Hindawi
Security and Communication Networks
Volume 2017, Article ID 6097253, 14 pages
https://doi.org/10.1155/2017/6097253

Research Article

PMDP: A Framework for Preserving Multiparty Data Privacy in Cloud Computing

Ji Li,[1,2] Jianghong Wei,[1,2] Wenfen Liu,[3] and Xuexian Hu[1]

[1]*State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450002, China*
[2]*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100088, China*
[3]*School of Computer Science and Information Security, Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, Guilin, Guangxi 541004, China*

Correspondence should be addressed to Wenfen Liu; liuwenfen@guet.edu.cn

Received 9 September 2017; Accepted 19 November 2017; Published 12 December 2017

Academic Editor: Krzysztof Szczypiorski

The amount of Internet data is significantly increasing due to the development of network technology, inducing the appearance of big data. Experiments have shown that deep mining and analysis on large datasets would introduce great benefits. Although cloud computing supports data analysis in an outsourced and cost-effective way, it brings serious privacy issues when sending the original data to cloud servers. Meanwhile, the returned analysis result suffers from malicious inference attacks and also discloses user privacy. In this paper, to conquer the above privacy issues, we propose a general framework for Preserving Multiparty Data Privacy (PMDP for short) in cloud computing. The PMDP framework can protect numeric data computing and publishing with the assistance of untrusted cloud servers and achieve delegation of storage simultaneously. Our framework is built upon several cryptography primitives (e.g., secure multiparty computation) and differential privacy mechanism, which guarantees its security against semihonest participants without collusion. We further instantiate PMDP with specific algorithms and demonstrate its security, efficiency, and advantages by presenting security analysis and performance discussion. Moreover, we propose a security enhanced framework sPMDP to resist malicious inside participants and outside adversaries. We illustrate that both PMDP and sPMDP are reliable and scale well and thus are desirable for practical applications.

1. Introduction

With the significantly increasing data size and the rapid development of the corresponding data analysis technology, the original data, which usually has characteristics of big volume, heterogeneity, and low quality, begins to play a very important role in various fields, such as healthcare, advertisement, government decision-making, and transportation. This is mainly because making deep mining and analysis over these large datasets (i.e., big data) would reveal some hidden and valuable information, and further produces great benefits. On the other hand, owing to the characters of big data and the pursuit of better outputs with more complex analysis, big data processing requires more computational overhead and resource expenditure, which challenges the traditional data processing model.

Cloud computing provides a ubiquitous and on-demand approach of accessing a shared pool of configurable computing resources, which can be rapidly provisioned and released with minimal management effort [1]. Therefore, it gives a desirable platform for big data processing and enables users to outsource their computations to cloud servers with powerful computing capabilities sufficient for big data processing. To this end, users need to outsource their original data to cloud servers. However, this will bring serious security and privacy issues, especially when the data is sensitive for users. For exemplary purpose, we consider the following scenario.

The mobile wearable device has been very popular in recent years. With a smart band on your wrist, you can not only collect your own health data like sleep time, heart rate. and motion trail, but also compare your quantity of motion with average level or other people's level. In this case, the data

containing your privacy is collected, aggregated, analyzed, and published by businesses with the assistance of cloud servers, which may result in privacy disclosure of users. Some related reports have been published (https://techcrunch.com/2016/11/03/fitbit-jawbone-garmin-and-mio-fitness-bands-criticized-for-privacy-failings/). Even worse, with the appearance of the sharing economy, user privacy issues become increasingly prominent. For instance, Uber, which brings convenience by providing car-ride service, is accused of allowing its employees to look through and gather users' travel data and device information at will, and its application named God View has always been criticized since it can track users even after they get off the car (https://www.nbcnews.com/tech/tech-news/uber-fined-settlement-ny-over-god-view-tracking-n491706.).

To guarantee data confidentiality and preserve user privacy, various security mechanisms have been developed and employed in each phase of the data life cycle, which roughly includes data storage, data processing, and data publishing (there also exist security issues in the process of data acquisition and data destruction, but they are out of the scope of this paper). For example, attribute-based encryption schemes [2, 3] are used to secure data storage in public cloud servers, secure multiparty computation schemes [4, 5] are introduced to protect data aggregation, and differential privacy mechanisms [6] provide a way to quantize the disclosed privacy in data publishing. In addition, we also note that there are several works proposed to protect data processing and data publishing by combining differential privacy with other cryptography primitives. However, there are few studies concerning the entire privacy preservation throughout the full life cycle of multiparty data, especially in the context of cloud computing. On the other hand, from a practical viewpoint, once the data privacy gets exposed in some phase of the data life cycle, the security mechanisms deployed in other phases would be useless. Therefore, it is necessary to conquer the privacy issues of big data in cloud computing from a global perspective.

