Distributed Differential Privacy via Shuffling Versus Aggregation: A Curious Study

Yu Wei, Jingyu Jia, Yuduo Wu, Changhui Hu, Changyu Dong, Zheli Liu, Xiaofeng Chen, *Senior Member, IEEE*, Yun Peng, and Shaowei Wang, *Member, IEEE*

***Abstract*— How to achieve distributed differential privacy (DP) without a trusted central party is of great interest in both theory and practice. Recently, the shuffle model has attracted much attention. Unlike the local DP model in which the users send randomized data directly to the data collector/analyzer, in the shuffle model an intermediate untrusted shuffler is introduced to randomly permute the data, which have already been randomized by the users, before they reach the analyzer. The most appealing aspect is that while shuffling does not explicitly add more noise to the data, it can make privacy better. The privacy amplification effect in consequence means the users need to add less noise to the data than in the local DP model, but can achieve the same level of differential privacy. Thus, protocols in the shuffle model can provide better accuracy than those in the local DP model. What looks interesting to us is that the architecture of the shuffle model is similar to private aggregation, which has been studied for more than a decade. In private aggregation, locally randomized user data are aggregated by an intermediate untrusted aggregator. Thus, our question is whether aggregation also exhibits some sort of privacy amplification effect? And if so, how good is this "aggregation model" in comparison with the shuffle model. We conducted the first comparative study between the two, covering privacy amplification, functionalities, protocol accuracy, and practicality. The results as yet suggest that the new shuffle model does not have obvious advantages over the old aggregation model. On the contrary, protocols in the aggregation model outperform those in the shuffle model, sometimes significantly, in many aspects.**

Manuscript received 14 October 2021; revised 23 February 2023, 18 July 2023, and 29 October 2023; accepted 27 November 2023. Date of publication 9 January 2024; date of current version 15 January 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62072132 and Grant 62261160651, in part by the National Key Research and Development Program of China under Grant 2020YFB1005700, and in part by the Engineering and Physical Sciences Research Council of U.K. under Grant EP/M013561/2. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Dali Kaafar. *(Corresponding authors: Changyu Dong; Zheli Liu.)*

Yu Wei, Jingyu Jia, Yuduo Wu, and Zheli Liu are with the College of Cyber Science and the College of Computer Science, Key Laboratory of Data and Intelligent System Security, Ministry of Education, Nankai University, Tianjin 300350, China (e-mail: stoneboat@mail.nankai.edu.cn; jiajingyu@mail.nankai.edu.cn; doria@mail.nankai.edu.cn; liuzheli@nankai.edu.cn).

Changhui Hu is with the School of Cyberspace Security and the School of Cryptology, Hainan University, Haikou 570228, China (e-mail: chu@hainanu.edu.cn).

Changyu Dong, Yun Peng, and Shaowei Wang are with the Institute of Artificial Intelligence, Guangzhou University, Guangzhou 511370, China (e-mail: changyu.dong@gzhu.edu.cn; yun-peng@gzhu.edu.cn; wangsw@gzhu.edu.cn).

Xiaofeng Chen is with the School of Cyber Engineering, Xidian University, Xi'an 710071, China (e-mail: xfchen@xidian.edu.cn).

Digital Object Identifier 10.1109/TIFS.2024.3351474

***Index Terms*— Differential privacy, shuffle model, aggregation model.**

I. Introduction

TODAY, data are more valuable than ever. Data are a key driver behind technological innovations that enable companies to provide more competitive and reliable products and services. In fact, the success of big name Internet companies, such as Google and Facebook, is largely due to the vast amount of data they collect from their users. While collecting data from users can provide clear benefits for businesses, it also means hefty privacy risks. With increasingly stricter privacy laws and regulations, companies are obliged to provide adequate protection to the data they collect, store, and process.

Differential privacy (DP) [1] has been regarded by many as a promising Privacy Enhancing Technology (PET). It allows companies to collect and share aggregated data while maintaining the privacy of individual users. Traditionally, DP was studied in the central model where a trusted data collector collects raw data from the users, then processes the data with a differentially private algorithm and publishes the results. Central DP guarantees the privacy of the final published statistics. However, the assumption that the data collector is trusted is too strong. In many real-world scenarios, it is just not possible for the users to trust the data collector. This led to the development of distributed DP mechanisms. One popular approach is local DP. In local DP, each user randomizes his/her data before sending it to the data collector. Under local DP, data are already differentially private when it leaves the user's control. Thus, the data collector cannot see the raw data and does not need to be trusted. Yet, the randomized data from the users still allow the data collector to extract useful statistics. Local DP mechanisms have been deployed by big names such as Google [2], Apple [3], and Microsoft [4] in their services, to encourage users to share their data.

While local DP is appealing in many ways, it has one vital weakness. Compared to central DP, the amount of noise being added is much larger, which causes excessive obfuscation, hence the loss of utility. This motivated the recent research of distributed DP with enhanced utility [5], [6], [7]. One notable line of research in this direction is DP protocols in the shuffle model [5], [8], [9], [10]. In the shuffle model, an additional untrusted shuffler is placed between the users and the data collector (analyst). Each user randomizes his/her data and

then sends it to the shuffler. After receiving all users' data, the shuffler randomly shuffles the data before sending them to the data collector. At first glance, it is not obvious what is the benefit of the shuffle model in comparison with the local DP model. However, in-depth analysis [8], [10] showed that introducing the shuffler can provide privacy amplification. That is, in the shuffle model, to achieve the desired differential privacy level, less noise is needed to be added by the users to their data compared to the local DP model. It has been shown in [8] and [9] that the accuracy of the statistics produced by the shuffle model is somewhere in between the local and the central model. Also, the shuffle model does not require a strong trust assumption as the central model because neither the shuffler nor the data collector needs to be trusted. Because of all these advantages, the shuffle model has attracted much attention from the research community [5], [8], [9], [10], [11], [12], [13], [14], [15], [16].

What inspired the study we present in this paper is the observation that the architecture of the shuffle model resembles that of private aggregation [17], [18], [19], [20], which has already been studied for more than a decade. In many private aggregation schemes, each user adds noise locally to his/her data and then sends it to an untrusted aggregator, who then outputs a differentially private aggregate to the final data collector. It is interesting to ask whether aggregation can achieve the privacy amplification effect, like protocols in the shuffle model. If so, which is better in various dimensions, such as privacy, utility, functionality, and efficiency?

Contributions: In this paper, we conducted the first comparative study between differentially private protocols in the shuffle model and those based on private aggregation.

- As the first step, we formally defined the aggregation model that captures private aggregation and is comparable to the shuffle model in the architecture, trust assumptions, and practical settings. As examples, we also gave two concrete protocols in the aggregation model, which are transformed from the well-known Gaussian and Laplace mechanisms in the central DP model.
- Our analysis revealed that protocols in the aggregation model can provide privacy amplification. Although much more analysis still needs to be done to fully understand this effect in the aggregation model, our initial results showed that protocols in the aggregation model can amplify privacy at least as well as those in the shuffle model and, in some cases, can do better.
- We showed that protocols in the aggregation model can support all algorithms in the Statistical Query (SQ) model. This is on par with the power of protocols in the shuffle model (see [9]).
- We compared the accuracy of aggregation protocols and shuffle protocols for a diverse set of tasks, including summation, histogram, top-k, sorting, SGD, and PCA. The results demonstrated the effectiveness of the aggregation protocols for each task, providing compelling evidence that the iterative use of summation does not render the aggregation model ineffective. Moreover, we consistently observed superior performance of the aggregation protocols compared to their corresponding shuffle protocols from both theoretical and empirical perspectives.
- In terms of practicality, we found that one constraint of protocols in the shuffle model is that they often require the user numbers to exceed a lower bound. This is generally not the case in aggregation protocols. Also, contrary to the claim in [9], we found private aggregation can be implemented much more efficiently than secure shuffle with state-of-the-art cryptographic protocols or trusted hardware.

We hope our findings in this paper could spark further discussion in the community, provide useful input to the future design of distributed privacy mechanisms, and help practitioners make better-informed decisions.

II. Preliminaries

A. Differential Privacy

Differential privacy is a mathematical definition of privacy with rigorous guarantees. If an individual's private record is used as part of the input dataset, the output from a differentially private mechanism has the property that anyone can learn almost nothing more about the individual than if that person's record were absent from the dataset. Informally, this intuition is captured by requiring that the output distributions produced by a mechanism should differ only slightly when accessing any two datasets that differ from each other only in one element. Formally, differential privacy is defined as the following:

Definition 1 (Differential Privacy [1]): Let $\varepsilon \geq 0$ *and* $\delta \in [0, 1)$. *A randomized mechanism* $M : \mathcal{X}^n \to \mathcal{Z}$ *satisfies* $(\varepsilon, \delta)-DP$ *if for any* $X, X' \in \mathcal{X}^n$ *that differ in only one element, and any* $Z \subseteq \mathcal{Z}$, *it holds*

$$\mathbb{P}[M(X) \in Z] \leq e^{\varepsilon}\mathbb{P}[M(X') \in Z] + \delta.$$

Note that in the above definition, the mechanism must be run by a trusted data collector/owner who has full access to the raw input dataset. However, in many scenarios, this trust may not exist. Thus, the notion of local differential privacy was proposed that requires each piece of data in the dataset to be randomized. If each individual locally randomizes his/her own record before handing it to the untrusted data collector, differential privacy guarantee also holds for the collector. Formally, local differential privacy is defined as the following:

Definition 2 (Local Differential Privacy [21]): Let $\varepsilon \geq 0$ *and* $\delta \in [0, 1)$. *A local randomizer* $R : \mathcal{X} \to \mathcal{Y}$ *satisfies* $(\varepsilon, \delta)-LDP$ *if and only if for any input* $x, x' \in \mathcal{X}$ *and any* $Y \subseteq \mathcal{Y}$, *it holds*

$$\mathbb{P}[R(x) \in Y] \leq e^{\varepsilon}\mathbb{P}[R(x') \in Y] + \delta.$$

B. Gaussian and Laplace Mechanisms

Now we briefly review the Gaussian mechanism and the Laplace mechanism in the central model. These two mechanisms achieve differential privacy by adding noise drawn from Gaussian (Laplace) distribution to query results. Since the noise added is closely related to the notion of global sensitivity, we first recall it as follows.

Definition 3 (Global Sensitivity [1]): Given any function $f : \mathcal{X}^n \to \mathbb{R}^d$, for any $X, X' \in \mathcal{X}^n$ that differ in only one element, the global sensitivity of the function f is:

$$\Delta f = \max_{X,X'} ||f(X) - f(X')||_p,$$

where $||\cdot||p$ is the l_p norm.

The value of the global sensitivity is decided by the query function f and the input domain. For instance, for real-valued summation function $f(x_1, \cdots, x_n) = \sum x_i$ with $x_i \in [-a, a]$, the value of the global sensitivity of f is $\Delta f = 2a$.

The Laplace mechanism [22] adds random noise drawn from the Laplace distribution to the results. We first recall that the Laplace distribution is defined as the following:

Definition 4 (The Laplace Distribution): The Laplace distribution (centered at 0) with scale b is the distribution with probability density function:

$$Lap(x \mid b) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right).$$

We use $Lap(b)$ to denote the Laplace distribution with scale b and the variance of $Lap(b)$ is $2b^2$.

Theorem 1 (Laplace Mechanism): Given any function $f : \mathcal{X}^n \to \mathbb{R}^d$, Δf measured by the l_1 distance, Laplace Mechanism defined as

$$M_L(X) = f(X) + (N_{L,1}, N_{L,2}, \cdots, N_{L,d})$$

provides ε-DP, where $\{N_{L,i}\}_{i\in[d]}$ are random variables independently drawn from $Lap(\Delta f/\varepsilon)$.

The Laplace distribution has a property called infinite divisibility [23] (Proposition 2.4.1), which says a Laplacian random variable can be expressed as a sum of i.i.d random variables (that follow a certain distribution). Later we will use this property to implement the distributed Laplace mechanism in Section III-C.

Theorem 2 (Infinite Divisibility of Laplace Distribution): Given a Laplace distribution $Lap(b)$, for any $n \in \mathbb{N}^+$, there exists a Gamma distribution $Ga(n, b)$ with probability density function:

$$\frac{(1/b)^{1/n}}{\Gamma(1/n)} x^{\frac{1}{n}-1} e^{-x/b},$$

such that $\sum_{i=1}^{n} (\gamma_{i,1} - \gamma_{i,2})$ follows $Lap(b)$, where $\gamma_{i,1}, \gamma_{i,2}$ are independently drawn from $Ga(n, b)$.

