0)$-DP, where $\mathrm{Lap}(1)$ denotes a random variable drawn from the unit Laplace distribution.*

For the sum estimation problem, $\mathrm{GS}_Q = U$, which means that the Laplace mechanism yields an error of $O(U/\varepsilon)$. As mentioned in Section 1, although such an error bound is already worst-case optimal, it is not very meaningful when typical data are much smaller than $\mathrm{GS}_Q = U$.

Clipping mechanism The clipping mechanism has been widely used to achieve an instance-specific error bound depending on $\mathrm{Max}(D)$. It first finds a clipping threshold τ, and then applies the Laplace mechanism on $\mathrm{Clip}(D, \tau)$ with $\mathrm{GS}_Q = \tau$. The clipping introduces a bias of $\mathrm{Max}(D) \cdot |\{i \in [n] \mid x_i > \tau\}|$, so it is important to choose a τ that balances the DP noise and bias. In the central model, the best result is [18], which finds a τ between $\mathrm{Max}(D, \log\log U/\varepsilon)$ and $2 \cdot \mathrm{Max}(D)$. Plugging this τ into the clipping mechanism yields an error of $O\big(\mathrm{Max}(D) \cdot \log\log U/\varepsilon\big)$.

3.3 Sum Estimation in Shuffle-DP

In shuffle-DP, the state-of-the-art protocol for sum estimation is proposed by Ghazi et al. [24] and achieves an error that is $1 + o(1)$ times that of the Laplace mechanism and each user sends $1 + o(1)$ messages in expectation. Both are optimal (in the worst-case sense) up to lower-order terms. We briefly describe their protocol below as it will also be used in our protocols.

Each user i first sends x_i if it is non-zero. To ensure privacy, users additionally send noises drawn from $\{-U, \ldots, U\} - \{0\}$ based on an ingeniously designed distribution $\mathcal{P}$, such that most noises cancel out while the remaining noises add up to a random variable drawn from the discrete Laplace distribution[5] with scale $(1 - \lambda)\varepsilon$ for some parameter λ. The cancelled out noises are meant to flood the messages containing the true data x_i so as to ensure $(\lambda\varepsilon, \delta)$-DP. Thus, the entire protocol satisfies (ε, δ)-DP.

[5]The Discrete Laplace distribution with scale s has a probability mass function $\frac{1-e^{-1/s}}{1+e^{-1/s}} \cdot e^{-|k|/s}$ for each $k \in \mathbb{Z}$.

Algorithm 1: Randomizer of BaseSumDP [24].

Input: $x_i, \varepsilon, \delta, n, U, \lambda, \zeta$

$U', x_i' \leftarrow U, x_i$;

if $U > \sqrt{n/\zeta}$ **then**

  /* Randomized rounding of x_i */

  $B \leftarrow \lceil U/(\sqrt{n/\zeta}) \rceil$;

  $U' \leftarrow \sqrt{n}$;

  $p \leftarrow \lceil x_i/B \rceil - x_i/B$;

  $x_i' \leftarrow \begin{cases} \lceil x_i/B \rceil & \text{with probability } p \\ \lceil x_i/B \rceil - 1 & \text{with probability } 1-p \end{cases}$;

$S_i \leftarrow \{\}$;

if $x_i' \neq 0$ **then**

  Add x_i' into S_i;

end

/* Sample a vector from $\mathcal{P}$ */

$(z^{-U'}, \ldots, z^{-1}, z^1, \ldots, z^{U'}) \sim \mathcal{P}(\varepsilon, \delta, n, \lambda, U')$;

for $j \leftarrow -U', -U'+1, \ldots, -1, 1, \ldots, U'-1, U'$ **do**

  Add z_j copies of j into S_i;

end

Send S_i;

Algorithm 2: Analyzer of BaseSumDP [24].

Input: $R = \cup_i S_i$ with S_i from user i, $\varepsilon, \delta, n, U, \lambda, \zeta$

$\widetilde{\text{Sum}}(D) \leftarrow \sum_{y \in R} y$;

if $U > \sqrt{n}$ **then**

  $B \leftarrow \lceil U/(\sqrt{n/\zeta}) \rceil$;

  $\widetilde{\text{Sum}}(D) \leftarrow \widetilde{\text{Sum}}(D) \cdot B$;

return $\widetilde{\text{Sum}}(D)$;

For a large U, their protocol should be applied after reducing the domain size to $\sqrt{n/\zeta}$ for some $\zeta = o(1)$. More precisely, we first randomly round each x_i to a multiple of $\frac{U}{\sqrt{n/\zeta}}$. This introduces an additional error of $O(\sqrt{\zeta}U)$, which is a lower-order term in the error bound. Meanwhile, it reduces the noise messages to $O(\log^2 n \log(1/\delta)/(\varepsilon\lambda\sqrt{\zeta n})) = o(1)$. The detailed randomizer and analyzer are given in Algorithm 1 and 2. The analyzer obviously runs in $O(n)$ time, and they show how to implement the randomizer in time $O(\min(U, \sqrt{n}))$. The following lemma summarizes their protocol:

Lemma 3.5. *Given any $\varepsilon > 0$, $\delta > 0$, n, U, any λ and any ζ,* BaseSumDP *solves the sum estimation problem under shuffle-DP with the following guarantees:*

(1) The messages received by the analyzer satisfy (ε, δ)-DP;

(2) With probability at least $1-\beta$, the error is bounded by $(\zeta + \frac{1}{\varepsilon(1-\lambda)}) \cdot U \ln(2/\beta)$;

(3) In expectation, each user sends

$$\mathbf{I}(x_i \neq 0) + O\Big(\frac{\log^2 n \log(1/\delta)}{\varepsilon\lambda\sqrt{\zeta n}}\Big)$$

messages with each containing $O\Big(\min\big(\log n, \log U\big)\Big)$ bits.

Remark: Setting $\lambda, \zeta = o(1)$ yields an error of $1 + o(1)$ error and $1 + o(1)$ messages. In this paper, we will invoke BaseSumDP with $\lambda = 0.1$ and $\zeta = \min(0.1, \frac{0.1}{\varepsilon})$. With this setting, the error bound is $1.3 \cdot U \ln(2/\beta)/\varepsilon$ and the message complexity is still $1 + o(1)$.

Combining the clipping technique with BaseSumDP immediately leads to a two-round protocol in shuffle-DP: In round one, we find τ; in round two, we invoke BaseSumDP on $\text{Clip}(D, \tau)$. This approach was suggested in [27]. However, since the optimal central-model τ-finding algorithm [18] cannot be used in the shuffle model, [27] instead used the complicated range-counting protocol of [22] to find a τ such that

$$\text{Max}(D, k) \leq \tau \leq \text{Max}(D) \tag{1}$$

for some $k = \tilde{O}\big(\log^{3.5} U \sqrt{\log(1/\delta)}/\varepsilon\big)$. Plugging this τ into the clipping mechanism yields an error of $O(\text{Max}(D) \cdot k)$. The message complexity of their protocol is $\tilde{O}\big(\log^6 U \log(1/\delta)/\varepsilon\big)$, dominated by the range-counting protocol [22].

4 A Straw-man One-Round Protocol

We first present a simple one-round shuffle-DP protocol for the sum estimation problem. Although it does not achieve either the desired error or communication rates from Section 1, it provides a foundation for our final solution.

4.1 Domain Compression

Our first observation is it is not necessary to consider all possible $\tau \in \{0\} \cup [U]$. Instead, we only need to consider $\tau \in \{0, 1, 2, 4, \ldots, U\}$. Specifically, we map the dataset D to[6]

$$\overline{D} = \Big\{\lceil \log(x_1) \rceil, \lceil \log(x_2) \rceil, \ldots, \lceil \log(x_n) \rceil\Big\}.$$

Note that this compresses the domain from $\{0\} \cup [U]$ to $\{-1, 0\} \cup [\log U]$. After compressing the domain size from $U+1$ to $\log U + 2$, running the round-one protocol of [27] on $\overline{D}$ can now find a $\bar{\tau}$ such that

$$\text{Max}(\overline{D}, k) \leq \bar{\tau} \leq \text{Max}(\overline{D}), \tag{2}$$

for some $k = \tilde{O}\big((\log\log U)^{3.5} \sqrt{\log(1/\delta)}/\varepsilon\big)$.

In the second round, we use $\tau = 2^{\bar{\tau}}$ as the clipping threshold and invoke BaseSumDP on D. Note that we always have $\tau \leq 2 \cdot \text{Max}(D)$. Furthermore, D contains at most k elements that are strictly larger than τ, so the clipping mechanism yields an error of $O(\text{Max}(D) \cdot k)$.

In addition to reducing the error, domain compression also reduces the message complexity of the first round from $\tilde{O}((\log U)^6 \log(1/\delta)/\epsilon)$ to $\tilde{O}((\log\log U)^6 \log(1/\delta)/\varepsilon)$. The message complexity of the second round is the same as that of BaseSumDP, i.e., $1 + o(1)$.

4.2 Try All Possible τ

The domain compression technique narrows down the possible values for τ to just $\log U + 2$. This allows us to try all possible τ simultaneously. We can run $\log U + 2$ instances of BaseSumDP, each with a different $\tau = 0, 1, 2, 4, \ldots, U$. That is, each client x_i runs BaseSumDP $\log U + 2$ times, each time clipping its data x_i with a different threshold before the randomizer protocol, and the analyzer computes $\log U + 2$ different sums, one for each threshold.

[6] All log have base 2. Specially, define $\log(0) := -1$ and $2^{-1} := 0$.