1.1. Our Contribution. In this paper we propose a general framework for Preserving Multiparty Data Privacy (PMDP for short) in cloud computing, which provides complete protection throughout the entire life cycle of users' data and is suitable for securing multiparty data aggregation and publication with the assistance of an *untrusted* cloud server. Specifically, the contributions of this study can be summarized as follows:

(1) Based on well studied security mechanisms for preserving user privacy in the process of data storage, processing, and publishing, respectively, we combine these techniques in a nontrivial and tight manner and propose the PMDP framework that covers the full lifecycle of multiple users' data.

(2) We present the principles of choosing the building security mechanisms involved in the PMDP framework and a specific instance. Furthermore, to illustrate the advantage and practicability of the PMDP framework, we evaluate the performance of the instance in terms of efficiency and functionalities by comparing it with other related works.

(3) We formally discuss the security of the PMDP framework. Concretely speaking, we reduce its security to the security of the building mechanisms including fully homomorphic encryption, secure multiparty computation, and differential privacy, which are all with the feature of provable security. Thus, the PMDP framework is also provably secure.

(4) We put forward a reinforced version of the PMDP framework named sPMDP, to provide stronger security and privacy guarantee. In addition, we also show the application scenarios of the PMDP and sPMDP frameworks.

1.2. Outline. The remainder of the paper is organized as follows. In Section 2, we review the related work on techniques for data privacy in different phases. Section 3 introduces some preliminary knowledge used in this paper. The PMDP framework is presented in Section 4. Section 5 illustrates an instantiation of the framework. In Section 6, we evaluate the performance of the framework and discuss its security along with its application scenarios. We propose the reinforced framework sPMDP and analyze its security in Section 7. Finally, some concluding remarks are given in Section 8.

2. Related Work

In this section, under the background of cloud computing, we briefly introduce the security mechanisms used to protect data privacy in each phase of the life cycle of data.

Secure Data Storage. When the data storage is outsourced to cloud servers, data owner completely loses the access control of his/her data. But data owner hopes that the outsourced data can only be accessed by authorized users for privacy issues. A natural solution is to encrypt the data before sending it to cloud servers so that the users holding corresponding secret keys can decrypt the data. Although traditional public key encryption schemes can guarantee data security, they suffer from the limitation of efficiency. A new approach is identity-based encryption, but it is faced with some new challenges [7]. As an extension of identity-based encryption, attribute-based encryption (ABE) [8] enables the data owner to place fine-grained access policy over the outsourced data and can perfectly conquer the problem of securing data storage in cloud computing. For this reason, many ABE schemes [9, 10] with extended functionalities have been proposed. In addition, there are also various privacy-preserving authentication protocols, like two-factor authentication [11, 12], three-factor authentication [13, 14], end-to-end authentication [15], and so on. And some new works focus on practical application fields, such as smart metering [16], Internet of Things [17], and WBAN [18].

Secure Data Processing. The purpose of aggregating and storing data is to make analysis on them and further find the valuable information. However, when the data is outsourced to cloud servers with the above encryption mechanisms,

the data analyst has to download and decrypt the data before processing it, which is not convenient enough to satisfy the data analyzing demands under the background of big data. Fortunately, fully homomorphic encryption (FHE) [19], which has the feature of allowing cloud servers to evaluate arbitrary functions on the encrypted data without decryption, can simultaneously guarantee the security of the data in phases of storage and processing. Due to its significant advantages of securing data sharing, many FHE schemes were proposed to improve the security and efficiency of the original one. Besides, some frameworks for efficient and privacy-preserving outsourced computation have been proposed based on FHE, such as EPOM [20], POFD [21], and POCR [22].

Another important security mechanism used to secure data processing is secure multiparty computation (MPC), which enables multiple users to perform the assigned computation on their collected data and obtain the computation result without getting any information about one another's data. To improve the efficiency of MPC in the context of cloud computing, several variants of MPC have been proposed, such as on-the-fly MPC [23] and cloud-assisted MPC. In addition, motivated by security requirements of practical applications (e.g., outsourced database query, private set intersection, and information retrieval), some efficient and specific cloud-based MPC protocols [24, 25] have been designed.