The Gaussian mechanism adds random noise to results. The noise follows the Gaussian distribution.

Definition 5 (Gaussian Distribution): Gaussian distribution with expectation 0 and variance σ^2 is the distribution with probability density function:

$$\mathcal{N}(x|\sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right). \quad (1)$$

We use $\mathcal{N}(0, \sigma^2)$ to denote the Gaussian distribution with expectation 0 and variance σ^2 in the rest of the paper.

Theorem 3 (Gaussian Mechanism): Given any function $f : \mathcal{X}^n \to \mathcal{R}^d$, global sensitivity Δf measured by the l_2 distance, the Gaussian Mechanism defined as

$$M_G(X) = f(X) + (N_{G,1}, N_{G,2}, \cdots, N_{G,d})$$

provides (ε, δ)-DP, where $\{N_{G,i}\}_{i\in[d]}$ are random variables independently drawn from $\mathcal{N}(0, \sigma^2)$ with $\sigma = \frac{\Delta f \sqrt{2log(1.25/\delta)}}{\varepsilon}$.

C. Local Model

In [24] a so-called local model was defined to capture private computation with local differential privacy. In the local model, algorithms cannot access the raw dataset, but only via local randomizers:

Definition 6 (Local Randomizers [24]): An ε-local randomizer $R : \mathcal{X} \to \mathcal{Y}$ is an ε-differentially private algorithm that takes a database of size $n = 1$. That is, $\mathbb{P}[R(x) = y] \leq e^{\varepsilon}\mathbb{P}[R(x') = y]$ for all $x, x' \in \mathcal{X}$ and all $y \in \mathcal{Y}$. The probability is taken over the coins of R (but not over the choice of the input).

Definition 7 (Local Oracles [24]): Let $X = (x_1, \ldots, x_n) \in \mathcal{X}^n$ be a database. An LR oracle $LR_X(\cdot, \cdot)$ gets an index $i \in [n]$ and an ε-local randomizer R, and outputs a random value $y \in \mathcal{Y}$ chosen according to the distribution $R(x_i)$. The distribution $R(x_i)$ depends only on the entry x_i in X.

Definition 8 (Local Algorithms [24]): An algorithm is ε-local if it accesses the database X via the oracle LR_X with the following restriction: for all $i \in [n]$, if $LR_X(i, R_1), \ldots, LR_X(i, R_k)$ are the algorithm's invocations of LR_X on index i,where each R_j is an ε_j-local randomizer, then $\varepsilon_1 + \cdots + \varepsilon_k \leq \varepsilon$.

D. SQ Model

To obtain differential privacy, noise often needs to be added to the data. One frequently asked question is whether the output is still useful. An often-used theoretical framework to answer this question is the Statistical Query (SQ) model [25]. In the SQ model, computational tasks are formulated as learning algorithms. Let $\mathcal{C}$ be a class of $\{-1, +1\}$-valued functions (also called *learning concepts*) over an input space $\mathcal{X}$. The aim of a learning algorithm is to learn a concept c. Normally, a learning algorithm is given examples randomly chosen from some unknown distribution $\mathcal{P}$ over $\mathcal{X}$ and should produce a hypothesis of c. In the SQ model, a learning algorithm (or SQ algorithm for short) instead of having access to examples, has only access to statistical properties of the distribution $\mathcal{P}$. Formally, the ability to access statistical properties is abstracted as SQ Oracle:

Definition 9 (SQ Oracle [24]): Let $\mathcal{P}$ be a distribution over a domain $\mathcal{X}$. An SQ oracle $SQ_{\mathcal{P}}$ takes as input a function $g : \mathcal{X} \to \{-1, +1\}$ and a tolerance parameter $\tau \in [0, 1]$. Its output v satisfies

$$|v - \underset{u\sim\mathcal{P}}{E}[g(u)]| \leq \tau,$$

where u denotes a random sample.

In the above definition, the SQ oracle takes as input a statistical query of the form (g, τ), where g is a $\{-1, +1\}$-valued query function on input u from domain $\mathcal{X}$, and $\tau \in [0, 1]$ is a tolerance parameter. It outputs an estimation for the expectation of g over the $\mathcal{P}$ that is accurate with additive error $\pm\tau$. An SQ algorithm can only access the distribution $\mathcal{P}$ indirectly via the SQ oracle $SQ_{\mathcal{P}}$. The significance of the

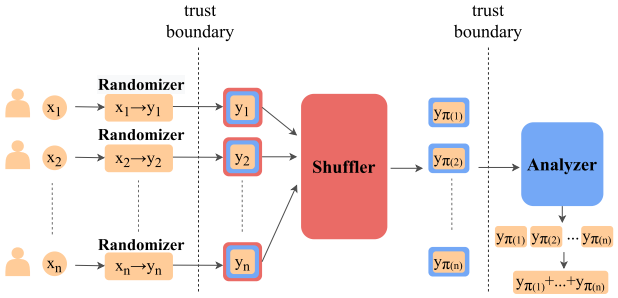


Fig. 1. Shuffle model.

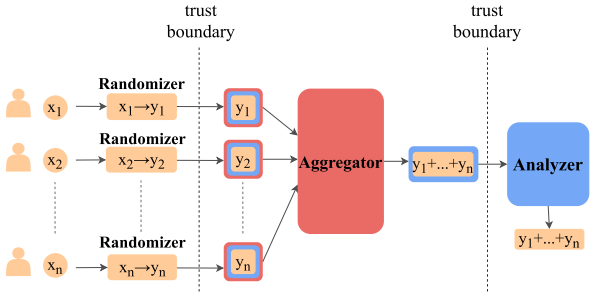


Fig. 2. Aggregation model.

SQ model is that any SQ algorithm can be automatically converted to a learning algorithm in the presence of certain noise, thus leading to noise-tolerant algorithms. It has been shown in [24] that the Local model (Section II-C) is equivalent to the SQ model, i.e., a concept class is learnable by a local differentially private algorithm if and only if it is learnable in the SQ model. Similarly, we will prove in Section V-A that private aggregation is sufficient to support any SQ algorithms.

Computation Tasks Supported by the SQ Model: In the SQ model, we can compute the bounded real-valued statistical query. We can compute almost every learning algorithm that works in the Probably Approximately Correct (PAC) model (with the exception of parity learning algorithms). For example, we can compute singular value decomposition, principal component analysis, k-means clustering, decision tree, and gradient descent.

III. Shuffle Model and Aggregation Model

In this section, we will introduce the shuffle model and aggregation model. In both models, there are n users, and each holds a piece of data $x_i \in \mathcal{X}$. We denote the n users' record using the vector $X = (x_1, \cdots, x_n)$.

A. Shuffle Model

The architecture of the shuffle model is illustrated in Figure 1, and we review the definition of the protocol in the shuffle model as the following:

Definition 10 (Shuffle Model [5], [9]): A protocol P in the shuffle model consists of three randomized algorithms:

- *A randomizer $R : \mathcal{X} \to \mathcal{Y}^m$ that takes as input a single user's record x_i and outputs a set of message $y_{i,1}, \cdots, y_{i,m} \in \mathcal{Y}$. If $m = 1$, then P is in the single-message shuffle model.*
- *A shuffler $S : \mathcal{Y}^* \to \mathcal{Y}^*$ that takes a set of messages and outputs these messages in a uniformly random order. Specifically, on input $y_1, \cdots, y_\mathcal{N}$, S chooses a uniformly random permutation $\pi : [\mathcal{N}] \to [\mathcal{N}]$ and outputs $y_{\pi(1)}, \cdots, y_{\pi(\mathcal{N})}$.*
- *An analyzer $A : \mathcal{Y}^* \to \mathcal{Z}$ that takes a set of messages $y_1, \cdots, y_\mathcal{N}$ and attempts to estimate some funtion $f(x_1, \cdots, x_n)$ from these messages.*

With this setup, we review the following definition of differential privacy in the shuffle model.

Definition 11 (Differential Privacy in the Shuffle Model): A protocol $P = (R, S, A)$ is (ε, δ)-differentially private in the shuffle model if, for $n \in \mathbb{N}^+$, the algorithm $(S \circ R^n)(X) := S(R(x_1), \cdots, R(x_n))$ is (ε, δ)-differentially private.

B. Aggregation Model

Figure 2 illustrates the aggregation model, whose architecture is similar to that of the shuffle model. We define the protocol in the aggregation model as the following:

Definition 12 (Aggregation Model): A protocol P in the aggregation model consists of three randomized algorithms:

- *A randomizer $R : \mathcal{X} \to \mathcal{Y}$ that takes as input a single user's record x_i and outputs a message $y_i \in \mathcal{Y}$.*
- *An aggregator $G : \mathcal{Y}^n \to \mathcal{Y}$ that takes n messages and aggregates messages. Specifically, on input $y_1, \cdots, y_n$, G outputs $z = \sum_{i=1}^{n} y_i$.*
- *An analyzer $A : \mathcal{Y} \to \mathcal{Z}$ that computes statistic $f(z)$ upon the obtained aggregate.*

We further define the definition of differential privacy in the aggregation model as the following:

Definition 13: (Differential Privacy in the Aggregation Model) A protocol $P = (R, G, A)$ is (ε, δ)-differentially private in the aggregation model if, for all $n \in \mathbb{N}^+$, the algorithm $(G \circ R^n)(X) := G(R(x_1), \cdots, R(x_n))$ is (ε, δ)-differentially private.

C. Concrete Protocols

To facilitate discussion, here we show two concrete protocols in the aggregation model. The two aggregation protocols presented here are the Gaussian aggregation protocol $A^G_{n,a,\sigma}$ and the Laplace aggregation protocol $A^L_{n,a,b}$, named after the noise distributions they use for randomizing the result. In Appendix B of the full version [26] of our paper, we also include three shuffle protocols from [8] and [9], for computing the sum of bits or real numbers in the range of $[0, 1]$.

The Gaussian aggregation protocol $A^G_{n,a,\sigma}$ is shown in Figure 3. Each Randomizer in $A^G_{n,a,\sigma}$ adds noise following Gaussian distribution. $A^G_{n,a,\sigma}(X)$ is (ε, δ)-DP for any $\varepsilon > 0$, $\delta \in (0, 1)$, $n \in \mathbb{N}^+$, and $X = (x_1, \cdots, x_n) \in [-a, a]^n$, when $\sigma = \frac{2a\sqrt{2\log 1.25/\delta}}{\varepsilon}$.

The Laplace aggregation protocol $A^L_{n,a,b}$ is shown in Figure 4. It utilizes the infinite divisibility of Laplace distribution (Section II-B) to add Laplace noise to the aggregated results. $A^L_{n,a,b}(X)$ is $(\varepsilon, 0)$-DP for any $\varepsilon > 0$, $n \in \mathbb{N}^+$ and $X = (x_1, \cdots, x_n) \in [-a, a]^n$, when $b = 2a/\varepsilon$.

Gaussian aggregation protocol $A^G_{n,a,\sigma} = (R^a_{n,\sigma}, G, A)$

Randomizer $R^a_{n,\sigma}(x)$:
Input: $x \in [-a, a]$, $\sigma > 0, n \in \mathbb{N}^+$
Output: $y \in \mathbb{R}$
1) Let $\sigma_0 = \frac{\sigma}{\sqrt{n}}$ and sample random variable n_g from Gaussian distribution $\mathcal{N}(0, \sigma_0^2)$.
2) Return $y = x + n_g$;

Aggregator $G(y_1, \cdots, y_n)$:
Input: $(y_1, \cdots, y_n) \in \mathbb{R}^n$
Output: $z \in \mathbb{R}$
1) Return $z = \sum_{i=1}^{n} y_i$;

Analyst A receives z;

Fig. 3. Protocol: $A^G_{n,a,\sigma}$.