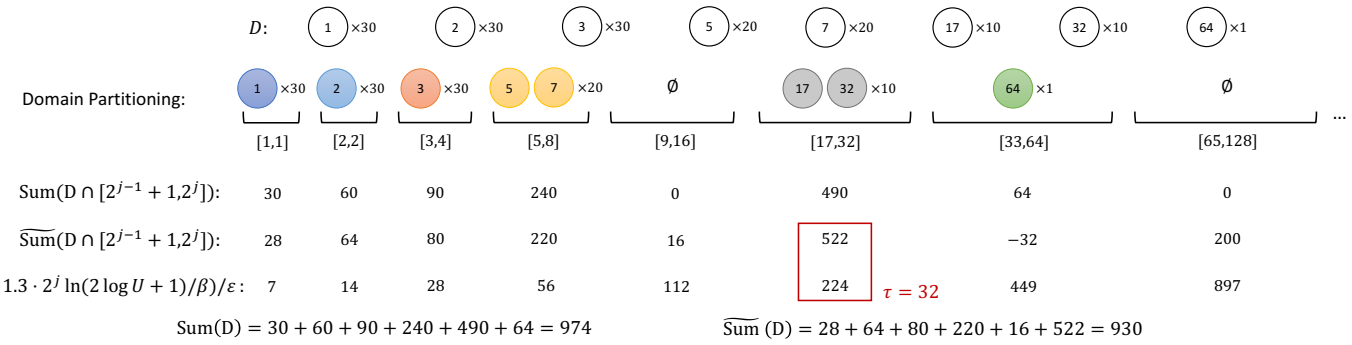

Figure 1: An illustration of our protocol for sum estimation. $U = 2^{10}$, $\varepsilon = 1$, **and** $\beta = 0.1$.

All these are done concurrently to the protocol for finding $\bar{\tau}$. Finally, the analyzer will return the output of the BaseSumDP instance that has been executed with the correct $\tau = 2^{\bar{\tau}}$. However, the $\log U + 2$ instances of BaseSumDP must split the privacy budget using sequential composition[7]. More precisely, we run each instance with privacy budget $\varepsilon/(2(\log U + 2))$, while reserving the other $\varepsilon/2$ privacy budget for finding $\bar{\tau}$. Thus, the BaseSumDP instance with clipping threshold τ must inject a DP noise of scale $O(\tau \log U/\varepsilon)$. The clipping still introduces a bias of $O(\mathrm{Max}(D) \cdot k)$, so the total error becomes $\tilde{O}(\mathrm{Max}(D) \cdot (\log U + \sqrt{\log(1/\delta)})/\varepsilon)$.

In terms of the message complexity, these $O(\log U)$ BaseSumDP instances together send $O(\log U)$ messages per user, in addition of the $O((\log\log U)^6 \log(1/\delta)/\varepsilon)$ messages for finding $\bar{\tau}$. So the message complexity is now $O(\log U + \log(1/\delta)/\varepsilon)$.

Simple tweaks to this straw-man solution do not give the desired properties. For instance, one may compress the domain to $\{0, 1, c, c^2, \dots\}$ for some $c \geq 2$. This lowers the message complexity to $O(\log_c U + \log(1/\delta)/\varepsilon)$ and each BaseSumDP instance has a privacy budget of $\varepsilon/\log_c U$. But now τ may be as large as $c \cdot \mathrm{Max}(D)$, so the error increases to $O(\mathrm{Max}(D) \cdot c \log_c U/\varepsilon)$. Thus, new ideas are needed to achieve our desiderata in a one-round protocol.

5 Our Protocol

In this section, we present our single-round protocol that achieves both optimal error and message complexity.

5.1 Domain Partitioning

We see that the $O(\log U)$ factor blowup in the error of the straw-man solution is due to the $\log U + 2$ BaseSumDP instances splitting the privacy budget using sequential composition. In order to avoid the splitting, our idea is to partition the domain and then use parallel composition. More precisely, we partition the domain into $\log U + 1$ disjoint sub-domains: $[1, 1], [2, 2], [3, 4], [5, 8], \dots, [U/2 + 1, U]$. It is clear that, for any D, we have

$$\mathrm{Sum}(D) = \sum_{j=0}^{\log U} \mathrm{Sum}\big(D \cap [2^{j-1} + 1, 2^j]\big).$$

[7]"Sequential composition" refers to privacy; all these instances are still executed in parallel in one round.

Furthermore, for any $\tau = 1, 2, 4, \dots, U$, the clipped sum is precisely the sum in the first $\log \tau + 1$ sub-domains:

$$\mathrm{Sum}\Big(\mathrm{Clip}(D, \tau)\Big) = \sum_{j=0}^{\log \tau} \mathrm{Sum}\big(D \cap [2^{j-1} + 1, 2^j]\big).$$

Therefore, it suffices to estimate $\mathrm{Sum}\big(D \cap [2^{j-1} + 1, 2^j]\big)$ for each $j \in 0, 1, \dots, \log U$. Importantly, since these sub-domains are disjoint, parallel composition can be applied and we can afford a privacy budget of ε on each sub-domain. We thus run a BaseSumDP instance on each $D \cap [2^{j-1} + 1, 2^j]$, which returns an estimate

$$\widetilde{\mathrm{Sum}}\big(D \cap [2^{j-1} + 1, 2^j]\big) := \mathrm{Sum}\big(D \cap [2^{j-1} + 1, 2^j]\big) + \mathrm{Lap}(2^j/\varepsilon).$$

Then for any $\tau = 1, 2, 4, \dots, U$, we estimate $\mathrm{Sum}(\mathrm{Clip}(D, \tau))$ as

$$\widetilde{\mathrm{Sum}}\big(D \cap [1, \tau]\big) = \sum_{j=0}^{\log \tau} \widetilde{\mathrm{Sum}}\big(D \cap [2^{j-1} + 1, 2^j]\big).$$

Importantly, the total noise level in $\widetilde{\mathrm{Sum}}\big(D \cap [1, \tau]\big)$ is still bounded by $O(\tau/\varepsilon)$, as the noise levels from the sub-domains form a geometric series. This ensures that the DP noise is bounded by $O(\mathrm{Max}(D)/\varepsilon)$, as long as we choose a $\tau \leq 2 \cdot \mathrm{Max}(D)$.

Meanwhile, this domain partitioning lowers the total message complexity of all the $\log U + 1$ BaseSumDP instances to $1 + o(1)$. This is because after domain partitioning, each user has a nonzero input only in one sub-domain, and the BaseSumDP protocol sends out $o(1)$ message when $x_i = 0$.

5.2 Finding τ with No Extra Cost

It remains to deal with the impractical τ-selection protocol used in [27], which has a message complexity of $O((\log\log U)^6 \log(1/\delta)/\varepsilon)$ and finds a τ such that

$$\mathrm{Max}(D, k) \leq \tau \leq 2 \cdot \mathrm{Max}(D), \tag{3}$$

for some $k = \tilde{O}\big((\log\log U)^{3.5} \sqrt{\log(1/\delta)}/\varepsilon\big)$. Recall that the bias introduced by the clipping is $O(\mathrm{Max}(D) \cdot k)$.

It turns out that we can find a τ that achieves the optimal $k = O(\log\log U/\varepsilon)$ with no extra cost at all! To illustrate the idea, first consider the non-private setting where we have access to the exact values of $\mathrm{Sum}\big(D \cap [2^{j-1} + 1, 2^j]\big)$ for each $j = 0, 1, 2, \dots, \log U$. Then it is easy to see that the last j on which $\mathrm{Sum}\big(D \cap [2^{j-1} + 1, 2^j]\big) > 0$

Algorithm 3: Randomizer of SumDP.

Input: x_i, ε, δ, β, n, U

for $j \leftarrow 0, 1, 2, \ldots, \log U$ **do**

  /* The messages for estimating $\mathrm{Sum}\big(D \cap [2^{j-1}+1, 2^j]\big)$ */

  $S_i^{[2^{j-1}+1,2^j]} \leftarrow \mathrm{Randomizer}\Big(x_i \cdot \mathbf{I}\big(x_i \in [2^{j-1}+1, 2^j]\big), \varepsilon, \delta, n, 2^j\Big)$ of BaseSumDP

end

Send $\big\{S_i^{[2^{j-1}+1,2^j]}\big\}_{j\in\{0,1,2,\ldots,\log U\}}$;

Algorithm 4: Analyzer of SumDP.

Input: $\Big\{R^{[2^{j-1}+1,2^j]} = \cup_i S_i^{[2^{j-1}+1,2^j]}\Big\}_{j\in\{0,1,2,\ldots,\log U\}}$ with $S_i^{[2^{j-1}+1,2^j]}$ from user i, ε, δ, β, n, U

$\tau \leftarrow 0$;

for $j \leftarrow 0, 1, 2, \ldots, \log U$ **do**

  $\widetilde{\mathrm{Sum}}\big(D \cap [2^{j-1}+1, 2^j]\big) \leftarrow \sum_{y\in R^{[2^{j-1}+1,2^j]}} y$;

  /* Set $\tau = 2^j$ for last j passing the condition of (4) */

  if $\widetilde{\mathrm{Sum}}\big(D \cap [2^{j-1}+1, 2^j]\big) > 1.3 \cdot 2^j \cdot \ln\big(2(\log U + 1)/\beta\big)/\varepsilon$ **then**

    $\tau \leftarrow 2^j$;

  end

end

$\widetilde{\mathrm{Sum}}(D) \leftarrow \sum_{j\in\{0,1,2,\ldots,\log\tau\}} \widetilde{\mathrm{Sum}}\big(D \cap [2^{j-1}+1, 2^j]\big)$;

return $\widetilde{\mathrm{Sum}}(D)$

yields a $\tau = 2^j$ such that $\mathrm{Max}(D) \le \tau \le 2 \cdot \mathrm{Max}(D)$, i.e., we can achieve (3) with $k = 1$. In the private setting, however, due to having access only to the noisy estimates $\widetilde{\mathrm{Sum}}\big(D \cap [2^{j-1}+1, 2^j]\big)$, we may easily overshoot: With probability at least $1/2$, the last sub-domain has $\widetilde{\mathrm{Sum}}\big(D \cap [\tau/2 - 1, \tau]\big) > 0$ (even if it is empty), which would set $\tau = U$.