Secure Data Publishing. The significance of output privacy is remarkable especially under the background of big data. Due to the technology upgrade of data mining, preserving data privacy is getting more and more difficult since the sensitive information in original data would suffer from direct and indirect (via inference) exposure during the mining process. Namely, not only the original data but also the data mining output can lead to the disclosure of sensitive information. So it is necessary to pay attention to output privacy. Anonymization technologies are widely used to preserve data privacy in the process of data publishing. Although classical anonymization methods (e.g., k-anonymity, l-diversity, and t-closeness) have been well studied and there are various corresponding algorithms, they cannot resist structure-based deanonymization attacks [26], which implies that the published data would reveal user privacy. In contrast, differential privacy [27] provides strong theoretical guarantees on the privacy of data by adding noise with specific distributions to raw data. Roughly, the research of differential privacy applied to data publishing is comprised of two aspects, interactive data publishing and noninteractive data publishing. In the first one, the fundamental methods are about responding to queries by disturbing the outcome derived from the original dataset, including Laplace mechanism and exponential mechanism [27]. These methods are easy to implement but the noises achieving privacy protection are relatively big. Afterwards, researchers developed techniques [28] providing responses to queries according to the histogram with noise generated from raw data, which have low sensitivity and comparatively small noise. As for the noninteractive mode, the research results at present are mostly focusing on batch query [29], contingency tab publishing [30], grouping and generalization [31], and sanitized dataset publishing [32].

Secure Data in Multiphase. The MPC technique can protect data privacy in the input and computation process, but it is not designed for output privacy. From the theoretical perspective, it is obvious that MPC cannot guarantee output privacy. With the rapid development of database technology and cloud computing, researchers begin to design security schemes with the capability of preserving data privacy in multiple phases of its lifecycle simultaneously. For instance, Pettai and Laud [33] gave a good example of combining MPC and differential privacy with reasonable performance to achieve both computational and output privacy. However, their framework, which we call DPSharemind, is built upon GUPT [34], which is secure under the assumption of the existence of a *trusted* third party. Consequently, in the situation where multiple clients intend to delegate the computation of a joint function on their data to an untrusted cloud, it can not guarantee the security of the framework. Bindschaedler et al. [35] showed how to obtain a noisy (differentially private) aggregation result in a star network topology using Shamir secret sharing scheme and additively homomorphic encryption; we call their work DPStar. They also ensured that the amount of noise in the final result would neither be reduced by colluding entities nor be influenced by a cheating aggregator secretly, which had important practical significance.

3. Preliminaries

In this section we review the concepts of secure multiparty computation and differential privacy, which are building tools of our PMDP framework.

3.1. Secure Multiparty Computation. Our framework is partially built upon the on-the-fly MPC protocol that is constructed from multikey FHE. In this paper we use the FHE from NTRU encryption scheme of Hoffstein with the modifications of Stehlé and Steinfeld [36]. So we start from the NTRU encryption.

NTRU Encryption. The NTRU cryptosystem is constructed over the ring $R \stackrel{\text{def}}{=} \mathbb{Z}[x]/\langle x^n + 1\rangle$, where $n = 2^u$ for some integer $u \in \mathbb{N}$. Let q be an odd prime number and χ be a B-bounded distribution over R ($B \ll q$). Denote the polynomial ring R/qR by R_q and the coefficient-wise reduction modulo q into the set $\{-\lfloor q/2\rfloor, \ldots, \lfloor q/2\rfloor\}$ by $[\cdot]_q$. Roughly, given a security parameter κ, the NTRU cryptosystem is specialized as follows.

Keygen(1^κ): the key generation algorithm samples polynomials $f', g \leftarrow \chi$ and lets $f = 2f' + 1$. Particularly, if f is not invertible in R_q, then f' is resampled. Denote the inverse of f in R_q by f^{-1}; then the public key pk and the secret key sk are calculated as follows:

$$\begin{aligned} \mathsf{sk} &= f, \\ \mathsf{pk} &= h = \left[2gf^{-1}\right]_q. \end{aligned} \tag{1}$$

Input:
Dataset $T \in \mathsf{R}^n$, length of the dataset n, privacy parameter ϵ, clipping range $(B_{\text{left}}, B_{\text{right}})$.
(1) Let $l = \lceil n^{0.4} \rceil$
(2) Randomly partition T into l disjoint blocks $T_1, \ldots, T_l$
(3) **for** $i = 1$ to l **do**
(4) $\quad O_{T_i} \leftarrow$ output of f_{DP} operates on dataset T_i
(5) $\quad$ If $O_{T_i} < B_{\text{left}}$, then $O_{T_i} \leftarrow B_{\text{left}}$
(6) $\quad$ If $O_{T_i} > B_{\text{right}}$, then $O_{T_i} \leftarrow B_{\text{right}}$
(7) **end for**
(8) **return** $(1/l) \sum_{i=1}^{l} O_{T_i} + \mathsf{Lap}((B_{\text{right}} - B_{\text{left}})/(l \cdot \epsilon))$;

Algorithm 1: Sample-and-Aggregate algorithm [37].