Laplace aggregation protocol $A^L_{n,a,b} = (R^a_{n,b}, G, A)$

Randomizer $R^a_{n,b}(x)$:
Input: $x \in [-a, a]$, $b > 0, n \in \mathbb{N}^+$
Output: $y \in \mathbb{R}$
1) Sample random variables γ_1, γ_2 from gamma distribution $Ga(n, b)$
2) Return $y = x + \gamma_1 - \gamma_2$;

Aggregator $G(y_1, \cdots, y_n)$:
Input: $(y_1, \cdots, y_n) \in \mathbb{R}^n$
Output: $z \in \mathbb{R}$
1) Return $z = \sum_{i=1}^{n} y_i$;

Analyst A receives z;

Fig. 4. Protocol: $A^L_{n,a,b}$.

IV. Privacy Amplification

One attractive property of protocols in the shuffle model is that they can achieve privacy amplification. That is, the output of the local randomizers satisfies only a weaker notion of differential privacy, but after shuffling, the output satisfies a stronger notion of differential privacy. Privacy amplification is the key reason why the shuffle model can achieve better utility than the local model. Can protocols in the aggregation model also achieve some level of privacy amplification? If so, how can we compare the level of privacy amplification between protocols in the shuffle and aggregation models? This is our first question.

A. Concrete Aggregation Protocol Privacy Amplification Analysis

To start with, we first investigate whether the two concrete aggregation protocols can amplify privacy. The answer is yes.

The following theorem states that for the Gaussian aggregation protocol, the differential privacy that the whole protocol can achieve is better than what the local randomizer alone achieves. More specifically, if the randomizer's output satisfies $(\sqrt{n}\varepsilon_A, \delta)$-differential privacy, then the whole protocol is (ε_A, δ)-differentially private. The amplification factor depends on n, the number of users. Therefore, the more users participating in the protocol, the stronger the amplification will be.

Theorem 4: Let $A^G_{n,a,\sigma}$ be the Gaussian aggregation protocol and $R^a_{n,\sigma}(x) : [-a, a] \to \mathbb{R}$ be the local randomizer, as defined in Figure 3. If $A^G_{n,a,\sigma}$ satisfies (ε_A, δ)-differential privacy, then $R^a_{n,\sigma}(x)$ is $(\sqrt{n}\varepsilon_A, \delta)$-differentially private.

The theorem follows directly from how noise is distributedly generated in the protocol and the differential privacy of the central Gaussian mechanism. The proof is straightforward and, thus, is omitted.

The analysis of the Laplace aggregation protocol is more involved because the noise distribution is rather complex. Yet, we can still show that the output of the whole protocol can be better in terms of privacy than that of the local randomizer. We summarize the amplification theorem of the Laplace aggregation protocol in Theorem 5 (Proof is available in Appendix C of the full version [26] of our paper).

Theorem 5: Let $A^L_{n,a,b} = (R^a_{n,b}, G, A)$ be the Laplace aggregation protocol and $R^a_{n,b} : [-a, a] \to \mathbb{R}$ be the local randomizer, as defined in Figure 4. If $A^L_{n,a,b}$ satisfies ε_A-differentially private, $R^a_{n,b}$ is (ε_L, δ)-differential privacy such that $\varepsilon_A < \varepsilon_L$, and $0 < \delta < \Delta$ for some $\Delta \in (\frac{1}{2}, 1)$.

B. Comparison Analysis With the Shuffle Model

We have proved that the two concrete aggregation protocols can achieve privacy amplification. Then, we are interested in asking can we compare the level of privacy amplification between shuffle protocols and the aggregation protocols. In particular, is there a separation of the privacy amplification ability between the protocols in the two models. We report two interesting observations as the following.

Informally, Theorem 6 says that if an aggregation protocol uses the same local randomizer as a shuffle protocol, then the aggregation protocol can achieve at least the same privacy as the shuffle protocol. In turn, it equals to say that adopting the same local randomizer in the aggregation model and the (single-message) shuffle model, the privacy amplification of the resulting aggregation protocol is at least as strong as that provided by the resulting shuffle model.

Theorem 6: Let $P_S = (R, S, A)$ be a single-message shuffle model protocol and $P_A = (R, G, A)$ an aggregation model protocol. The randomizer $R : \mathcal{X} \to \mathcal{Y}$ satisfies (ε_L, δ)-differential privacy. For $n \in \mathbb{N}^+$, if P_S is (ε_S, δ)-differentially private, then P_A is (ε_S, δ)-differentially private.

Proof: Recall n is the number of users participating in the shuffle protocol P_S and the aggregation protocol P_A. Without loss of generality, we fix $n \in \mathbb{N}^+$ as an arbitrary positive integer in the following.

Let $\mathcal{T}$ be the range of the aggregation protocol P_A, and let $\mathcal{W}$ be the range of the shuffle protocol P_S. By theorem's condition, if P_S is (ε_S, δ)-DP, we have for every neighboring input $X, X' \in \mathcal{X}^n$ which only differ by one coordinate and every event $W \in \mathcal{W}$, the following inequality holds

$$\mathbb{P}[P_S(X) \in W] \leq e^{\varepsilon_S}\mathbb{P}[P_S(X') \in W] + \delta. \tag{2}$$

Without loss of generality, let $X = (x_1, \cdots, x_{n-1}, x_n) \in \mathcal{X}^n$, $X' = (x_1, \cdots, x_{n-1}, x'_n) \in \mathcal{X}^n$ be an arbitrary pair of

datasets, which differs at the n-th coordinates, and let $T \subseteq \mathcal{T}$ be an arbitrary event. Using event T, we define the event $W = \{(r_1, \cdots, r_n) \in \mathcal{Y}^n | \sum_{i=1}^{n} r_i \in T\}$, which is the set of all n-elements tuples whose sum is in the set T. Then, we have $\mathbb{P}[P_S(X) \in W] = \mathbb{P}[P_A(X) \in T]$. This is because

$$\begin{aligned}
&\mathbb{P}[P_S(X) \in W] \\
&\quad = \mathbb{P}[S(R(x_1), \cdots, R(x_n)) \in W] \\
&\qquad\qquad \text{(by protocol's Definition (Def. 10))} \\
&\quad = \mathbb{P}[(R(x_1), \cdots, R(x_n)) \in W] \\
&\text{(event } W \text{ is not sensitive to uniformly random permutation)} \\
&\quad = \mathbb{P}[\sum_{i=1}^{n} R(x_i) \in T] \qquad \text{(by event } W\text{'s definition)} \\
&\quad = \mathbb{P}[P_A(X) \in T].
\end{aligned}$$

Similarly, we have $\mathbb{P}[P_S(X') \in W] = \mathbb{P}[P_A(X') \in T]$. Combining with Inequality 2, we can see that, if P_S is (ε_S, δ)-DP, then for every neighboring input $X, X' \in \mathcal{X}^n$ which only differ by one coordinate and every event $T \in \mathcal{T}$, the following inequality holds

$$\mathbb{P}[P_A(X) \in T] \leq e^{\varepsilon_S} \mathbb{P}[P_A(X') \in T] + \delta.$$

That is, P_A is (ε_S, δ)-DP. □

Theorem 7 provides a separation of the privacy amplification ability between protocols in the aggregation models and the shuffle models. Theorem 7 says that there exist aggregation protocols that can provide meaningful ε-differential privacy amplification. In comparison, it has been shown in [27] (claim 4.2) that if the protocol has to satisfy ε-differential privacy (rather than (ε, δ)-differential privacy), then (single-message) shuffle model cannot offer privacy amplification.

Theorem 7: Let $R : \{-1, 1\} \mapsto \{-2, -1, 1, 2\}$ be a ε_L-differential private randomizer, which takes as input x from the set $\{-1, 1\}$, and outputs x with probability $\frac{e^{\varepsilon_L}}{e^{\varepsilon_L}+3}$, or a value from $\{-2, -1, 1, 2\} \backslash x$ with probability $\frac{1}{e^{\varepsilon_L}+3}$. Then, the aggregation protocol $P_A = (R, G, A)$ satisfies ε_A-differential privacy, where

$$\varepsilon_A = max\{\ln \frac{2e^{\varepsilon_L} + 1}{3}, \ln \frac{e^{3\varepsilon_L} + 3}{e^{2\varepsilon_L} + e^{\varepsilon_L} + 2}\}.$$

Proof: We first construct a randomizer $R : \{-1, 1\} \to \{-2, -1, 1, 2\}$ which works as the following: it takes a value $x \in \{-1, 1\}$ as input, and then with probability $\frac{e^{\varepsilon_L}}{e^{\varepsilon_L}+3}$ returns x, with the rest of probability it uniformly returns a value from $\{-2, -1, 1, 2\} \backslash \{x\}$.

Without loss of generality, let $X = (x_1, \cdots, x_n, x_{n+1}) \in \mathcal{X}^{n+1}$, $X' = (x_1, \cdots, x_n, x'_{n+1}) \in \mathcal{X}^{n+1}$ be an arbitrary pair of datasets, which differs at the $(n+1)$-th coordinates. We further fix $x_{n+1} = -1$ and $x'_{n+1} = 1$, which does not harm the proof's generality. Let $n + 1$ be the number of users, $Y_n = \sum_{i=1}^{n} R(x_i)$, $Z = R(x_{n+1})$, $Z' = R(x'_{n+1})$. For every $k \in \{-2n-2, -2n-1, \cdots, 2n+1, 2n+2\}$, the probability ratio evaluating at the point k is

$$\begin{aligned}
&\frac{Pr[\sum_{i=1}^{n} R(x_i) + R(x_{n+1}) = k]}{Pr[\sum_{i=1}^{n} R(x_i) + R(x'_{n+1}) = k]} \\
&= \frac{Pr[Y_n + Z = k]}{Pr[Y_n + Z' = k]} \\
&= \frac{\sum_{j \in \{-2,-1,1,2\}} Pr[Y_n = k - j] Pr[Z = j]}{\sum_{j \in \{-2,-1,1,2\}} Pr[Y_n = k - j] Pr[Z' = j]}. \qquad (3)
\end{aligned}$$

To show that $\varepsilon_A < \varepsilon_L$, it suffices to show for every $k \in \{-2n-2, -2n-1, \cdots, 2n+1, 2n+2\}$ and every assignment of $(x_1, \cdots, x_{n-1}, x_n)$, one of the quantities $Pr[Y_n = k + 2]$, $Pr[Y_n = k - 1]$, $Pr[Y_n = k - 2]$ is larger than 0 when $Pr[Y_n = k + 1] > 0$. Because for the case $-2n - 2 \leq k \leq -2n - 1$ and $2n + 1 \leq k \leq 2n + 2$ (when $Pr[Y_n = k + 1] = 0$), we have

$$(3) \leq \max\{1, e^{-\varepsilon_L}\} < e^{\varepsilon_L}.$$

For the case $-2n - 2 \leq k \leq 2n + 2$ and $-2n \leq k + 1 \leq 2n$ (when $Pr[Y_n = k + 1] > 0$), we know that when $n > 1$, one of the quantity $k + 2, k - 1, k - 2$ must be in the set $k \in \{-2n, -2n + 1, \cdots, 2n - 1, 2n\}$, and hence one of the quantities $Pr[Y_n = k + 2]$, $Pr[Y_n = k - 1]$, $Pr[Y_n = k - 2]$ must be larger than 0. So far, we conclude that $\varepsilon_A < \varepsilon_L$.

We are interested in how large the privacy amplification in this aggregation protocol can be, in other words, how large the difference between ε_L and ε_A can be.