To prevent this overshooting, our idea is to use a higher bar. Instead of finding the last j on which $\mathrm{Sum}\big(D \cap [2^{j-1}+1, 2^j]\big) > 0$, we change the condition to

$$\widetilde{\mathrm{Sum}}\big(D \cap [2^{j-1}+1, 2^j]\big) > 1.3 \cdot 2^j \cdot \ln\big(2(\log U + 1)/\beta\big)/\varepsilon. \quad (4)$$

The RHS of (4) follows from the error bound of BaseSumDP (see the remark after Lemma 3.5), where we replace U with 2^j (since the largest value in this sub-domain is 2^j) and replace β with $\beta/(\log U + 1)$, so that when this sub-domain is empty, (4) happens with probability at most $\beta/(\log U + 1)$. Then by a union bound, with probability at least $1 - \beta$, none of the empty sub-domains passes the condition (4). In this case, we are guaranteed to find a $\tau = 2^j \le 2 \cdot \mathrm{Max}(D)$, namely, we will not overshoot. Meanwhile, we can also show that we will not undershoot too much, either. More precisely, with probability at least $1 - \beta$, there are at most $O(\log(\log U/\beta)/\varepsilon)$ elements greater than 2^j. Therefore, plugging $\tau = 2^j$ into the clipping mechanism yields the optimal central-DP error of $O(\mathrm{Max}(D) \cdot \log(\log U/\beta)/\varepsilon)$.

To summarize, our final protocol works as follows. After domain partitioning, each user i executes an instance of BaseSumDP for every sub-domain $[2^{j-1}+1, 2^j]$ with the input $x_i \cdot \mathbf{I}\big(x_i \in [2^{j-1}+1, 2^j]\big)$ and the whole privacy budget ε, δ. As all the messages are shuffled together, they need to identify themselves with which BaseSumDP instance they belong to. This just requires extra $O(\log\log U)$ bits. From the perspective of the analyzer, based on the received messages, we compute $\widetilde{\mathrm{Sum}}\big(D \cap [2^{j-1}+1, 2^j]\big)$ for each $j \in \{0, 1, \ldots, \log U\}$. Then, τ is set to 2^j for the last j passing condition (4). Finally, we sum up all $\widetilde{\mathrm{Sum}}\big(D \cap [2^{j-1}+1, 2^j]\big)$ for $j \le \log\tau$. The detailed algorithms for the randomizer and analyzer are presented in Algorithm 3 and Algorithm 4. Besides, we give an example to demonstrate the protocol in Figure 1.

Theorem 5.1. *Given any $\varepsilon > 0$, $\delta > 0$, $n \in \mathbb{Z}_+$, and $U \in \mathbb{Z}_+$, for any $D \in [U]^n$,* SumDP *achieves the following:*

(1) The messages received by the analyzer preserves (ε, δ)-DP;

(2) With probability at least $1 - \beta$, the error is bounded by

$$O\Big(\mathrm{Max}(D) \cdot \log(\log U/\beta)/\varepsilon\Big);$$

(3) In expectation, each user sends

$$1 + O\big(\log U \cdot \log^2 n \cdot \log(1/\delta)/(\varepsilon\sqrt{n})\big)$$

messages with each containing $O\Big(\min\big(\log n, \log U\big)\Big)$ bits.

Proof. For privacy, invoking Lemma 3.5, we have that for every $j \in \{0, 1, 2, \ldots, \log U\}$, $R^{[2^{j-1}+1,2^j]}$ preserves (ε, δ)-DP. Given that each x_i only has an impact on a single $R^{[2^{j-1}+1,2^j]}$, it follows that the collection $\big\{R^{[2^{j-1}+1,2^j]}\big\}_{j\in 0,1,2,\ldots,\log U}$ preserves (ε, δ)-DP.

For utility, Lemma 3.5 ensures that for any $j \in \{0, 1, 2, \ldots, \log U\}$, with probability at least $1 - \frac{\beta}{\log U + 2}$, we have

$$\begin{aligned}&\Big|\widetilde{\mathrm{Sum}}\big(D \cap [2^{j-1}+1, 2^j]\big) - \mathrm{Sum}\big(D \cap [2^{j-1}+1, 2^j]\big)\Big| \\ \le& 1.3 \cdot 2^j \cdot \ln\big(2(\log U + 2)/\beta\big)/\varepsilon. \end{aligned} \quad (5)$$

Aggregating probabilities across all j yields that, with probability at least $1-\beta$, (5) holds across all j.

First, (5) implies τ will not surpass $\left\lceil \log\big(\mathrm{Max}(D)\big)\right\rceil$:

$$\tau \le \left\lceil \log\big(\mathrm{Max}(D)\big)\right\rceil.$$

Combining this with (5), we have

$$\begin{aligned}&\Big|\sum_{j=0}^{\log(\tau)}\big(\widetilde{\mathrm{Sum}}(D\cap[2^{j-1}+1,2^j])-\mathrm{Sum}(D\cap[2^{j-1}+1,2^j])\big)\Big|\\ =&O\Big(\mathrm{Max}(D)\cdot\log(\log U/\beta)/\varepsilon\Big).\end{aligned}\tag{6}$$

Meanwhile, with (5), we also have that all sub-domains over τ will not contain too many elements: for any $j>\log(\tau)$,

$$\mathrm{Sum}\big(D\cap[2^{j-1}+1,2^j]\big)\le 2.6\cdot 2^j\cdot\ln\big(2(\log U+2)/\beta\big)/\varepsilon,$$

which sequentially deduces

$$\begin{aligned}&\sum_{j=\log\tau+1}^{\log U}\mathrm{Sum}\big(D\cap[2^{j-1}+1,2^j]\big)\\ =&O\Big(\mathrm{Max}(D)\cdot\log(\log U/\beta)/\varepsilon\Big).\end{aligned}\tag{7}$$

Finally, (6), and (7) lead to our desired statement for the utility.

The statement for communication directly follows from Lemma 3.5 and the observation that each x_i uniquely corresponds to a single interval $[2^{j-1}+1,2^j]$. □

Additionally, each randomizer incurs a computational cost of $O\big(\log U\cdot\min(U,\sqrt{n})\big)$ and each analyzer operates with a running time of $O(n)$.

6 High-Dimensional Sum Estimation

In this section, we consider the high-dimensional scenario, i.e., each x_i is a d-dimensional vector in $\mathbb{Z}^d$ with ℓ_2 norm bounded by some given (potentially large) U_{ℓ_2}. Thus, D can also be thought of as an $n\times d$ matrix. Let $\mathrm{Max}_{\ell_2}(D):=\max_{x_i\in D}\|x_i\|_2$ be the maximum ℓ_2 norm among the elements (columns) of D. The goal is to estimate $\mathrm{Sum}(D)=\sum_i x_i$ with small ℓ_2 error. For each $x_i\in\mathbb{Z}^d$ and any $k\in[d]$, we use x_i^k to denote its k-th coordinate.

In the central model, the standard *Gaussian mechanism* achieves an error of $O\big(U_{\ell_2}\sqrt{d\log(1/\delta)}/\varepsilon\big)$, which is worst-case optimal up to logarithmic factors [35]. The best clipping mechanism for this problem [17] achieves an error of

$$O\Big(\mathrm{Max}_{\ell_2}(D)\cdot\big(\sqrt{d\log(1/\delta)}+\log\log(U_{\ell_2})\big)/\varepsilon\Big).$$

In the shuffle model, [27] presented a two-round protocol achieving a (theoretically) similar bound:

$$\tilde O\Big(\mathrm{Max}_{\ell_2}(D)\cdot\Big(\sqrt{d\log(nd)\log(1/\delta)}+\log^{3.5}U_{\ell_2}\cdot\sqrt{\log(1/\delta)}\Big)/\varepsilon\Big).$$

But similar to their 1D protocol, this algorithm is not practical due to the $\log^{3.5}U_{\ell_2}$ factor and the use of the complicated range-counting shuffle-DP protocol of [22].

In this section, we present a simple and practical one-round shuffle-DP protocol that achieves an error of

$$O\Big(\mathrm{Max}_{\ell_2}(D)\cdot\sqrt{d\log(nd)\log(1/\delta)}\cdot\log\log(U_{\ell_2})/\varepsilon\Big).$$

6.1 Random Rotation

As in [27], we first perform a random rotation of the dataset D, resulting in $\overline{D}=WD$. Here, W denotes a rotation matrix, constructed as per the following lemma:

Lemma 6.1 ([3]). *Let $W=HP$, where H is the Hadamard matrix and P is a diagonal matrix whose diagonal entry is independently and randomly sampled from $\{-1,+1\}$. Then, for any $x\in\mathbb{Z}^d_{\ge 0}$, and any $\beta>0$, we have*

(1) $Wx\in\mathbb{Z}^d$ and $\|Wx\|_2=\sqrt{d}\|x\|_2$

(2)

$$\Pr\Big[\big\|Wx\big\|_\infty\ge\|x\|_2\cdot\sqrt{2\log(4d/\beta)}\Big]\le\beta.$$

The first property means, matrix W performs a rotation while preserving the integer domain, aside from scaling the vector by a factor of $\sqrt{d}$. The second property ensures that the random rotation spreads out the norm evenly across all dimensions. After the rotation, [27] clips each coordinate to $O(U_{\ell_2}\log(nd))$ and invokes BaseSumDP with bounded domain size $O(U_{\ell_2}\log(nd))$ in each dimension. To guarantee DP, they use advanced composition to allocate each dimension with the privacy budget $\varepsilon'=\varepsilon/\big(2\sqrt{d\log(2/\delta)}\big)$, $\delta'=\delta/(2d)$. This approach results in an error of $\tilde O\Big(U_{\ell_2}d\sqrt{\log n\log(1/\delta)}/\varepsilon\Big)$ for the estimation of $\mathrm{Sum}\big(\overline{D}\big)$. Upon reorienting to the original domain with W^{-1}, the estimation of $\mathrm{Sum}(D)$ has error $\tilde O\Big(U_{\ell_2}\sqrt{d\log n\log(1/\delta)}/\varepsilon\Big)$. Note that the randomness in W is only needed for the utility analysis, and does not affect privacy, so it can be derived from public randomness.

6.2 Extending SumDP to High Dimensions

To adapt SumDP to high dimensions, one naïve approach is to use the advanced composition to divide the privacy budget and apply SumDP to each dimension. This will lead to an error of

$$\tilde O\left(\sqrt{d\log(1/\delta)\cdot\log\log U_{\ell_2}\cdot\sum_{k=1}^{d}\mathrm{Max}(D^{(k)})}\right),\tag{8}$$

where $\mathrm{Max}(D^{(k)})$ is the maximum value in the k dimension of D. Since $\sqrt{\sum_{k=1}^d\mathrm{Max}(D^{(k)})}$ can be as large as $\sqrt{d}\cdot\mathrm{Max}_{\ell_2}(D)$, (8) has a $\sqrt{d}$ degradation compared with the optimal error.

To achieve our error with a dependency on $\sqrt{d}$, we apply a rotation as in Lemma 6.1, and then clip each coordinate to the range $[-c\cdot U_{\ell_2}\log(nd),c\cdot U_{\ell_2}\log(nd)]$ for some constant c. One would then apply SumDP in each dimension. However, after the rotation, the resulting domain of $\overline{D}=HD$ spans both positive and negative integers. Note that SumDP only works on the non-negative integer domain, because its utility guarantee is based on the property that, clipping elements should only make the sum smaller, which is not true if negative numbers are present.

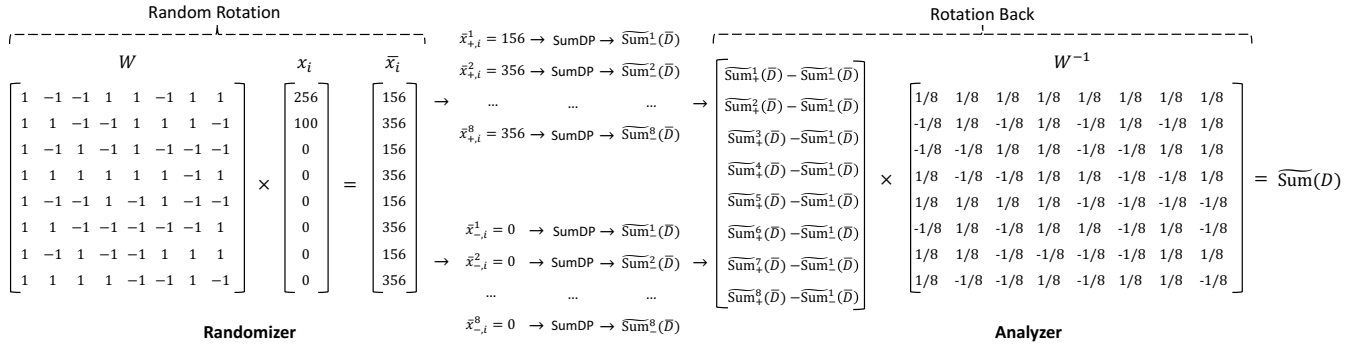

Figure 2: An illustration of our protocol for high-dimensional sum estimation. $d = 8$.

Algorithm 5: Randomizer of HighDimSumDP.

Input: $x_i \in \mathbb{Z}^d_{\geq 0}$, ε, δ, β, n, U_{ℓ_2}, W

$\bar{x}_i \leftarrow W x_i$;

$\bar{x}_{+,i} \leftarrow \left(\bar{x}^1_{+,i}, \bar{x}^2_{+,i}, \ldots, \bar{x}^d_{+,i}\right)$ with $\bar{x}^k_{+,i} \leftarrow \min\left(\bar{x}^1_i \cdot \mathbf{I}(\bar{x}^1_i > 0), U_{\ell_2}\sqrt{2\log(8nd/\beta)}\right)$ for any $k \in [d]$;

$\bar{x}_{-,i} \leftarrow \left(\bar{x}^1_{-,i}, \bar{x}^2_{-,i}, \ldots, \bar{x}^d_{-,i}\right)$ with $\bar{x}^k_{-,i} \leftarrow \min\left(-\bar{x}^1_i \cdot \mathbf{I}(\bar{x}^1_i < 0), U_{\ell_2}\sqrt{2\log(8nd/\beta)}\right)$ for any $k \in [d]$;

$\varepsilon', \delta' \leftarrow \varepsilon/\left(4\sqrt{d\log(2/\delta)}\right), \delta/(4d)$;

for $k \leftarrow 1, 2, \ldots, d$ **do**

  $S^k_{+,i} \leftarrow \text{Randomizer}\left(\bar{x}^k_{+,i}, \varepsilon', \delta', n, \beta/(2d), U_{\ell_2}\sqrt{2\log(8nd/\beta)}\right)$ of SumDP;

  $S^k_{-,i} \leftarrow \text{Randomizer}\left(\bar{x}^k_{-,i}, \varepsilon', \delta', n, \beta/(2d), U_{\ell_2}\sqrt{2\log(8nd/\beta)}\right)$ of SumDP;

end

Send $\{S^k_{-,i}\}_{k\in[d]}, \{S^k_{+,i}\}_{k\in[d]}$;

Algorithm 6: Analyzer of HighDimSumDP.

Input: $\{R^k_+ = \cup_i S^k_{+,i}\}_{k\in[d]}$ and $\{R^k_- = \cup_i S^k_{-,i}\}_{k\in[d]}$ with $\{S^k_{+,i}\}_{k\in[d]}$ and $\{S^k_{-,i}\}_{k\in[d]}$ from user i, ε, δ, β, n, U_{ℓ_2}, W

$\varepsilon', \delta' \leftarrow \varepsilon/\left(4\sqrt{d\log(2/\delta)}\right), \delta/(4d)$;

for $k \leftarrow 1, 2, \ldots, d$ **do**

  $\widetilde{\text{Sum}}^k_+(\overline{D}) \leftarrow \text{Analyzer}\left(R^k_+, \varepsilon', \delta', \beta/(2d), U_{\ell_2}\sqrt{2\log(8nd/\beta)}\right)$ of SumDP;

  $\widetilde{\text{Sum}}^k_-(\overline{D}) \leftarrow \text{Analyzer}\left(R^k_-, \varepsilon', \delta', \beta/(2d), U_{\ell_2}\sqrt{2\log(8nd/\beta)}\right)$ of SumDP;

end

$\widetilde{\text{Sum}}(\overline{D}) \leftarrow \left(\widetilde{\text{Sum}}^1_+(\overline{D}) - \widetilde{\text{Sum}}^1_-(\overline{D}), \widetilde{\text{Sum}}^2_+(\overline{D}) - \widetilde{\text{Sum}}^2_-(\overline{D}), \ldots, \widetilde{\text{Sum}}^d_+(\overline{D}) - \widetilde{\text{Sum}}^d_-(\overline{D})\right)$;

$\widetilde{\text{Sum}}(D) \leftarrow W^{-1}\widetilde{\text{Sum}}(\overline{D})$;

return $\widetilde{\text{Sum}}(D)$

A simplistic strategy is to shift the domain from $[-c\cdot U_{\ell_2}\log(nd), c\cdot U_{\ell_2}\log(nd)]$ to $[0, 2c\cdot U_{\ell_2}\log(nd)]$, and then apply SumDP. However, this shift escalates the maximum value for each coordinate to $\geq c\cdot U_{\ell_2}\log(nd)$, potentially inducing an error proportional to U_{ℓ_2} in the estimation of $\text{Sum}(\overline{D})$. To fix this issue, we process the positive and negative domains separately, and then take their difference as the final estimate for $\text{Sum}(\overline{D})$. We call this algorithm HighDimSumDP. The detailed algorithms for the randomizer and analyzer are shown in Algorithm 5 and Algorithm 6. We also show an example in Figure 2.

Theorem 6.2. *Given any $\varepsilon > 0$, $\delta > 0$, $n \in \mathbb{Z}_+$, and $U \in \mathbb{Z}_+$, for any $D \in \mathbb{Z}^{n\times d}$ with $\text{Max}_{\ell_2}(D) \leq U$, we have*

(1) The messages received by the analyzer preserves (ε, δ)-DP;
(2) With probability at least $1-\beta$, the ℓ_2 error is bounded by

$$O\Big(\text{Max}_{\ell_2}(D)\cdot\sqrt{d\log(nd/\beta)\log(1/\delta)} \cdot\log(d\log(U_{\ell_2})/\beta)/\varepsilon\Big);$$

(3) In expectation, each user sends

$$d + O\big(d^{1.5} \cdot \log(U_{\ell_2} \log(nd/\beta)) \cdot \log^{1.5}(d/\delta) \cdot \log^2 n/(\varepsilon\sqrt{n})\big)$$

messages with each message containing $O\Big(\log d + \min\big(\log n, \log(U_{\ell_2})\big)$ *bits.*

Proof. For privacy, invoking Theorem 5.1, we deduce that each of R_+^k and R_-^k adheres to $\Big(\frac{\varepsilon}{4\sqrt{d\log(2/\delta)}}, \frac{\delta}{4d}\Big)$-DP. By advanced composition, the collections of $\{R_+^k\}k \in [d]$ and $\{R-^k\}_{k\in[d]}$ maintain (ε, δ)-DP.