Let $f(n, k)$ be the probability ratio evaluating at the point k for the neighboring dataset X, X'. Formally,

$$f(n, k) = \frac{Pr[P_A(X) = k]}{Pr[P_A(X') = k]}.$$

Recall $Y_n = \sum_{i=1}^{n} R(x_i)$, $Z = R(x_{n+1})$, $Z' = R(x'_{n+1})$, then we have

$$\begin{aligned}
f(n, k) &= \frac{Pr[P_A(X) = k]}{Pr[P_A(X') = k]} \\
&= \sum_{j \in \{-2,-1,1,2\}} \frac{Pr[Y_{n-1} + Z = k - j] Pr[R(x_n) = j]}{Pr[Y_{n-1} + Z' = k - j] Pr[R(x_n) = j]} \\
&\leq \max_{j \in \{-2,-1,1,2\}} \left\{ \frac{Pr[Y_{n-1} + Z = k - j]}{Pr[Y_{n-1} + Z' = k - j]} \right\}. \qquad (4)
\end{aligned}$$

Let $h(n - 1, k - j) = \frac{Pr[Y_{n-1}+Z=k-j]}{Pr[Y_{n-1}+Z'=k-j]}$, which is the inner expression of the right hand-side of Inequality 4. We observe that the maximum probability ratio for the neighboring dataset X, X' is obtained at $n = 2$. This is because

$$\begin{aligned}
&\max_{k \in \{-2n-2, \cdots, 2n+2\}} \{f(n, k)\} \\
&\quad \leq \max_{\substack{k \in \{-2n-2, \cdots, 2n+2\} \\ j \in \{-2,-1,1,2\}}} \{h(n - 1, k - j)\} \qquad \text{(by Inequality 4)} \\
&\quad = \max_{k \in \{-2n, \cdots, 2n\}} \{f(n - 1, k)\}.
\end{aligned}$$

Therefore, the value of e^{ε_A} will be obtained at $n = 2$ for some k, i.e. $f(2, k)$. $f(2, k)$ achieves its maximum (minimum)

at $k = -3(4)$ with $X = (-1, -1, -1)$, $X' = (-1, -1, 1)$. Then, we have the following expression of ε_A:

$$e^{\varepsilon_A} = \begin{cases} \dfrac{2e^{\varepsilon_L}+1}{3}, & 0 < \varepsilon_L < 1 + 2\sqrt{2}, \\ \dfrac{e^{3\varepsilon_L}+3}{e^{2\varepsilon_L}+e^{\varepsilon_L}+2}, & \varepsilon_L \geq 1 + 2\sqrt{2}. \end{cases}$$

When ε_L is set in the common range, the difference between ε_A and ε_L does not converge to 0. □

V. Functionality

In this section, we explore the question of 'what functionalities can be computed in the aggregation model' and compare its capabilities with those of the shuffle model. Initially, the shuffle model appears to provide the analyzer with a richer set of functionalities, as it outputs a vector of randomized data, while the aggregation model only offers a single sum (with noise). However, upon closer examination, we carefully review the concrete computation tasks that existing shuffle protocols can perform and discover that all of these tasks are achievable in the aggregation model as well. This finding raises doubts about the speculation that the shuffle model is inherently more functionality-rich than the aggregation model.

From a theoretical perspective, literature [9] demonstrates that functionalities computable in the SQ model can also be privately computed in the shuffle model. As a comparison, we prove in section V-A, and all computation tasks in the SQ model can also be privately computed in the aggregation model. An astute reader might question why we chose the SQ model for comparison. The reason is simple; currently, no other computational model are known to capture the classes of functionalities that current shuffle protocols can achieve. Section V-B provides a few concrete examples showing aggregation protocols can do some complex tasks, hence justifying these counter-intuitive theoretical results.

A. Theoretical Results

In the SQ model, the algorithm learns by accessing statistical properties provided by the SQ oracle. To show that aggregation protocols can support all algorithms in the SQ model, it is sufficient to show that aggregation protocols can simulate the SQ oracle. That is, whatever the SQ oracle can do, the aggregation protocol can do as well, with a high probability. Thus, the SQ algorithms can query an aggregation protocol instead of the SQ Oracle and should produce the same quality output.

For any statistical query (g, τ), the SQ oracle can output an estimation for the expectation of g over the domain $\mathcal{X}$ that is accurate with additive error $\pm\tau$. Following [28], here we consider g to be a real-valued function $g : \mathcal{X} \to [-a, a]$ that is more general than Boolean. The global sensitivity of g is thus $2a$. We can construct an algorithm A_g, as shown in Figure 5, that simulates the SQ oracle using an aggregation protocol A_{sum}. Corollary 1 guarantees that A_g produces the same quality output of the SQ oracle with probability at least $1-\beta$, where β can be arbitrarily small given enough samples.

Corollary 1: Algorithm A_g approximates $E_{u\sim\mathcal{P}}[g(u)]$ within additive error $\pm\tau$ with probability at least $1-\beta$, if input database z has $n = \Omega(\frac{a^2 \log\frac{1}{\beta}}{\tau^2} + \frac{a\sqrt{\log\frac{1}{\beta}\log\frac{1}{\delta}}}{\tau\varepsilon})$ entries sampled i.i.d. from a distribution $\mathcal{P}$ on domain $\mathcal{X}$ for $A_{\text{sum}} = A^G_{n,a,\sigma}$; or $n = \Omega(\frac{a^2 \log\frac{1}{\beta}}{\tau^2} + \frac{a \log\frac{1}{\beta}}{\tau\varepsilon})$ for $A_{\text{sum}} = A^L_{n,a,b}$.

Algorithm $A_g(n, \varepsilon, \delta, g, A_{\text{sum}})$ that simulates an SQ Oracle

Input: $u_1, u_2, \cdots, u_n \in \mathcal{X}$, query $g : \mathcal{X} \to [-a, a]$, A_{sum} which can be a aggregation summation protocol.
Output: $\frac{1}{n} A_{\text{sum}}(g(u_1), \cdots, g(u_n))$.

Fig. 5. Algorithm A_g that simulates an SQ Oracle.

Proof: To prove the Corollary 1, we first recall the accuracy of the two aggregation protocols $A^G_{n,a,\sigma}$ and $A^L_{n,a,b}$ in Claim 1 and 2, separately. The proof can be found in Appendix D in our paper's full version [26].

Claim 1: For any $\varepsilon, \delta \in (0, 1]$, $n \in \mathbb{N}^+$, $X = (x_1, \cdots, x_n) \in [-a, a]^n$ and $0 < \beta < 1$, with probability at least $1-\beta$:

$$|A^G_{n,a,\sigma}(X) - \sum_{i=1}^n x_i| \leq \frac{4a}{\varepsilon}\sqrt{\log\frac{1}{\beta}\log\frac{1.25}{\delta}}.$$

Claim 2: For any $\varepsilon \in (0, 1)$, $n \in \mathbb{N}^+$, $X = (x_1, \cdots, x_n) \in [-a, a]^n$ and $0 < \beta < 1$, with probability $1-\beta$:

$$\left|A^L_{n,a,b}(X) - \sum_{i=1}^n x_i\right| \leq \frac{2a}{\varepsilon}\log\frac{1}{\beta}.$$

Let $v = E_{u\sim\mathcal{P}}[g(u)]$ denote the expectation of function g over the domain $\mathcal{X}$. Recalling the Hoeffding's inequality, for any $\beta_0 \in (0, 1)$:

$$\mathbb{P}\left[\left|\frac{1}{n}\sum_{i=1}^n g(u_i) - v\right| < 2a\sqrt{\frac{1}{2n}\log\frac{2}{\beta_0}}\right] > 1 - \beta_0.$$

Substituting $2a\sqrt{\frac{1}{2n}\log\frac{2}{\beta_0}}$ with $\frac{\tau}{2}$ and β_0 with $\frac{\beta}{2}$ in the above inequality, we have

$$\mathbb{P}\left[\left|\frac{1}{n}\sum_{i=1}^n g(u_i) - v\right| \geq \frac{\tau}{2}\right] \leq 2e^{-\frac{\tau^2 n}{8a^2}}.$$

Solving the equation $2e^{-\frac{\tau^2 n}{8a^2}} = \beta/2$, we obtain that without adding random noise, $\frac{8a^2\log\frac{4}{\beta}}{\tau^2}$ examples are enough to approximate $E_{u\sim\mathcal{P}}[g(u)]$ within additive error $\pm\frac{\tau}{2}$ with probability at least $1 - \frac{\beta}{2}$.

Recall the relationship of A_g and A_{sum} shown in Figure 5:

$$A_g(u_1, \cdots, u_n) = \frac{1}{n} A_{\text{sum}}(g(u_1), \cdots, g(u_n)).$$

In the case of $A_{\text{sum}} = A^G_{n,a,\sigma}$, by the result of Claim 1,

$$\left|\frac{A^G_{n,a,\sigma}}{n} - \frac{1}{n}\sum_{i=1}^n g(u_i)\right| \leq \frac{4a}{n\varepsilon}\sqrt{\log\frac{2}{\beta}\log\frac{1.25}{\delta}}.$$

Substituting $\frac{4a}{n\varepsilon}\sqrt{\log\frac{2}{\beta}\log\frac{1.25}{\delta}} = \frac{\tau}{2}$, we have $\frac{8a\sqrt{\log\frac{2}{\beta}\log\frac{1.25}{\delta}}}{\tau\varepsilon}$ samples are sufficient to ensure the noise added through gausian-based aggregation algorithm lies outside $\pm\frac{\tau}{2}$ with

probability at most $\frac{\beta}{2}$. Combining the above, we have that A_g estimates $E_{u\sim\mathcal{P}}[g(u)]$ within additive error $\pm\tau$ with probability at least $1-\beta$ if $n = \frac{8a^2\log\frac{4}{\beta}}{\tau^2} + \frac{8a\sqrt{\log\frac{2}{\beta}\log\frac{1.25}{\delta}}}{\tau\varepsilon}$.

When $A_{\text{sum}} = A^L_{n,a,b}$, with the same method, $\frac{8a^2\log\frac{4}{\beta}}{\tau^2} + \frac{4a\log\frac{2}{\beta}}{\tau\varepsilon}$ samples are enough to draw the result that A_g estimates $E_{u\sim\mathcal{P}}[g(u)]$ within additive error $\pm\tau$ with probability at least $1-\beta$. □

The proof of Corollary 1 is quite general and similar argument can apply to any aggregation protocols, except the number of samples required would change depending on the noise added in the sum. From Corollary 1, it follows directly that an SQ algorithm can be simulated by an aggregation algorithm. Furthermore, the simulation also preserves the differential privacy property of the underlying aggregation protocol.

Theorem 8: Let A_{SQ} be an SQ algorithm that makes at most t queries to an SQ oracle $SQ_{\mathcal{P}}$, each with tolerance at least τ. The simulation above is ε-differentially private (resp. (ε,δ)-differentially private) when A_g parameterized with $A_{\text{sum}} = A^L_{n,a,b}$ (resp. $A^G_{n,a,\sigma}$). If dataset X has $n' = tn = \Omega(\frac{ta^2\log\frac{t}{\beta}}{\tau^2} + \frac{ta\log\frac{t}{\beta}}{\varepsilon\tau})$ (resp. $\Omega(\frac{ta^2\log\frac{t}{\beta}}{\tau^2} + \frac{ta\sqrt{\log\frac{t}{\beta}\log\frac{1}{\delta}}}{\tau\varepsilon})$) entries sampled i.i.d. from the distribution $\mathcal{P}$, then the simulation above gives the same output as A_{SQ} with probability at least $1-\beta$.

Proof: On the aspect of privacy, it provides ε-differential privacy (resp. (ε,δ)-differential privacy) because each piece of data in X is independent. On the aspect of probability of failure, the SQ algorithm queries an SQ oracle $SQ_{\mathcal{P}}$ at most t times, and the aggregation algorithm simulates each query (g,τ) by running A_g on n samples. The allowed probability of failure for each query is $\beta' = \frac{\beta}{t}$. By the union bound, the probability of any of the queries not being approximated within additive error $\pm\tau$ is bounded by β. □

B. Concrete Examples

The crux of the discrepancy between the theoretical results and the intuition is that although the aggregator outputs a single sum in each run of the aggregation protocol, the functionality can be decomposed, and the analyzer in the aggregation model can obtain a vector of data values through multiple (possibly parallel) runs of the aggregation protocol. Hence, the analyzer can compute any SQ algorithm in the aggregation model. As concrete examples, in the following, we show how to obtain histograms, compute sample variance, and optimize using the Stochastic Gradient Descent algorithm.

We start by introducing a vector aggregation protocol A_H (See Figure 6), which is essentially composed of multiple instances of the aggregation protocol A_{sum} that privately sums scalar values. A_H satisfies (ε,δ)-differential privacy if the aggregation protocol A_{sum} satisfies (ε,δ)-differential privacy and the elements in the vector are independent (Proof is available in Appendix A of the full version [26] of our paper). A_H satisfies (ε,δ)-differential privacy if A_{sum} satisfies $(\frac{\varepsilon}{d},\frac{\delta}{d})$-differential privacy and d out of k dimensions in the vector are dependent (by the sequential composition theorem of differential privacy [1]).