Regarding utility, Lemma 6.1 guarantees, with probability at least $1 - \frac{2}{\beta}$, for every $i \in [n]$ and $j \in [d]$, we have

$$|\bar{x}_i^k| \le \mathrm{Max}_{\ell_2}(D) \cdot \sqrt{2\log(8nd/\beta)}, \tag{9}$$

which further implies

$$\sum_i \bar{x}_i = \sum_i \bar{x}_{+,i} - \sum_i \bar{x}_{-,i}. \tag{10}$$

Subsequently, Theorem 5.1 coupled with Equation 9 implies that for each $k \in [d]$, with probability at least $1 - \frac{\beta}{2d}$,

$$\Big|\widetilde{\mathrm{Sum}}_+^k(\overline{D}) - \sum_i \bar{x}_{+,i}^k\Big| = O\Big(\mathrm{Max}_{\ell_2}(D) \cdot \sqrt{d\log(1/\delta)} \cdot \sqrt{\log(nd/\beta)} \cdot \log(d\log(U_{\ell_2})/\beta)/\varepsilon\Big) \tag{11}$$

and

$$\Big|\widetilde{\mathrm{Sum}}_-^k(\overline{D}) - \sum_i \bar{x}_{-,i}^k\Big| = O\Big(\mathrm{Max}_{\ell_2}(D) \cdot \sqrt{d\log(1/\delta)} \cdot \sqrt{\log(nd/\beta)} \cdot \log(d\log(U_{\ell_2})/\beta)/\varepsilon\Big) \tag{12}$$

Combining the probabilities across all $k \in [d]$, we have with probability at least $1 - \frac{\beta}{2}$, (11) and (12) hold for all dimensions.

By synthesizing (10), (11), and (12), we

$$\begin{aligned}&\Big\|\widetilde{\mathrm{Sum}}(\overline{D}) - \mathrm{Sum}(\overline{D})\Big\|_2 \\ =&\frac{1}{\sqrt{d}} \cdot \Big\|\widetilde{\mathrm{Sum}}(\overline{D}) - \mathrm{Sum}(\overline{D})\Big\|_2 \\ =&O\Big(\mathrm{Max}_{\ell_2}(D) \cdot \sqrt{d\log(nd/\beta)\log(1/\delta)} \cdot \log(d\log(U_{\ell_2})/\beta)/\varepsilon\Big)\end{aligned}$$

Finally, our assertion on communication cost derives from (9), Theorem 5.1, along with the observation that for any $i \in [n]$ and $k \in [d]$, either $\bar{x}_{+,i}^k$ or $\bar{x}_{-,i}^k$ is necessarily zero. □

7 Sparse Vector Aggregation

As the last application of our technique, we study the sparse vector aggregation problem. In this problem, each x_i is a binary vector in $\{0,1\}^d$. We use $(\mathrm{Max}_{\ell_2}(D))^2 = \max_i \|x_i\|_1$ to quantify the data's sparsity and are interested in the sparse case where $(\mathrm{Max}_{\ell_2}(D))^2 \ll d$. We want to estimate $\mathrm{Sum}(D)$ with an ℓ_∞ error $\mathrm{Max}_{\ell_2}(D)/\varepsilon \cdot \mathrm{poly}\log(d/\delta)$. Meanwhile, we would like the message complexity of each user i to depend on $\|x_i\|_1$, i.e., the number of 1's in x_i. Note that the ℓ_2 error in Theorem 6.2 can only imply the same ℓ_∞ error, namely, it is $\sqrt{d}$ times larger than desired. Moreover, it requires d messages per user.

7.1 Clipping on Sparsity

If an upper bound of sparsity $S \ge \big(\mathrm{Max}_{\ell_2}(D)\big)^2$ is given, we can estimate the count for each coordinate independently with the privacy budget $\varepsilon' = \varepsilon/\big(\sqrt{S\log(2/\delta)}\big)$, $\delta' = \delta/(2S)$. Given that each x_i at most affects the counting for S dimensions, with advanced composition, this whole process preserves (ε, δ)-DP. The state-of-the-art protocol for counting under the shuffle-DP model is BaseSumDP without random rounding, where the communication is improved to $1 + (\log(1/\delta)/(\varepsilon n))$ messages per user. Feeding this into the above protocol for sparse vector aggregation yields an error proportional to $\sqrt{S}$ and $\|x_i\|_1 + O(d^{1.5}\log^{1.5}/(\varepsilon n))$ messages per user.

In the absence of a good upper bound S, one could apply the clipping mechanism on sparsity. Specifically, for some τ, we only retain the first τ non-zero coordinates of each x_i and set the rest to 0. Then we apply the mechanism above with $S = \tau$. However, as in the sum estimation problem, the key is to choose a good τ that balances the DP noise and bias, and the optimal τ should achieve an error proportional to $\mathrm{Max}_{\ell_2}(D)$. More importantly, we would like to choose τ and compute the noisy counts of all dimensions clipped by τ simultaneously in one round.

7.2 Sparsity Partitioning

We use the idea of domain partitioning from our sum estimation protocol. But for the sparse vector aggregation problem, we partition the domain of possible sparsity levels $[d]$ into $\log d + 1$ disjoint sub-domains: $[1,1]$, $[2,2]$, $[3,4]$, …, $[d/2+1, d]$. Then, we divide the vectors according to their sparsity. More precisely, for each $j \in \{0, 1, 2, \ldots, \log d\}$, let

$$D[2^{j-1}+1, 2^j] = \big\{x_i \in D : \|x_i\|_1 \in [2^{j-1}+1, 2^j]\big\}.$$

Since each vector in $D[2^{j-1}+1, 2^j]$ has the sparsity bounded by 2^j, we can use the idea discussed in the last section. For the error, the estimation of $\mathrm{Sum}\big(D[2^{j-1}+1, 2^j]\big)$ has an ℓ_∞ error bounded by $\tilde{O}(\sqrt{2^j})$. In terms of the communication, since each x_i will only be involved in $D[2^{j-1}+1, 2^j]$, each user sends $\|x_i\|_1 + \tilde{O}(d^{1.5}\log^{1.5}(1/\delta)/(\varepsilon n))$ messages in expectation.

Next, let us discuss how to use the estimations of $\mathrm{Sum}(D[2^{j-1}+1, 2^j])$ to reconstruct $\mathrm{Sum}(D)$. Recall that, in sum estimation, we have the estimations for each value domain, i.e., $\mathrm{Sum}(D \cap [2^{j-1}+1, 2^j])$ find the last $[2^{j-1}+1, 2^j]$ with a large noisy sum result. This is to guarantee enough elements are located in the domain $[2^{j-1}+1, 2^j]$. Unfortunately, such an idea cannot be extended to the high-dimensional case directly. The problem is that, even though, there are a large number of vectors with the sparsity in the range of $[2^{j-1}+1, 2^j]$, i.e., $\big|D[2^{j-1}+1, 2^j]\big|$ is large enough, each coordinate of $\mathrm{Sum}\big(D[2^{j-1}+1, 2^j]\big)$ can still be very small since those vectors can contribute totally different coordinates.

The solution here is that we build an extra counter for the number of vectors with sparsity within each $[2^{j-1}+1, 2^j]$ as a judgment for whether to include $\mathrm{Sum}\big(D[2^{j-1}+1, 2^j]\big)$ in the final result. More precisely, each user first executes $\log d + 1$ number of instances

Algorithm 7: Randomizer of SparVecSumDP.

Input: $x_i \in \{0,1\}^d$, ε, δ, β, n

for $j \leftarrow 0, 1, 2, \ldots, \log d$ **do**

  /* The messages for counting vectors with sparsity between $2^{j-1}+1$ and 2^j */

  $S_{\mathrm{cnt},i}^{[2^{j-1}+1,2^j]} \leftarrow \mathrm{Randomizer}\Big(\mathbf{I}\big(|x_i|_1 \in [2^{j-1}+1, 2^j]\big), \varepsilon/2, \delta/2, n, 1\Big)$ of BaseSumDP;

  /* The messages for sum for vectors with sparsity between $2^{j-1}+1$ and 2^j */

  $\varepsilon', \delta' \leftarrow \varepsilon/\big(2\sqrt{2^{j+1}\log(2/\delta)}\big), \delta/(2^{j+1})$;

  for $k \leftarrow 1, 2, \ldots, d$ **do**

    $S_{\mathrm{sum},i}^{[2^{j-1}+1,2^j],k} \leftarrow \mathrm{Randomizer}\big(x_i^k \cdot \mathbf{I}(|x_i|_1 \in [2^{j-1}+1, 2^j]), \varepsilon', \delta', n, 1\big)$ of BaseSumDP;

  end

end

Send $\big\{S_{\mathrm{sum},i}^{[2^{j-1}+1,2^j],k}\big\}_{j\in\{0,1,2,\ldots,\log d\},k\in[d]}$ and $\big\{S_{\mathrm{cnt},i}^{[2^{j-1}+1,2^j]}\big\}_{j\in\{0,1,2,\ldots,\log d\}}$;

Algorithm 8: Analyzer of SparVecSumDP.

Input: $\big\{R_{\mathrm{sum}}^{[2^{j-1}+1,2^j],k}\big\}_{j\in\{0,1,2,\ldots,\log d\},k\in[d]} = \cup_i S_{\mathrm{sum},i}^{[2^{j-1}+1,2^j],k}\big\}_{j\in\{0,1,2,\ldots,\log d\},k\in[d]}$ and $\big\{R_{\mathrm{cnt}}^{[2^{j-1}+1,2^j]} = \cup_i S_{\mathrm{cnt},i}^{[2^{j-1}+1,2^j]}\big\}_{j\in\{0,1,2,\ldots,\log d\}}$, ε, δ, β, n

$\tau \leftarrow 0$;

for $j \leftarrow 0, 1, 2, \ldots, \log d$ **do**

  $\widetilde{\mathrm{Count}}\big(D[2^{j-1}+1, 2^j]\big) \leftarrow \mathrm{Analyzer}\Big(R_{\mathrm{cnt}}^{[2^{j-1}+1,2^j]}, \varepsilon/2, \delta/2, n, 1\Big)$ of BaseSumDP;

  $\varepsilon', \delta' \leftarrow \varepsilon/\big(2\sqrt{2^{j+1}\log(2/\delta)}\big), \delta/(2^{j+1})$;

  for $k \leftarrow 1, 2, \ldots, d$ **do**

    $\widetilde{\mathrm{Sum}}^k\big(D[2^{j-1}+1, 2^j]\big) \leftarrow \mathrm{Analyzer}\Big(R_{\mathrm{sum}}^{[2^{j-1}+1,2^j],k}, \varepsilon', \delta', n, 1\Big)$ of BaseSumDP;

  end

  $\widetilde{\mathrm{Sum}}\big(D[2^{j-1}+1, 2^j]\big) \leftarrow \Big(\widetilde{\mathrm{Sum}}^1\big(D[2^{j-1}+1, 2^j]\big), \widetilde{\mathrm{Sum}}^2\big(D[2^{j-1}+1, 2^j]\big), \ldots, \widetilde{\mathrm{Sum}}^d\big(D[2^{j-1}+1, 2^j]\big)\Big)$;

  if $\widetilde{\mathrm{Count}}\big(D[2^{j-1}+1, 2^j]\big) > 1.3 \cdot \frac{2}{\varepsilon} \cdot \log(2(\log d + 1)/\beta)$ **then**

    $\tau \leftarrow 2^j$;

  end

end

$\widetilde{\mathrm{Sum}}(D) \leftarrow \sum_{j\in\{0,1,2,\ldots,\log\tau\}} \widetilde{\mathrm{Sum}}\big(D[2^{j-1}+1, 2^j]\big)$;

of BaseSumDP, each of which is to estimate the number of vectors with sparsity within $[2^{j-1}+1, 2^j]$ and uses privacy budget $\varepsilon/2$ and $\delta/2$. Then, for each $j \in \{0, 1, 2, \ldots, \log d\}$, we estimate $\mathrm{Sum}\big(D[2^{j-1}+1, 2^j]\big)$, where we use one CounDP with the privacy budget $\varepsilon/\big(2\sqrt{2^{j+1}\log(2/\delta)}\big)$ and $\delta/(2^{j+1})$ to do the sum estimation in kth coordinate. In the view of the analyzer, with the received messages, we can easily get the estimation for $\mathrm{Count}\big(D[2^{j-1}+1, 2^j]\big)$ and $\mathrm{Sum}\big(D[2^{j-1}+1, 2^j]\big)$ for each $j \in \{0, 1, 2, \ldots, \log d\}$. We set $\tau = 2^j$ with the last j such that $\mathrm{Count}\big(D[2^{j-1}+1, 2^j]\big)$ is large enough. Finally, we sum all estimations for $\mathrm{Sum}\big(D[2^{j-1}+1, 2^j]\big)$ for $j \le \log(\tau)$. The detailed algorithms for the randomizer and analyzer are shown in Algorithms 7 and 8.

Theorem 7.1. *Given any $\varepsilon > 0$, $\delta > 0$, $n \in \mathbb{Z}_+$, and for any $D \in \{0,1\}^{n\times d}$, the* SparsVecSumDP *achieves the following:*

(1) The messages received by the analyzer preserves (ε, δ)-DP;

(2) With probability at least $1-\beta$, for every $k \in [d]$, the ℓ_∞ error is bounded by

$$O\Big(\big(\mathrm{Max}_{\ell_2}(D) \cdot \sqrt{\log(1/\delta)} + \log\log d\big) \cdot \log(d/\beta)/\varepsilon\Big);$$

(3) In expectation, each user sends $\|x_i\|_1 + 1 + O\big(d^{1.5} \cdot \log d \cdot \log^{1.5}(1/\delta)/(\varepsilon n)\big)$ messages with each containing $O(\log d)$ bits.

Proof. For privacy, invoking Lemma 3.5 ensures that for each $j \in \{0, 1, 2, \ldots, \log(d)\}$, $R_{\mathrm{cnt}}^{[2^{j-1}+2^j]}$ preserves $(\varepsilon/2, \delta/2)$-DP. Additionally, for each $j \in \{0, 1, 2, \ldots, \log(d)\}$, by combining Lemma 3.5 with advanced composition and the fact that each $x_i \in D[2^{j-1}+1, 2^j]$ affects at most 2^j number of $R_{\mathrm{sum}}^{[2^{j-1}+2^j],k}$, we have that, $\big\{R_{\mathrm{sum}}^{[2^{j-1}+2^j],k}\big\}_{k\in[d]}$ preserves $(\varepsilon/2, \delta/2)$-DP. Given that each x_i impacts exactly one $R_{\mathrm{cnt}}^{[2^{j-1}+2^j]}$ and one $\big\{R_{\mathrm{sum}}^{[2^{j-1}+2^j],k}\big\}_{k\in[d]}$, the overall privacy guarantee is achieved.

Concerning utility, Theorem 3.5 implies that for each $j \in \{0, 1, 2, \ldots, \log d\}$, with probability at least $1 - \frac{\beta}{2(\log d+1)}$, we have

$$\left|\mathrm{Count}([2^{j-1}+1, 2^j]) - \widetilde{\mathrm{Count}}([2^{j-1}+1, 2^j])\right| \leq \frac{2}{\varepsilon} \cdot \log\left(2(\log d + 1)/\beta\right), \tag{13}$$

and the difference in sums, also with probability at least $1-\frac{\beta}{2(\log d+1)}$, is well bounded:

$$\left|\mathrm{Sum}([2^{j-1}+1, 2^j]) - \widetilde{\mathrm{Sum}}([2^{j-1}+1, 2^j])\right|_\infty = O\left(\sqrt{2^j \log(1/\delta)} \cdot \log(d/\beta)/\varepsilon\right). \tag{14}$$

Aggregating these probabilities, we ensure both (13) and (14) hold for all j with probability at least $1 - \beta$.

(13) implies that

$$j \leq \left\lceil \log\left(\mathrm{Max}_{\ell_2}(D)\right)\right\rceil. \tag{15}$$

and

$$\sum_{j=\log\tau+1}^{\log(d)} \mathrm{Count}\big(D[2^{j-1}+1, 2^j]\big) = O\Big(\log d \log(\log d/\beta)/\varepsilon\Big),$$

which sequentially deduces

$$\Big|\sum_{j=\log\tau+1}^{\log(d)} \mathrm{Sum}([2^{j-1}+1, 2^j])\Big|_\infty = O\Big(\log d \log(\log d/\beta)/\varepsilon\Big). \tag{16}$$

Combining (14) and (15), we have

$$\Big|\sum_{j=0}^{\log\tau} \left(\mathrm{Sum}([2^{j-1}+1, 2^j]) - \widetilde{\mathrm{Sum}}([2^{j-1}+1, 2^j])\right)\Big|_\infty = O\Big(\mathrm{Max}_{\ell_2}(D) \cdot \sqrt{\log(1/\delta)} \cdot \log(d/\beta)/\varepsilon\Big). \tag{17}$$

Finally, combining (16) and (17) leads to our statement for utility.

For communication, recall that each Baseline without random rounding yields $1 + O(\log(1/\delta)/(\varepsilon n))$ messages per user in expectation. Combing this with facts that each x_i has $|x_i|_1$ number of non-zero coordinates, and there is only one $[2^{j-1}+1, 2^j]$ such that $|x_i| \in [2^{j-1}+1, 2^j]$, we derive the desired statement. One special note is that each message requires $O(\log d)$ bits to specify the dimension. □

8 Practical Optimizations

In this section, we briefly discuss some practical optimizations for our protocols, although they do not affect the asymptotic results.

As mentioned, for sum estimation protocol of [7] attains an error very closely to that of [24]. Meanwhile, [7] send $O(1)$ messages per user while [24] sends $1 + o(1)$ messages. Although the former is asymptotically smaller, the $o(1)$ term, or $O\big(\log^2(n) \cdot \log(1/\delta)/(\varepsilon\sqrt{n})\big)$ to be more precise, is actually not negligible for n not too large. Since our mechanism uses sum estimation as a black box, in our implementation we choose either [24] or [7] based on the concrete values of n, ε, δ.

Furthermore, recall that in SumDP, we invoke $\log(U) + 1$ instances of BaselineSumDP, corresponding to different domain sizes $1, 2, 4, \ldots, U$. We note that using the protocol of [24] without random rounding yields a message number of $1+O(U \log^2(U) \log(U/\delta)/(n\varepsilon))$, which may be better than doing a random rounding when the domain size is small. Therefore, for different domain sizes, we adopt different baselines: [24] with or without random rounding or [7]. We again choose the best one based on the concrete values of n, ε, δ, and domain size.

9 Experiments

In addition to the improved asymptotic results, we have also conducted experiments comparing our protocols with the previous algorithms.

Sum estimation: Our SumDP mechanism was evaluated alongside two baselines: GKMPS [24] and BBGN [7]. The two-round protocol from [27] is solely a theoretical result. It not only has a large message number and errors but also has an impractical running time (detailed in Appendix A). We also compared its error to the state-of-the-art central-DP mechanism [18] as a gold standard.

High dimensional sum: For high-dimensional sum estimation, our HighDimSumDP was compared against the one-round protocol HLY proposed in [27]. Similar to sum estimation, the two-round protocol from [27] faced efficiency challenges, as outlined in Appendix A. For this problem, we use the central-DP mechanism in [17] as the gold standard.