Vector aggregation protocol A_H

Randomizer: $R(X)$:
Input: $X = (x_1,\cdots,x_k) \in \mathcal{X}^k$,
Output: $Y = (y_1,\cdots,y_k) \in \mathbb{R}^k$

1) For $1 \le i \le k$, randomize each x_i by using the randomizer in A_{sum}, and produce y_i=$A_{\text{sum}}.R(x_i)$
2) Return $Y = (y_1,\cdots,y_k)$

Aggregator $G(Y_1,\cdots,Y_n)$:
Input: $Y_1,\cdots,Y_n \in \mathbb{R}^k$
Output: $Z = (z_1,,\cdots,z_k) \in \mathbb{R}^k$

1) For each $1 \le i \le k$, sum the i-th value in vectors Y_1 to Y_n, and set $z_i = A_{\text{sum}}.G(Y_{1,i},\cdots,Y_{n,i})$;
2) Return $Z = (z_1,,\cdots,z_k)$;

Analyzer receives Z;

Fig. 6. Protocol: A_H.

The following examples all use A_H. At a high level, each user uses a local encoding algorithm to encode his/her record into a vector and runs A_H, then the aggregator outputs a perturbed aggregated vector, and the analyzer can use the vector as the input to an estimation algorithm (depending on the randomizer) to compute the desired statistics.

1) Histogram: For n users each hold a record, we show how to privately generate a histogram through aggregation. Let $Q:\mathcal{X}\to\mathcal{Z}^k$ be a histogram query that partitions the data values into k bins. For convenience, we also define predicates $q_1,\ldots,q_k$ such that $q_i:\mathcal{X}\to\{0,1\}$ evaluates to 1 if the data value falls into the i-th bin, and 0 otherwise. To generate a histogram, each user encodes its record x as a vector $U = (u_1,\cdots,u_k) = (q_1(x),\cdots,q_k(x))$. It is clear that U is a standard basis vector whose elements are all 0, except one that equals 1. Then, all the users run the vector aggregation protocol A_H that aggregates their vectors. The sum of those n standard basis vectors gives the histogram, and the histogram is protected by the noise added by the randomizers.

2) Sample Variance: For n users such that each user i holds a real value $x_i \in \mathbb{R}$, the sample variance is defined as $S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2 = \frac{1}{n-1}(\sum_{i=1}^{n}x_i^2 - n\bar{x}^2)$, where the sample mean $\bar{x} = \frac{1}{n}\sum_{i=1}^{n}x_i$. To do so, each user encodes its record x_i as a vector $U_i = (u_{i,1},u_{i,2}) = (x_i^2,x_i)$ and uses U_i in the vector aggregation protocol A_H. A_H outputs $Y = (y_1,y_2)$ to the analyzer, which outputs $z = \frac{1}{n-1}(y_1 - \frac{1}{n}y_2^2)$.

3) Stochastic Gradient Descent: Stochastic gradient descent (SGD) is popular in machine learning and is one of the most fundamental components in Neural Networks. It is an iterative approach that can be used to learn linear classifiers and regressors. We here describe how to implement mini-batch SGD, the most common form of SGD, in the aggregation model. Without the loss of generality, there are n users each has a labeled example (x,l), where record $x \in \mathbb{R}^d$, and

label $l \in \{-1, 1\}$. The analyzer begins with an initial vector $w_0 \in \mathbb{R}^d$. At step t, it randomly chooses b users and sends w_t to them, where w_t is the vector computed from w_0 after $t-1$ times update. Each of these users computes (sub)gradient $U = \nabla(w_t, x, l)$ and sends this d-dimension vector to A_H, which outputs $Y = (y_1, \cdots, y_d)$ to the analyzer. Finally, the analyzer updates $w_{t+1} = w_t - \eta_t(\lambda w_t + \frac{1}{b}Y)$, where η_t and λ are some fixed learning algorithm parameters.

Beyond these examples, in the aggregation model, the analyzer can compute various statistics based on the output of aggregation protocols. For example, with the sum, mean can be easily computed. Also, with histograms, median or most frequent items can be computed. More complex functions, e.g., k-means, can be computed by iteratively calling the aggregation protocols. In principle, since all SQ queries can be answered in the aggregation model, the aggregation protocols can be used to compute a fairly wide range of functions, including complex ones like expectation-maximization, SVM, linear/convex optimization, MCMC, simulated annealing, and so on [29]. Real-world private analytics, such as what Apple [3] and Google [2] do, can all be computed in the aggregation model.

VI. Accuracy Analysis

In this section, we analyze and compare the accuracy of concrete protocols in both the shuffle and aggregation models. We begin with the summation task, evaluating it from both theoretical and empirical perspectives. Our analyses from both angles consistently demonstrate that the aggregation protocols have better accuracy. Subsequently, we delve into more complex tasks, including histogram, top-k, sorting, SGD, and PCA. The empirical evaluation reveals that aggregation protocols consistently outperform shuffle protocols in those statistical analyses. This experimental validation also demonstrates the usefulness of the aggregation protocols, as the iterative use of summation can achieve acceptable utility levels even for complex tasks.

A. Theoretical Analysis on Summation Task

We measure the accuracy of protocols using two metrics: mean square error (MSE) and (α, β)-accuracy. Both metrics are commonly used in the analysis of shuffle protocols. The MSE quantifies the average noise introduced during protocol execution. On the other hand, (α, β)-accuracy bounds the worst-case noise added, guaranteeing that it remains below a threshold α with a probability of at least $1-\beta$.

We focus on the summation protocols, which calculate the sum of binary or real-valued data. We compare seven shuffle protocols[1] and two aggregation protocols ($A^G_{n,a,\sigma}$ and $A^L_{n,a,b}$),[2] and the results are presented in Table I.

Table I highlights the Laplace aggregation protocol $A^L_{n,a,b}$ as the most accurate among all protocols with the same privacy guarantee. Following closely is the shuffle protocol [30]-real-2. These two protocols stand out because they distributively add Laplace noise and discrete Laplace noise, respectively, which, as it turns out, are the most effective noise distributions for the DP summation task.

TABLE I
Accuracy Comparison of Shuffle and Aggregation Protocols

Protocols	MSE	α
[9]-bit	$O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$	$\frac{30}{\varepsilon}\sqrt{\log\frac{4}{\delta}\log\frac{2}{\beta}}$*
[9]-real	$O(\frac{1}{\varepsilon^2}\log^2\frac{n}{\delta})$	$\frac{122}{\varepsilon}\log\frac{8}{\delta}\sqrt{\log\frac{4}{\beta}}$*
[8]-real	$O(n^{\frac{1}{3}}\frac{\log^{\frac{2}{3}}\frac{1}{\delta}}{\varepsilon^{\frac{4}{3}}})$*	$\frac{2n^{\frac{1}{6}}}{\varepsilon^{\frac{2}{3}}}\log^{\frac{1}{3}}\frac{2}{\delta}\sqrt{19\log\frac{2}{\beta}}$*
[27]-bit	–	$\frac{50}{\varepsilon^2 n}\log\frac{2}{\delta}+\frac{\sqrt{200}}{\varepsilon n}\sqrt{\log\frac{2}{\delta}\log\frac{2}{\beta}}$
[30]-real-1	$O(\frac{(\log\log n)^2}{\varepsilon^2}\log\frac{1}{\delta})$	–
[30]-real-2	$O(\frac{1}{\varepsilon^2})$	$\sqrt{\frac{1}{\beta}\left(\frac{2}{\varepsilon^2}+\frac{1}{4}+5n^2e^{-\frac{\varepsilon n}{2}}\right)}$
[31]-real	$O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$	$\sqrt{\frac{1}{\beta}\left(\frac{1}{300n}+\frac{2000}{\varepsilon^2}\log\frac{1}{\delta}\right)}$*
$A^G_{n,a,\sigma}$	$O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$	$\frac{2}{\varepsilon}\sqrt{\log\frac{1.25}{\delta}\log\frac{1}{\beta}}$*
$A^L_{n,a,b}$	$O(\frac{1}{\varepsilon^2})$	$\frac{1}{\varepsilon}\log\frac{1}{\beta}$*

The deeper and more intuitive reason why aggregation protocols outperform shuffle protocols can be attributed to the noise addition paradigm. From the central view, all nine summation protocols follow the paradigm of outputting the true summation plus a noise random variable. The Gaussian and Laplacian mechanisms have already demonstrated their utility in the central model and can be efficiently implemented in the aggregation model. Compared to other methods of adding randomness in the shuffle model, it is no surprise that the aggregation protocols perform better.

We also observed that multiple-message shuffle protocols exhibit higher accuracy compared to single-message shuffle protocols. This advantage stems from the flexibility offered by multiple-message protocols in their design. Single-message protocols [9]-bit, [9]-real, [8]-real employ a technique known as the "privacy blanket" [8], where some individual records are randomly replaced with noise. Multi-message protocols [27]-bit and [30]-real-1 extend the "privacy blanket" technique to the multi-message case. Notably, [30]-real-1 adopts a recursive approach, leveraging the single-message protocol to bootstrap the privacy amplification it can achieve. As a result, the multi-message version adds less noise while ensuring the same privacy guarantee, leading to improved utility.

On the other hand, multi-message protocols such as [31]-real and [30]-real-2 employ a different technique. They utilize an analog of secret sharing in the distributional DP setting, splitting each individual record into a set of random look-like messages. This technique is exclusive to the multi-message model, allowing for more effective noise management and further enhancing accuracy.

B. Experimental Validation on Summation Task

In this subsection, we present the results of empirical experiments conducted on both aggregation and shuffle summation protocols. Our experiments take into account three factors that

[1]While some of the shuffle protocols' accuracy is analyzed using one metric in the original paper, we also compute the other metric (marked with "*" in Table I) whenever possible. The detailed calculations are provided in Appendix D and E of the full version [26] of our paper.

[2]$A^L_{n,a,b}$ satisfies ε-differential privacy, and others protocols satisfy (ε, δ)-differential privacy.

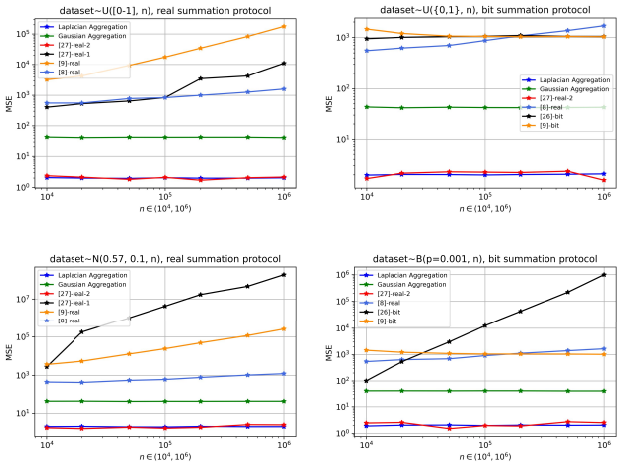


Fig. 7. MSE under different n values and dataset.

can influence performance: the differential privacy parameters, data distribution, and dataset size. For the Laplace aggregation protocol, we set $\epsilon = 1$, while for other protocols, we fixed the privacy parameters at $\epsilon = 1$ and $\delta = 2^{-30}$. The input dataset size varied from 10,000 to 1,000,000.

Regarding data distribution, we considered two options for real summation protocols: uniformly chosen inputs from the real domain [0, 1], and inputs following a normal distribution with mean 0.57 and standard deviation 0.1. For bit summation protocols, we explored two dataset distributions: uniformly chosen inputs from the binary domain 0, 1, and inputs following a Bernoulli distribution with a probability of 0.001 being 1 and 0 otherwise.

Figure 7 presents the results, with each point representing the empirical MSE of the respective protocol, averaged from 1000 protocol executions. Notably, the MSE of both the Laplace aggregation protocol and the multi-message shuffle protocol [30]-real-2 is significantly lower, by orders of magnitude, compared to that of other protocols. As discussed in Section VI-A, these two protocols add (almost) the same noise distribution from the central view, resulting in comparable and better accuracy compared to other protocols.

Our second observation is the consistency of accuracy for the two aggregation protocols and the shuffle protocol [30]-real-2 across different dataset sizes (n) and dataset distributions. In contrast, the accuracy of other shuffle protocols is dependent on these factors, validating the theoretical results discussed earlier. This observation further highlights the potential for more stable and reliable performance from aggregation protocols.