Sparse vector aggregation: For sparse vector aggregation, we assessed SparVecSumDP against NaiveVecSumDP, which uses the dimension d as the upper bound for sparsity. Here, we also use the central-DP mechanism in [17] as the gold standard.

Dataset	n	U	$\mathrm{Max}(D)$
SF-Sal	1.49×10^5	2.5×10^5	568
Ont-Sal	5.75×10^5	2.5×10^5	1750
BR-Sal	1.09×10^6	2.5×10^5	343
JP-Trad	1.81×10^5	2×10^5	2810

Table 2: Real-world datasets used in sum aggregation

9.1 Setup

Datasets. We used both synthetic and real-world datasets in the experiments. For sum estimation, the synthetic data was generated from two families of distributions over $[U]$ with $n = U = 10^5$: Zipf distribution $f(x) \propto (x + a)^{-b}$ with $a = 1$, $b = 3$ and $a = 1$, $b = 5$; Gauss distribution with $\mu = 5$, $\sigma = 5$ and $\mu = 50$, $\sigma = 50$. Here, we utilize the Gaussian distribution for its symmetry and the Zipf distribution for its asymmetry and both distributions allow for easy generation of datasets with varying skewness through parameter adjustments. The real-world datasets were collected from Kaggle, including San-Francisco-Salary (SF-Sal) [28], Ontario-Salary (Ont-Sal) [32], Brazil-Salary (BR-Sal) [31], and Japan-trade

Dataset			Simulated Data				Real-world Data			
			Zipf		Gauss					
			$a=1$ $b=3$	$a=1$ $b=5$	$\mu=5$ $\sigma=5$	$\mu=50$ $\sigma=50$	SF-Sa	Ont-Sa	BR-Sa	JP-Trad
1-D Sum	SOTA under central-DP RE(%)		0.53	0.0247	0.00921	0.00737	0.00936	0.0028	0.0452	0.168
	SumDP (Ours)	RE(%)	**1.13**	**0.0661**	**0.00351**	**0.00452**	**0.00989**	**0.0075**	**0.0249**	**0.101**
		#Messages/user	140	140	139	139	143	126	119	140
	GKMPS	RE(%)	54.5	96.3	22.5	3.11	2.44	0.372	9.26	9.02
		#Messages/user	14200	14300	14300	14300	12200	7770	5950	11200
	BBGN	RE(%)	53	76.6	17.6	1.43	1.96	0.294	3.53	4.46
		#Messages/user	**9**	**9**	**9**	**9**	**9**	**8**	**8**	**9**

Table 3: Comparison among sum estimation mechanisms under shuffle-DP ($\varepsilon = 1$). RE denotes the relative error.

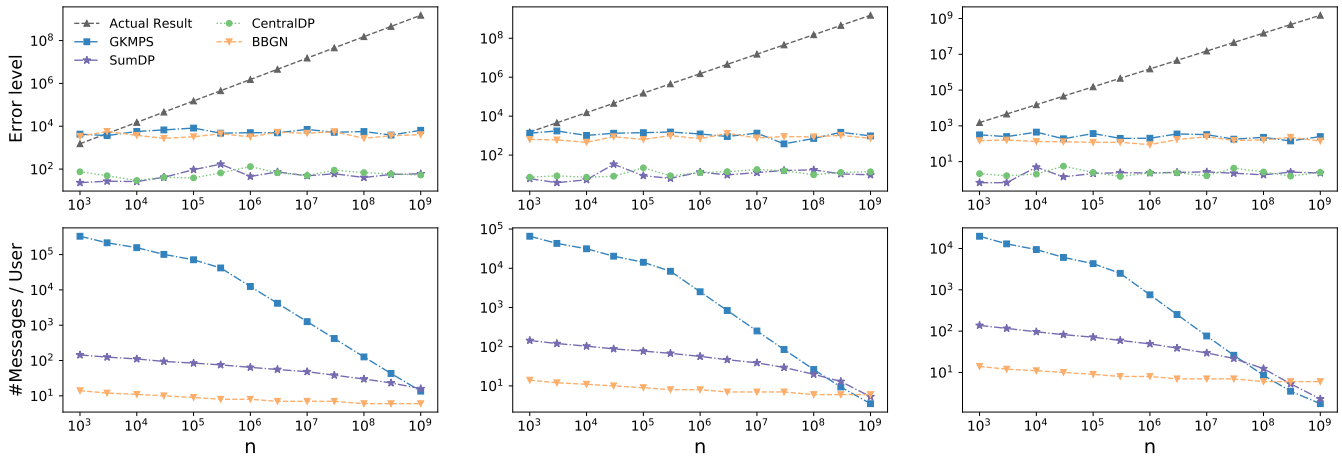

Figure 3: Error levels and average messages per user for the sum estimation mechanisms under shuffle-DP with different data size n. CentralDP **represents the state-of-the-art algorithm for sum estimation under central-DP.**

(JP-Trad) [29]. Here, we use them to perform summing salaries and trading amounts, which are two common data analytical tasks in real life. Given the significant variance in salaries among different groups, achieving instance-specific error in these tasks is crucial. SF-Sal, BR-Sal, and Ont-Sal are salary data from San Francisco, Brazil, and Ontario for the years 2014, 2020, and 2020, respectively, with amounts presented in thousands of US dollars (K USD). For the salary data, we set the domain limit U to 2.5×10^5, which is the world's highest recorded salary [40]. The JP-Trad dataset, capturing Japan's trade statistics from 1988 to 2019, includes 100 million entries. We selected a subset of approximately 200,000 tuples, covering Japan's trade activities with a designated country. We set U as the maximum value across the entire dataset. This dataset also has the amounts expressed in K USD. The details of these real-world data can be found in Table 2.

For high-dimensional sum estimation, we utilized the MNIST dataset [30], comprising 70,000 digit images, with each represented by a vector of dimension $d = 28 \times 28 = 784$. The U_{ℓ_2} parameter was set to 2^{20}.

The sparse vector aggregation experiments were conducted using the AOL-user-ct-collection (AOL) [38], documenting 500,000 users' clicks on 1,600,000 URLs. we consolidated every 100 webpages into a single dimension, resulting in a dimensionality of 1.6×10^4, and selected the first 50,000 users as our testing dataset.

Experimental parameters. All experiments are conducted on a Linux server equipped with a 24-core 48-thread 2.2GHz Intel Xeon CPU and 256GB memory. We used absolute error, ℓ_2 error, and ℓ_∞ error metrics for sum estimation, high-dimensional sum, and sparse vector aggregation respectively. We repeated each experiment 50 times, discarding the 10 largest and smallest errors for an averaged result from the remaining 30. The message complexity was quantified by the average number of messages per user, with each message containing $O(\log(d) + \log(U) + \log(n))$ bits. For the privacy budget, we used $\varepsilon = 0.2, 1, 5$, and the default value was set to 1. To protect data privacy, δ should be set to a value significantly

smaller than the inverse of the data size. Therefore, δ was fixed at 10^{-12} in our experiments. [8] The failure probability β was set at 0.1.

9.2 Experimental Results

In this section, we discuss our experimental results for sum aggregation, where we include the experiments to investigate the influence of different data size. The experiments to assess the impacts of domain size and data skewness and the results for high-dimensional sum and sparse vector aggregation are deferred to our full-version paper [16].

Utility and communication. Table 3 shows the errors and the average number of messages per user across various mechanisms for sum estimation under shuffle-DP over both simulated and real-world data. The results indicate a clear superiority of SumDP in terms of utility. SumDP consistently maintains an error below 2% across all eight tests, further reducing it to below 0.2% in seven cases. In contrast, GKMPS and BBGN exhibit significantly higher error levels. Our improvement over GKMPS and BBGN can be up to 3000×. This superiority is particularly evident in the JP-Trad dataset, where SumDP surpasses GKMPS and BBGN by more than 40× even with a pre-established U based on strong prior knowledge. This validates our theoretical analysis: SumDP achieves an instance-specific error, unlike GKMPS and BBGN, which target worst-case errors. Furthermore, we observe that SumDP attains error levels similar to the gold standard, and produces even smaller errors in about half of the cases. This is because while the two methods have the same asymptotic error bounds, they are both upper bounds that may not be tight (in constant factors) on all instances. Therefore, the actual error of either mechanism could be smaller than the other.

In terms of communication, neither SumDP nor GKMPS achieves the theoretical ideal of single-message communication per user in all tests. In contrast, BBGN requires fewer messages. This is because even though SumDP and GKMPS theoretically reach $1 + o(1)$ messages per user, the term $o(1)$ masks substantial logarithmic factors, leading to significantly higher actual message counts, especially when n is small. In contrast, BBGN maintains constant messages per user. Later, we will show that as n increases, both GKMPS and SumDP exhibit a trend towards achieving a single-message communication per user. Additionally, SumDP requires much fewer messages than GKMPS. This is attributed to the optimization described in Section 8, where our mechanism intelligently chooses the more communication-efficient method between GKMPS and BBGN.

Data size. To assess the impact of varying data sizes, we conducted experiments using simulated data generated from a Gaussian distribution with $\mu = 1$, $\sigma = 1$, and domain size $U = 10^3$. The data size varied from 10^3 to 10^9, and we tested with different privacy budgets $\varepsilon = 0.2, 1, 5$. The error levels and average messages per user are depicted in Figure 3. Note that in all our figures, both axes are in log-scale and the actual query results are plotted alongside the error levels to provide a benchmark for assessing the utility of the mechanisms. In terms of utility, SumDP consistently has a high utility even with small n and ε, akin to the state-of-the-art central-DP mechanism. Notably, the error levels for all mechanisms did not exhibit significant changes with varying n, matching our analytical analyses that the errors in BBGN and GKMPS are dependent on U, while the errors in SumDP and the central-DP mechanism depend on $\mathrm{Max}(D)$, all of which are not directly influenced by the data size.