We also observe multi-message shuffle protocol [30]-real-1 and [27]-bit don't always outperform single-message shuffle protocols, as shown in Figure 7 (bottom figures). We first look at the protocol [27]-bit in the bottom left figure. The dataset used here only has a small fraction of 1, and the rest is 0. When the sum x is much smaller than the dataset size n, protocol [27]-bit outputs 0 as the estimate of x with high probability. This probability increases as n increases and x decreases. Consequently, when we set $x = \sqrt{n}$ and n sufficiently large, the MSE for [27]-bit becomes $O(n)$. Notably, the MSE of [8]-real is $O(n^{1/3})$, and [9]-bit does not depend on n. This difference reveals that the distribution of the added noise in [27]-bit can vary depending on the data distribution, and that is one reason why, in some cases, the accuracy can be worse than single-message shuffle protocols.

Figure 7 (bottom right figure) shows that the accuracy of the multi-message protocol [30]-real-1 is worse than that of its single-message version [8]-real. The errors in both [8]-real and [30]-real-1 arise from the noise for privacy guarantee and the rounding error used to convert real data into integer data. The multi-message protocol incurs more rounding errors than the single-message one, as each data piece is split into multiple messages, each requiring rounding. Consequently, the overall rounding error is larger for the multi-message protocol. In scenarios where the rounding error dominates the overall error, the utility of the multi-message protocol is worse than that of the single-message protocol.

C. Experimental Validation on Complex Computation Tasks

In this subsection, we evaluate and compare the performance of aggregation and shuffle protocols designed for a diverse set of computation tasks, including histogram, top-k, sorting, Stochastic Gradient Descent (SGD), and Principal Component Analysis (PCA). The results show that the aggregation protocol provides useful results for each specific task. The utility of the aggregation protocols is satisfactory even for SGD with multiple iterations. Moreover, in comparison with their corresponding shuffle protocols, the aggregation protocols consistently demonstrate superior performance. We present the problem settings, utility metrics, and detailed protocol performance results for each task in the following.

1) Histogram, Top-K, and Sorting: The evaluated aggregation histogram protocols are instantiations of protocol A_H (Fig. 6). The summation protocol A_{sum} within A_H is instantiated with $A^L_{n,a,b}$ and $A^G_{n,a,\sigma}$, respectively. Similarly, the shuffle histogram protocols are instantiations of the histogram protocol presented in [27] (Fig. 2), using the summation protocols [9]-bit and [8]-real. We conducted these protocols on the Fire dataset [32], which contains 681,174 user calls to the San Francisco Fire Department, classified into 272 "Alert" types. We use privacy parameters $\varepsilon = 0.05, 0.1, 0.5, 1$ and $\delta = 10^{-5}$. The histogram query results in 272 noised counters, each representing an "Alert" type. To evaluate the error of the query results, we measure the empirical MSE of the obtained counters, averaged from 100 protocol executions.

To obtain the top-k and sorting query results, we perform post-processing on the histogram query results. For the top-k query, we extract the first k largest counters and return the corresponding "Alert" types associated with them. In the case of the sorting query, we sort the top-k counters in descending order and retrieve the corresponding "Alert" types accordingly. To evaluate the error of these query results, we count the number of types that are in the wrong position in the returned "Alert" types list. The average accuracy is computed from 100 protocol executions. In experiments, we set k to 20.

Table II presents a comparison of errors between aggregation protocols and shuffle protocols for the histogram, top-k, and sorting tasks. The results show that the error of

TABLE II
ACCURACY COMPARISON OF PROTOCOLS IN THE SHUFFLE MODEL AND THE AGGREGATION MODEL

Protocols	$\varepsilon = 0.05$			$\varepsilon = 0.1$			$\varepsilon = 0.5$			$\varepsilon = 1$		
	MSE	top k	sorting	MSE	top k	sorting	MSE	top k	sorting	MSE	top k	sorting
[9]-bit	3.02e6	2.55	12.17	6.66e5	1.43	1.06e4	0.06	0	2.30	2.57e3	0	1.35
[8]-real	7.14e6	3.52	14.23	1.63e5	0.64	7.21	7.59e3	0.02	2.13	2.54e3	0	1.34
Laplace	3.71e3	0.01	1.19	7.91e2	0	0.83	25.31	0	0.24	8.62	0	0.02
Gaussian	4.70e4	0.18	4.95	1.06e4	0.03	2.22	401.32	0	0.76	107.55	0	0.50

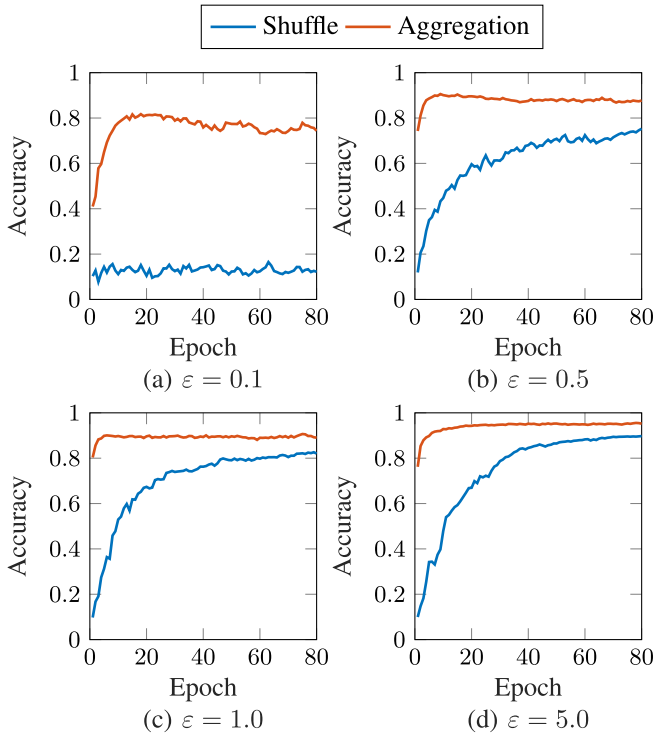


Fig. 8. Accuracy of SGD for aggregation and shuffle protocols.

aggregation protocols is consistently lower than that of the shuffle protocols.

2) Stochastic Gradient Descent (SGD): The aggregation SGD protocol is an instantiation of the DP SGD scheme of Abidi et al. [33] in the aggregation model, and the shuffle SGD protocol is from the work of Girgis et al. [34]. We conducted these protocols on the MNIST dataset [35] and evaluated their utility by computing the prediction accuracy rate on the testing set. For each scheme, training was performed for 80 epochs, using different privacy parameters, namely $\varepsilon = 0.1, 0.5, 1, 5$, and $\delta = 10^{-5}$.

Figure 8 demonstrates that the aggregation SGD protocol consistently outperforms the shuffle protocol. The upper left figure shows that even for high privacy settings ($\varepsilon = 0.1$), the aggregation protocol can still achieve an impressive accuracy rate of up to 75%. In contrast, the shuffle protocol's accuracy is around 10%, equivalent to random guessing. These results serve as a compelling example of the effectiveness of the aggregation model in handling complex tasks that require iteratively publishing sums.

3) Principal Component Analysis (PCA): The aggregation PCA protocol is a distributed implementation of the central PCA algorithm [36], both of which perform the same operations. Therefore, the accuracy of the aggregation PCA is the same as that of the central algorithm [36]. We defer the introduction of the Principal Component Analysis and the comparison between the central PCA algorithm and the aggregation PCA protocol to Appendix F of the full version [26] of our paper.

In the experiment, we run the aggregation PCA protocol on 10,000 samples taken from the MNIST dataset [35]. We use privacy parameters $\varepsilon = 0.1, 0.5, 1.0, 5.0$ and $delta = 10^{-5}$. To evaluate the error of the query result, we calculate the l_2 distance between the k normalized raw singular vectors V_k and the noised singular vectors $\hat{V}$ with the largest singular values, denoted as $\|V_k V_k^T - \hat{V}_k \hat{V}_k^T\|_2$. For this experiment, we set k to 10.

As a shuffle PCA protocol was not available for comparison, we report the errors of the aggregation PCA protocol at different privacy levels: 0.7436 at $\varepsilon = 0.1$, 0.1863 at $\varepsilon = 0.5$, 0.0543 at $\varepsilon = 1.0$, and 0.0022 at $\varepsilon = 5.0$.

Remark: The utility of the aggregation protocols depends largely on the utility of the corresponding central differential privacy protocols, especially in scenarios where the data needs to be queried multiple times. We emphasize that the DP research community has devoted considerable effort to addressing the issue of noise accumulation and its potential impact on utility over the years. Leveraging existing advancements, such as the improved privacy accounting method [33] used in DP SGD, and noise reduction techniques for counting queries [37], [38], [39], the aggregation protocols can effectively mitigate the impact of noise accumulation.

VII. PRACTICALITY ANALYSIS

A. Minimal Number of Users

One factor that can restrict the application of the shuffle protocols in real-world scenarios is that they often require a large number of users to participate, in order to achieve adequate privacy. For instance, in [9], the bit sum protocol can only be proved to be (ε, δ)-differentially private under the constraints that $\varepsilon \in (\frac{\sqrt{3456}}{n} \log \frac{4}{\delta}, 1)$ and $n \geq 14 \log \frac{4}{\delta}$. Therefore, given a particular (ε, δ) pair, n is lower bounded by both ϵ and δ. The real sum protocol in [9] also requires a minimal n because this protocol is essentially realized by invoking the bit sum protocol multiple times. The lower bound of n is much worse than that of the bit sum protocol because

TABLE III
MINIMUM NUMBER OF USERS REQUIRED FOR PROTOCOLS IN THE SHUFFLE MODEL

ε	$min(n)$		
	[9]-bit	**[9]-real**	**[8]-real**
0.01	130396	1791262	6016518
0.1	13040	179127	60166
0.2	6520	89564	15042
0.5	4259	35826	2407
1.0	4259	17913	602

to achieve a certain (ε, δ), the base bit sum protocol being invoked has to satisfy smaller privacy parameters $(\varepsilon_0, \delta_0)$. Similarly, in [8], the real sum protocol has a constraint $\frac{14(k+1)\log(2/\delta)}{n\varepsilon^2} < 1$ that lower bounds n. in Table III, we show the minimal n calculated under various ε for those protocols in the shuffle model, when δ is fixed to 2^{-30}. As we can see, better privacy generally requires more users. In contrast, user numbers in aggregation protocols generally are not a concern. For example, the two aggregation protocols in Section III-C can have an arbitrary number of users, as few as 1, for any (ε, δ).

B. Efficiency Analysis

In the previous sections, we treated the shuffler and the aggregator as ideal functionalities. In the real world, there are no such ideal functionalities, and they have to be implemented somehow. This brings on the question of which model is more efficient in reality. In this section, we try to answer this question.

In either model, the shuffler or the aggregator is assumed to be untrusted. This is because if there is a trusted party, one can be better off by using the trusted party to realize a central mechanism for differential privacy. To ensure correctness and security when utilizing an untrusted shuffler or aggregator, some technical measures are inevitably needed. In this section, we show the results obtained via two different routes: by using a cryptographic protocol and by using trusted hardware.

1) Cryptographic Protocol: A shuffler can be realized via a mixnet. A mixnet [40] is a protocol involving a sequence of untrusted nodes. The first node takes as input a set of encrypted messages and outputs a uniformly random permutation of those messages (after re-encryption/randomization). The first node's output is taken by the second node as input, which will permute the messages again. As long as there is one honest node, the messages will be shuffled randomly in this process. To ensure that the nodes cannot manipulate the messages, each node also produces a cryptographic proof to show that the plaintexts of messages in the output set are the same as those of the input set.

An aggregator can be realized via Multiparty Computation (MPC). Specifically, the protocol involves several untrusted nodes as computation parties. The users send their inputs to the computation parties in an encrypted form, and then the computation parties compute the sum of the data. It is easy to use a generic MPC framework such as SPDZ [41], [42] to implement the aggregator, and SPDZ guarantees that as long as there is one honest computation party, the sum can be computed securely and correctly.

a) A remark: There are three different flavors of aggregation protocols in the literature: (a) the users use MPC and interact among themselves, without intermediate parties, to realize a virtual aggregator that computes the sum [19]; (b) the aggregator is a single physical node, and computes the sum by running a cryptographic protocol with the users [17], [18], [20]; (c) the aggregator is a group of nodes, and compute the sum by running a cryptographic protocol with the users [43]. Here we adopt (c) in the comparison because the shuffler has to be made of at least two nodes – a virtual shuffler run by all users is not practically feasible, and a single node shuffler means we have to trust the shuffler to shuffle properly. Otherwise, there is no guarantee that the permutation is random. For this reason, if the aggregator is by approach (a) or (b), then the comparison is not fair because of the difference in the trust assumptions.

b) Complexity analysis: We first compare the computational and communication complexity for protocols realizing the shuffler and the aggregator. The shuffler protocol is based on the state-of-the-art verifiable shuffle protocol [44], and the aggregator protocol is based on the SPDZ framework (the framework can be found in Appendix H of the full version [26] of our paper). The results can be found in Table IV. Both protocols can be divided into an offline and online phase such that the offline phase is used for pre-processing and the online phase is used for the actual computation. In particular, in the offline phase of the aggregator protocol, the computation parties generate secret shares of random numbers, while in the offline phase of the shuffler protocol, the mixnet nodes generate a common reference string (CRS).

For the computational complexity, in Table IV, we count the number of most computationally costly operations. The shuffler protocol relies heavily on public key operations, i.e., group exponentiation (scalar multiplication in an Elliptic Curve group) and pairing. The aggregator protocol in the online phase involves only modular addition in a small field. Although in the online phase, the computational costs of both protocols are linear in the number of nodes and users, the operations in the aggregator protocols are much cheaper (e.g., see Table V). In the offline phase, the aggregator protocol requires somewhat homomorphic encryption whose computation is dominated by multiplications in a polynomial ring. Note that although the polynomial multiplication is a more costly operation, the aggregator can benefit from the SIMD parallelization of the underlying homomorphic encryption scheme, and reduce the number of operations by a large factor $\phi(M)$, which is often in the order of thousand. Therefore, the aggregator protocol is more efficient overall.

For the communication complexity, the messages in the shuffler protocol consist of elements in two elliptic curve groups $\mathbb{G}_1$ and $\mathbb{G}_2$, and the messages in the aggregator protocol consist of elements in finite field $\mathbb{F}_q$ and $\mathbb{F}_p$ $(q > p)$. Usually, elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ have to be large enough to be secure, e.g. ∼256-bit (with point compression) to achieve 128-bit security. The size of $\mathbb{F}_q$ and $\mathbb{F}_q$ can be much smaller

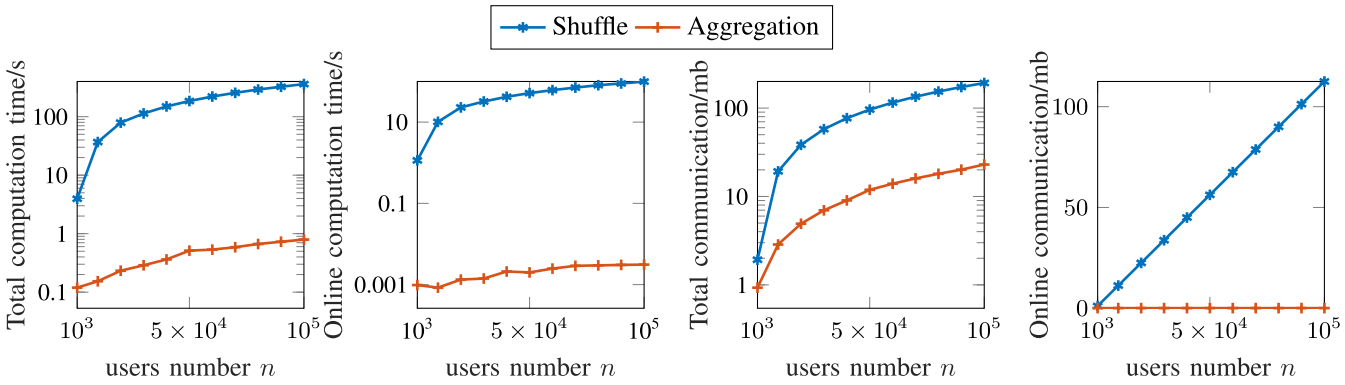


Fig. 9. The computation and communication overhead of the shuffler protocol and the aggregator protocol.

TABLE IV

EFFICIENCY COMPARISON (k: THE NUMBER OF THE MIXNET NODES/COMPUTATION PARTIES; n: THE NUMBER OF USERS)

Efficiency		Shuffler	Aggregator
Comp.	Online	$13kn$ exp., kn pair.	kn add.
	Offline	$4kn$ exp., $2kn$ pair.	$10kn/\phi(M)$ polymul.
Comm.	Online	$4kn \times \mathbb{G}_1$, $3kn \times \mathbb{G}_2$	$k^2\mathbb{F}_p$
	Offline	$4kn \times \mathbb{G}_1$, $kn \times \mathbb{G}_2$	$\frac{kn}{\phi(M)}(M\mathbb{F}_q + 3\phi(M)\mathbb{F}_p)$

TABLE V

COST COMPARISON OF PRIMITIVE OPERATIONS AND ELEMENTS IN THE UNDERLYING GROUPS/FIELDS

Computation Time	n	1000	10000
	exp.	0.10s	1.03s
	pair.	0.54s	5.27s
	add.	25.1 ns	222.3 ns
	polymul.	0.55s	5.84s
Size	$\mathbb{G}_1$	254 bits	
	$\mathbb{G}_2$	254 bits	
	$\mathbb{F}_p$	70 bits	
	$\mathbb{F}_q$	249 bits	

depending on the plaintext domain (e.g., see Table V). Note that in the online phase, the communication complexity of the aggregator protocol is k^2, which is due to each computation party broadcasting one message. Here, k is often a small number compared to n (a few vs thousands).

c) Experimental evaluation: We also implemented the shuffler and the aggregator protocols in C++ and measured the performance based on our implementation. The implementation of the shuffler protocol is based on the source code[3] provide by the authors of [44], which uses libff[4] library for the underlying ECC and pairing operations. The particular curve used is BN-128, a Barreto-Naehrig curve that provides 128 bits of security. The aggregator protocol[5] is implemented on top of the SPDZ-2[6] library, in which it implemented the BGV somewhat homomorphic encryption [45]. The parameter of BGV was set to $|p| = 70$-bit, $|q| = 249$-bit, and $M = 8192$ to achieve 128 bits security. In the experiment, we employed two mixnet nodes for the shuffler and two computation parties for the aggregator, all of which have the same hardware (an Intel Core i7-7700 3.60GHz CPU and 16GB RAM). Note that here, we only used two nodes for each protocol because, in the aggregator protocol, the summation is done in parallel at all computation parties non-interactively, while the execution of the shuffler protocol is sequential, one node after another. Therefore, the difference would be more significant if more nodes were employed in the experiment.

Table V shows the computational cost of primitive operations (total time for 1,000 and 10,000 operations) in the two protocols as well as the size of the elements in the underlying groups and fields. The figures can be used in conjunction with those in Table IV to understand the actual cost of the protocols.

In Figure 9, we show the running time and communication cost of the shuffler and aggregator protocols, with different numbers of users ranging from 1,000 to 100,000. In Figure 9(a), we show the total computation time in seconds for both protocols and in Figure 9(b), we show the online computation time. As we can see, the difference is about 2 - 3 orders of magnitude. We can also see that the online computation phase in the aggregation model is very fast. This is because aggregation only involves the addition operation, and the addition operation in SPDZ is just an addition operation in some small fields, which is very fast. On the other hand, the shuffler protocols involve public key operations and thus are much slower. We also show the communication cost of the aggregator and shuffler protocols in Figure 9(c), Figure 9(d). Figure 9(c) shows the total communication cost. We can see the aggregator protocol uses much less bandwidth than the shuffler protocol. Figure 9(d) shows the online communication cost, from which we can see that the cost is linear to the number of users in the shuffler protocol but is constant in the aggregator protocol. It is easy to understand: in the shuffler protocol, one node has to pass the whole shuffled set to the other node, while the aggregators can perform the computation locally (since it is just an addition) and only need to send out the shares of one value that is the final sum.

2) Protocol Based on Trusted Hardware: We also implemented a shuffler and an aggregator based on Intel SGX.

[3]https://bitbucket.org/JannoSiim/hat_shuffle_implementation/src/master/
[4]https://github.com/scipr-lab/libff
[5]https://github.com/PuzzleEAA/eaa
[6]https://github.com/bristolcrypto/SPDZ-2

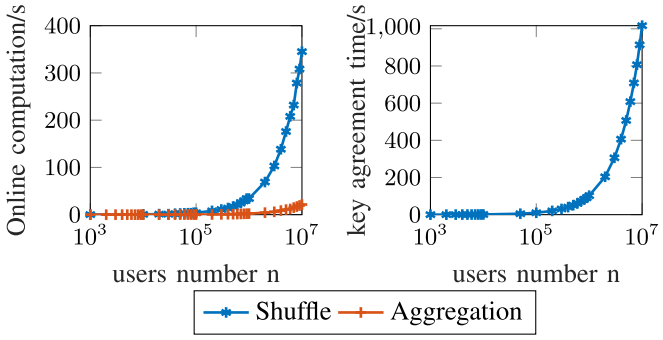


Fig. 10. The computation overhead of the SGX-based implementation.

The protocols are simple: the shuffler and the aggregator are programs running in SGX protected memory space, and the users communicate with the shuffler/aggregator through an authenticated secure channel and send in the randomized data, which is stored and processed (shuffle/aggregate) in SGX protected memory space. For the shuffler, we use the implementation of the stash shuffle[7] [5]. We run the experiments on a PC with Intel Core i7-7700 CPU and 3.60GHz, 16GB RAM. Figure 10(a) compares the total running time of shuffle and aggregation operation. In Figure 10(a), we can see that the running time of the aggregator protocol is significantly less than that of the shuffler, mainly because shuffling is a more costly operation. The stash shuffle requires $2n$ hash operations, while the aggregation requires only n addition operations. Note that when using SGX-based protocol, the users have to establish an authenticated secure channel with the shuffler/aggregator and thus need to run a key agreement protocol. This key establish phase is the same in both shuffler/aggregator protocols and is actually quite expensive (Figure 10(b)). If taking this into account, the cryptographic protocols actually need less time than SGX-based protocols when the number of users is large.

VIII. Related Work

The research on the shuffle model aims to improve local DP with better utility. Bittau et al. [5] first proposed an architecture called ESA (Encode, Shuffle, Analyze) for online monitoring tasks, but without rigorous analysis. Then, Erlingsson et al. [10] provided a privacy amplification bound of the shuffle model, quantifying the privacy of protocols in the shuffle model in terms of the local differential privacy provided by the local randomizer. The work by Cheu et al. [9] gave a protocol for the summation of bits, which can be extended to the real-valued case with an additional cost in communication. They showed that shuffle protocols provide strictly better accuracy than local protocols in some cases. Balle et al. [8] proposed a protocol for real number summation with better accuracy and communication cost than the protocol in [9]. In addition, it gave a new privacy amplification bound, generalizing the results in [10] to a wider range of parameters.

In the literature of private aggregation, there has been a lot of work on distributed realization of different privacy, to eliminate the requirement of trusted data collectors. This line of work follows a similar model, where users perturb their data locally and upload encrypted noisy data to the untrusted aggregator, such that the final decrypted result satisfies differential privacy. In 2006, Dwork et al. [19] first proposed a distributed implementation of privacy-preserving statistical databases, where the users generate Gaussian or exponential noise to make the database queries differentially private. Later, Shi et al. [20] proposed a private aggregation protocol, where the users distributively add geometrical noise to the sum. Chan et al. [18] proposed an approach, which is resilient to user failure and compromise. Differential privacy can be guaranteed even when some users are disconnected, at the cost of higher communication overhead and estimation error. Moreover, Ács and Castelluccia [17] proposed a protocol that realizes distributed Laplace mechanism for differential privacy. Eigner et al. [43] designed a generic architecture for distributed private aggregation, which supports the Laplace mechanism, Discrete Laplace, and Exponential mechanism.

[7]https://github.com/google/prochlo/tree/master/prochlo_stash_shuffler

IX. Conclusion and Future Work

In this paper, we conducted the first comparative study between the shuffle model and the aggregation model, both of which can achieve distributed differential privacy. Firstly, it demonstrates that the aggregation model, in contrast to the (single message) shuffle model, can provide ε-DP amplification. Secondly, it showcases that the aggregation model supports a wide range of computation tasks, including those supported by existing shuffle protocols. Furthermore, it compares the accuracy and efficiency of aggregation and shuffle protocols for various computation tasks from both theoretical and empirical perspectives.