Regarding communication, GKMPS showed a decrease in the average messages per user with larger n values, while BBGN maintained a constant message complexity. SumDP displayed a unique trend: it maintained its message complexity for smaller n values, then gradually decreased it as n increased. This pattern is attributed to SumDP initially leveraging BBGN for smaller datasets and then transitioning to GKMPS for larger datasets. Moreover, as n increases, both GKMPS and SumDP demonstrate a progression towards single-message communication per user, aligning with our theoretical analysis that both mechanisms achieve $1+o(1)$ messages per user.

[8] Notably, for our mechanism, a larger δ will not affect error but will benefit the communication, albeit minimally as it affects only the logarithmic term.

10 Conclusion

In this paper, we study answering sum estimation under the shuffle-DP model, where prior works either only achieve worst-case optimal error or have very a heavy communication cost. We introduce the first protocol that not only has instance-optimal error but also achieves optimal communication efficiency, i.e., requiring only $1 + o(1)$ messages per user. Furthermore, we successfully extend our technique to address high-dimensional sum estimation and sparse vector aggregation. Finally, we would like to mention two interesting directions for future research. The first is how to extend our domain division technique to private sum estimation in various models, such as the multi-party secure computation model. Besides, since the private summation is the foundation to private protocols for various machine learning models, investigating its potential to enhance utility in these advanced tasks would also be valuable.

Acknowledgements

This work has been in part supported by a grant from ONR, a grant from the DARPA SIEVE program under a subcontract from SRI, a Packard Fellowship, NTU-NAP start up grant, and contributions from Intel, Bosch, and Cisco. Additionally, this work has been funded by NSF awards under grant numbers 2128519, 2044679, 2338772, and 2148359. Qiyao Luo and Ke Yi have been supported by HKRGC under grants 16205420, 16205422, and 16204223.

References

[1] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security* (2016), pp. 308–318.

[2] Agarwal, N., Suresh, A. T., Yu, F. X. X., Kumar, S., and McMahan, B. cpsgd: Communication-efficient and differentially-private distributed sgd. *Advances in Neural Information Processing Systems 31* (2018).

[3] Ailon, N., and Chazelle, B. The fast johnson–lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on computing 39*, 1 (2009), 302–322.

[4] Andrew, G., Thakkar, O., McMahan, B., and Ramaswamy, S. Differentially private learning with adaptive clipping. *Advances in Neural Information Processing Systems 34* (2021), 17455–17466.

[5] Asi, H., and Duchi, J. C. Instance-optimality in differential privacy via approximate inverse sensitivity mechanisms. *Advances in neural information processing systems 33* (2020).

[6] Balle, B., Bell, J., Gascón, A., and Nissim, K. The privacy blanket of the shuffle model. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International*

Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II 39 (2019), Springer, pp. 638–667.

[7] Balle, B., Bell, J., Gascón, A., and Nissim, K. Private summation in the multi-message shuffle model. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (2020), pp. 657–676.

[8] Bassily, R., Smith, A., and Thakurta, A. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th annual symposium on foundations of computer science* (2014), IEEE, pp. 464–473.

[9] Biswas, S., Dong, Y., Kamath, G., and Ullman, J. Coinpress: Practical private mean and covariance estimation. *Advances in Neural Information Processing Systems 33* (2020).

[10] Bittau, A., Erlingsson, Ú., Maniatis, P., Mironov, I., Raghunathan, A., Lie, D., Rudominer, M., Kode, U., Tinnes, J., and Seefeld, B. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th symposium on operating systems principles* (2017), pp. 441–459.

[11] Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017), pp. 1175–1191.

[12] Cheu, A., Smith, A., Ullman, J., Zeber, D., and Zhilyaev, M. Distributed differential privacy via shuffling. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I 38* (2019), Springer, pp. 375–403.

[13] Damgård, I., Nielsen, J. B., Ostrovsky, R., and Rosén, A. Unconditionally secure computation with reduced interaction. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35* (2016), Springer, pp. 420–447.

[14] Dick, T., Kulesza, A., Sun, Z., and Suresh, A. T. Subset-based instance optimality in private estimation. *arXiv preprint arXiv:2303.01262* (2023).

[15] Dong, W., Fang, J., Yi, K., Tao, Y., and Machanavajjhala, A. R2t: Instance-optimal truncation for differentially private query evaluation with foreign keys. In *Proceedings of the 2022 International Conference on Management of Data* (2022), pp. 759–772.

[16] Dong, W., Luo, Q., Fanti, G., Shi, E., and Yi, K. Almost instance-optimal clipping for summation problems in the shuffle model of differential privacy. *arXiv preprint arXiv:2403.10116* (2024).

[17] Dong, W., Sun, D., and Yi, K. Better than composition: How to answer multiple relational queries under differential privacy. *Proceedings of the ACM on Management of Data 1*, 2 (2023), 1–26.

[18] Dong, W., and Yi, K. Universal private estimators. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems* (2023), pp. 195–206.

[19] Dwork, C., and Roth, A. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science 9*, 3–4 (2014), 211–407.

[20] Erlingsson, Ú., Feldman, V., Mironov, I., Raghunathan, A., Talwar, K., and Thakurta, A. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms* (2019), SIAM, pp. 2468–2479.

[21] Fang, J., Dong, W., and Yi, K. Shifted inverse: A general mechanism for monotonic functions under user differential privacy. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (2022), pp. 1009–1022.

[22] Ghazi, B., Golowich, N., Kumar, R., Pagh, R., and Velingker, A. On the power of multiple anonymous messages: Frequency estimation and selection in the shuffle model of differential privacy. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (2021), Springer, pp. 463–488.

[23] Ghazi, B., Kumar, R., Manurangsi, P., and Pagh, R. Private counting from anonymous messages: Near-optimal accuracy with vanishing communication overhead. In *International Conference on Machine Learning* (2020), PMLR, pp. 3505–3514.

[24] Ghazi, B., Kumar, R., Manurangsi, P., Pagh, R., and Sinha, A. Differentially private aggregation in the shuffle model: Almost central accuracy in almost a single message. In *International Conference on Machine Learning* (2021), PMLR, pp. 3692–3701.

[25] Ghazi, B., Manurangsi, P., Pagh, R., and Velingker, A. Private aggregation from fewer anonymous messages. In *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part II 30* (2020), Springer, pp. 798–827.

[26] Ghazi, B., Pagh, R., and Velingker, A. Scalable and differentially private distributed aggregation in the shuffled model. *arXiv preprint arXiv:1906.08320* (2019).

[27] Huang, Z., Liang, Y., and Yi, K. Instance-optimal mean estimation under differential privacy. *Advances in Neural Information Processing Systems* (2021).

[28] Kaggle. San francisco city employee salary data. https://www.kaggle.com/datasets/kaggle/sf-salaries/data, 2014.

[29] Kaggle. Japan's 100 million customs trade statistics since 1988. https://www.kaggle.com/datasets/zanjibar/100-million-data-csv, 2020.

[30] Kaggle. Mnist - digit recognizer dataset. https://www.kaggle.com/c/digit-recognizer/data, 2020.

[31] Kaggle. Monthly salary of public worker in brazil. https://www.kaggle.com/datasets/gustavomodelli/monthly-salary-of-public-worker-in-brazil, 2020.

[32] Kaggle. Ontario public sector salary 2019. https://www.kaggle.com/datasets/rajacsp/ontario, 2020.

[33] Kamath, G., Li, J., Singhal, V., and Ullman, J. Privately learning high-dimensional distributions. In *Conference on Learning Theory* (2019), PMLR, pp. 1853–1902.

[34] Kamath, G., Singhal, V., and Ullman, J. Private mean estimation of heavy-tailed distributions. In *Conference on Learning Theory* (2020), PMLR, pp. 2204–2235.

[35] Kamath, G., and Ullman, J. A primer on private statistics. *arXiv preprint arXiv:2005.00010* (2020).

[36] McMahan, H. B., Ramage, D., Talwar, K., and Zhang, L. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963* (2017).

[37] McSherry, F. D. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data* (2009), pp. 19–30.

[38] Pass, G., Chowdhury, A., and Torgeson, C. A picture of search. In *Proceedings of the 1st international conference on Scalable information systems* (2006).

[39] Pichapati, V., Suresh, A. T., Yu, F. X., Reddi, S. J., and Kumar, S. Adaclip: Adaptive clipping for private sgd. *arXiv preprint arXiv:1908.07643* (2019).

[40] Schaal, D. Expedia ceo's total compensation pegged at $296 million for 2021. https://skift.com/blog/expedia-ceos-total-compensation-pegged-at-296-million-for-2021, 2022.

[41] Song, S., Chaudhuri, K., and Sarwate, A. D. Stochastic gradient descent with differentially private updates. In *2013 IEEE global conference on signal and information processing* (2013), IEEE, pp. 245–248.

[42] Stemmer, U. Locally private k-means clustering. *The Journal of Machine Learning Research 22*, 1 (2021), 7964–7993.

[43] Stemmer, U., and Kaplan, H. Differentially private k-means with constant multiplicative error. *Advances in Neural Information Processing Systems 31* (2018).

[44] Tao, Y., He, X., Machanavajjhala, A., and Roy, S. Computing local sensitivities of counting queries with joins. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (2020), pp. 479–494.

[45] Vadhan, S. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*. Springer, 2017, pp. 347–450.

A Computational Issue of the Two-round Protocal in [27]

To obtain a good clipping threshold τ under the shuffle-DP model, [27] applies the method from [22] to approximate $\mathrm{Max}(D)$. In [22], each randomizer has a computation of $O(n \log^2 U)$. Additionally, both [27] and [22] only present their theoretical results without any concrete implementation. Our implementation with domain compression still resulted in a long running time, failing to give the results within a couple of days in our experimental settings.