Our analysis reveals that protocols in the aggregation model, despite being considered old fashioned, often outperform the newer protocols in the shuffle model in many aspects, and perhaps are more suitable for practical use in the current state. This observation prompts a research question for the distributed DP community: Can we design a shuffle protocol that outperforms the aggregation protocol, especially for computation tasks that extend beyond simple aggregations? To fully understand the strengths and limitations of both the shuffle and aggregation models in theory and practice, we believe further research is needed.

Acknowledgment

Yu Wei, Jingyu Jia, and Yuduo Wu are co-supervised by Prof. Changyu Dong and Zheli Liu and have the same contribution to this paper.

References

[1] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2014.

[2] Ú. Erlingsson, V. Pihur, and A. Korolova, "RAPPOR: Randomized aggregatable privacy-preserving ordinal response," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2014, pp. 1054–1067.

[3] A. D. P. Team, "Learning with privacy at scale," *Mach. Learn. J.*, vol. 1, no. 8, pp. 1–25, 2017.

[4] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting telemetry data privately," in *Proc. NIPS*, 2017, pp. 3571–3580.

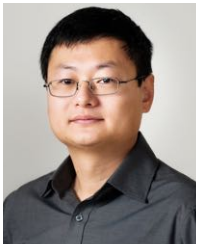
[5] A. Bittau et al., "PROCHLO: Strong privacy for analytics in the crowd," in *Proc. SOSP*, 2017, pp. 441–459.
[6] B. Avent, A. Korolova, D. Zeber, T. Hovden, and B. Livshits, "BLENDER: Enabling local search with a hybrid differential privacy model," in *Proc. USENIX Secur. Symp.*, 2017, pp. 747–764.
[7] A. Beimel, A. Korolova, K. Nissim, O. Sheffet, and U. Stemmer, "The power of synergy in differential privacy: Combining a small curator with local randomizers," in *Proc. ITC*, vol. 163, 2020, pp. 1–25.
[8] B. Balle, J. Bell, A. Gascon, and K. Nissim, "The privacy blanket of the shuffle model," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 11693. Berlin, Germany: Springer, 2019, pp. 638–667.
[9] A. Cheu, A. D. Smith, J. Ullman, D. Zeber, and M. Zhilyaev, "Distributed differential privacy via shuffling," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 11476. Berlin, Germany: Springer, 2019, pp. 375–403.
[10] U. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta, "Amplification by shuffling: From local to central differential privacy via anonymity," in *Proc. SIAM*, 2019, pp. 2468–2479.
[11] J. Allen, B. Ding, J. Kulkarni, H. Nori, O. Ohrimenko, and S. Yekhanin, "An algorithmic framework for differentially private data analysis on trusted processors," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 13657–13668.
[12] B. Bichsel, T. Gehr, D. Drachsler-Cohen, P. Tsankov, and M. Vechev, "DP-Finder: Finding differential privacy violations by sampling and optimization," in *Proc. CCS*, 2018, pp. 508–524.
[13] J. Hayes and O. Ohrimenko, "Contamination attacks and mitigation in multi-party machine learning," in *Proc. NIPS*, 2018, pp. 6604–6615.
[14] I. Kotsogiannis et al., "PrivateSQL: A differentially private SQL query engine," *Proc. VLDB Endowment*, vol. 12, no. 11, pp. 1371–1384, Jul. 2019.
[15] A. Sokolovska and L. Kocarev, "Integrating technical and legal concepts of privacy," *IEEE Access*, vol. 6, pp. 26543–26557, 2018.
[16] N. Volgushev, M. Schwarzkopf, B. Getchell, M. Varia, A. Lapets, and A. Bestavros, "Conclave: Secure multi-party computation on big data," in *Proc. 14th EuroSys Conf.*, Mar. 2019, pp. 1–18.
[17] G. Acs and C. Castelluccia, "I have a DREAM! (DiffeRentially private smart metering)," in *Information Hiding*. Prague, Czech Republic: Springer, 2011, pp. 118–132.
[18] T.-H. H. Chan, E. Shi, and D. Song, "Privacy-preserving stream aggregation with fault tolerance," in *Proc. Int. Conf. Financial Cryptogr. Data Security*. Cham, Switzerland: Springer, 2012, pp. 200–214.
[19] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *Advances in Cryptology—EUROCRYPT*. Berlin, Germany: Springer, 2006, pp. 486–503.
[20] E. Shi, T. H. Chan, E. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," in *Proc. NDSS*, vol. 2, 2011, pp. 1–17.
[21] T. Wang, N. Li, and S. Jha, "Locally differentially private frequent itemset mining," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 127–143.
[22] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. Theory Cryptography Conf.* Cham, Switzerland: Springer, 2006, pp. 265–284.
[23] S. Kotz, T. Kozubowski, and K. Podgorski, *The Laplace Distribution and Generalizations: A Revisit With Applications to Communications, Economics, Engineering, and Finance*. Berlin, Germany: Springer, 2012.
[24] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith, "What can we learn privately?" *SIAM J. Comput.*, vol. 40, no. 3, pp. 793–826, Jan. 2011.
[25] M. Kearns, "Efficient noise-tolerant learning from statistical queries," *J. ACM*, vol. 45, no. 6, pp. 983–1006, Nov. 1998.
[26] Y. Wei et al., "Distributed differential privacy via shuffling vs aggregation: A curious study," *IACR Cryptol. ePrint Arch.*, vol. 2023, p. 1764, 2023.
[27] V. Balcer and A. Cheu, "Separating local & shuffled differential privacy via histograms," in *Proc. ITC*, vol. 163, 2020, pp. 1–14.
[28] N. H. Bshouty and V. Feldman, "On using extended statistical queries to avoid membership queries," *J. Mach. Learn. Res.*, vol. 2, pp. 359–395, Mar. 2002.
[29] L. Reyzin, "Statistical queries and statistical algorithms: Foundations and applications," 2020, *arXiv:2004.00557*.
[30] B. Balle, J. Bell, A. Gascón, and K. Nissim, "Private summation in the multi-message shuffle model," 2020, *arXiv:2002.00817*.
[31] B. Ghazi, R. Pagh, and A. Velingker, "Scalable and differentially private distributed aggregation in the shuffled model," 2019, *arXiv:1906.08320*.
[32] (2023). *San Francisco Fire Department Calls for Service*. [Online]. Available: http://bit.ly/336sddL
[33] M. Abadi et al., "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 308–318.
[34] A. Girgis, D. Data, S. Diggavi, P. Kairouz, and A. T. Suresh, "Shuffled model of differential privacy in federated learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2021, pp. 2521–2529.
[35] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[36] C. Dwork, K. Talwar, A. Thakurta, and L. Zhang, "Analyze gauss: Optimal bounds for privacy-preserving principal component analysis," in *Proc. 46th Annu. ACM Symp. Theory Comput.* New York, NY, USA: Association for Computing Machinery, May 2014, pp. 11–20, doi: 10.1145/2591796.2591883.
[37] W. Qardaji, W. Yang, and N. Li, "Understanding hierarchical methods for differentially private histograms," *Proc. VLDB Endowment*, vol. 6, no. 14, pp. 1954–1965, Sep. 2013.
[38] M. Hay, V. Rastogi, G. Miklau, and D. Suciu, "Boosting the accuracy of differentially-private histograms through consistency," 2009, *arXiv:0904.0942*.
[39] T.-H.-H. Chan, E. Shi, and D. Song, "Private and continual release of statistics," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 3, pp. 1–24, Nov. 2011.
[40] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–90, Feb. 1981.
[41] I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 7417. Berlin, Germany: Springer, 2012, pp. 643–662.
[42] M. Keller, V. Pastro, and D. Rotaru, "Overdrive: Making SPDZ great again," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 10822. Berlin, Germany: Springer, 2018, pp. 158–189.
[43] F. Eigner, A. Kate, M. Maffei, F. Pampaloni, and I. Pryvalov, "Differentially private data aggregation with optimal utility," in *Proc. 30th Annu. Comput. Secur. Appl. Conf.*, Dec. 2014, pp. 316–325.
[44] P. Fauzi, H. Lipmaa, J. Siim, and M. Zajac, "An efficient pairing-based shuffle argument," in *Advances in Cryptology—ASIACRYPT* (Lecture Notes in Computer Science), vol. 10625. Berlin, Germany: Springer, 2017, pp. 97–127.
[45] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Proc. 3rd Innov. Theor. Comput. Sci. Conf.*, Jan. 2012, pp. 309–325.

Yu Wei received the dual bachelor's degrees in information security and law, and the master's degree in computer science from Nankai University, Tianjin, China, in 2018 and 2021, respectively. He is currently pursuing the Ph.D. degree in computer science with Purdue University, West Lafayette, IN, USA. His research interests include differential privacy, applied cryptography, and data privacy protection.

Jingyu Jia received the dual bachelor's degrees in information security and law from Nankai University, Tianjin, China, in 2019, where he is currently pursuing the Ph.D. degree in computer science. His research interests include differential privacy and data privacy protection.

Yuduo Wu received the dual bachelor's degrees in information security and law, and the master's degree in computer science from Nankai University, Tianjin, China, in 2019 and 2022, respectively. Her research interests include differential privacy and data privacy protection.

Changhui Hu received the B.S. degree in mathematics and the Ph.D. degree in information security from Shandong University, Jinan, China, in 2007 and 2012, respectively. Since 2018, he has been a Research Associate with the School of Computer Science, Newcastle University. He is currently a Professor with the School of Cyberspace Security and the School of Cryptology, Hainan University. His research interests include MPC, differential privacy, and privacy in machine learning.



Changyu Dong received the Ph.D. degree from Imperial College London. He is currently a Professor with the Institute of Artificial Intelligence and Blockchain, Guangzhou University. He has authored over 70 publications in international journals and conferences. His research interests include applied cryptography, trust management, data privacy, and security policies. His recent work focuses mostly on designing practical secure computation protocols. The application domains include secure cloud computing and privacy preserving data mining.

Zheli Liu received the B.Sc. and M.Sc. degrees in computer science and the Ph.D. degree in computer application from Jilin University, China, in 2002, 2005, and 2009, respectively. After a post-doctoral fellowship with Nankai University, he joined the College of Computer and Control Engineering, Nankai University, in 2011. He is currently a Professor with Nankai University. His current research interests include applied cryptography and data privacy protection.

Xiaofeng Chen (Senior Member, IEEE) received the B.S. and M.S. degrees in mathematics from Northwest University, Xi'an, China, in 1998 and 2000, respectively, and the Ph.D. degree in cryptography from Xidian University, Xi'an, in 2003. He is currently a Professor with the School of Cyber Engineering, Xidian University. He has authored or coauthored more than 200 research papers in refereed international conferences and journals. His work has been cited more than 10000 times on Google Scholar. His research interests include applied cryptography and cloud computing security. He was the program/general chair or a program committee member of more than 30 international conferences. He is on the editorial board of IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, *Security and Privacy*, and *Computing and Informatics*.

Yun Peng received the B.Sc. degree in computer science from Shandong University (SDU) in 2006, the M.Phil. degree in computer science from the Harbin Institute of Technology (HIT) in 2008, and the Ph.D. degree in computer science from Hong Kong Baptist University (HKBU) in 2013. He is currently a Professor with the Institute of Artificial Intelligence, Guangzhou University. He has published several papers in top-tier conferences and journals, including SIGMOD, VLDB, *The VLDB Journal*, and IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING. His research interests include graph databases, vector databases, and privacy computing. He has served as a Program Committee Member for ICDE, IJCAI, and DASFAA, and a reviewer for IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING and *IJIS*.

Shaowei Wang (Member, IEEE) received the Ph.D. degree from the School of Computer Science and Technology, University of Science and Technology of China (USTC), in 2019. He is currently an Associate Professor with the Institute of Artificial Intelligence and Blockchain, Guangzhou University. He has published over 30 papers in top-tier conferences and journals, such as VLDB, INFOCOM, IJCAI, ICDE, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, and IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING. His research interests include data privacy and federated learning.