$\mathsf{Enc}(\mathsf{pk}, m)$: the encryption algorithm samples polynomials $s, e \leftarrow \chi$ for plaintext $m \in \{0, 1\}$, and outputs the corresponding ciphertext as follows:

$$c = [hs + 2e + m]_q . \tag{2}$$

$\mathsf{Dec}(\mathsf{sk}, c)$: the decryption algorithm decrypts the ciphertext c by computing $m = [fc]_q \bmod 2$.

With the NTRU encryption scheme and the conversion techniques introduced in [23], we can derive a multikey fully homomorphic encryption scheme. Denote by $\{\mathscr{E}^{(N)} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})\}_{N>0}$ the family of the resulting multikey fully homomorphic encryption schemes and by U the collection of parties. The notation Eval means the homomorphic evaluation performed by the cloud (relinearization and squashing involved in the homomorphic multiplication are not detailed due to space limitations). Then, an on-the-fly MPC protocol secure against semimalicious adversaries can be constructed as follows.

Step 1. For each $i \in U$, the participant P_i samples a key tuple $(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa; r_i)$ and uses pk_i to encrypt his/her input $x_i : c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i)$, where ek_i is an evaluation key. Then $(\mathsf{pk}_i, \mathsf{ek}_i, c_i)$ is sent to a cloud server S. At this point a function F, represented as a circuit C, has been selected on $\{x_i\}_{i \in V}$ for some $V \subseteq U$. Let $N = |V|$.

Step 2. The cloud server S performs homomorphic evaluation on ciphertexts

$$c \longleftarrow \mathsf{Eval}\left(C, \left(c_1, \mathsf{pk}_1, \mathsf{ek}_1\right), \ldots, \left(c_N, \mathsf{pk}_N, \mathsf{ek}_N\right)\right), \tag{3}$$

and broadcasts c to parties $P_1, \ldots, P_N$.

Step 3. By running a general secure MPC protocol $\Pi_{\text{dec}}^{\text{mul}}$, these parties $P_1, \ldots, P_N$ compute the function

$$g_c\left(sk_1, \ldots, sk_N\right) \longleftarrow \mathsf{Dec}\left(sk_1, \ldots, sk_N, c\right). \tag{4}$$

The above on-the-fly MPC protocol can be modified to achieve security against *malicious* adversaries by adding zero-knowledge proofs and succinct noninteractive arguments of knowledge system, which is used in both of our frameworks to guarantee their security property.

3.2. Differential Privacy. Informally, differential privacy guarantees that a single record in a dataset being missing has only limited impact on the outputs of the queries executed on the dataset. The formal definition is captured as follows.

Definition 1 (ϵ-differential privacy [27]). An algorithm $\mathscr{A}$ satisfies ϵ-differential privacy (ϵ-DP) if for any pair of neighboring datasets D and D', and any $S \subseteq \text{Range}(\mathscr{A})$, it holds that

$$\Pr\left[\mathscr{A}(D) = S\right] \leq e^{\epsilon} \cdot \Pr\left[\mathscr{A}\left(D'\right) = S\right], \tag{5}$$

where $\text{Range}(\mathscr{A})$ denotes the collection of all possible outputs of the algorithm $\mathscr{A}$.

The datasets D and D' are neighboring provided that they differ by only one tuple. We denote this by $D \simeq D'$. We can see that the change in the probability distribution of the output caused by adding/removing any single tuple is bounded by e^{ϵ}.

As a major ϵ-differential privacy mechanism, Laplace mechanism perturbs the output of a function f_{DP} on a dataset D by adding to $f_{\text{DP}}(D)$ a noise randomly sampled from the Laplace distribution. We define the global sensitivity of f_{DP} as

$$\Delta f = \max_{(D, D') : D \simeq D'} \left| f_{\text{DP}}(D) - f_{\text{DP}}\left(D'\right) \right|. \tag{6}$$

Then a Laplace mechanism $\mathscr{A}_f$ is given as follows:

$$\mathscr{A}_{f_{\text{DP}}}(D) = f_{\text{DP}}(D) + \mathsf{Lap}\left(\frac{\Delta f}{\epsilon}\right), \tag{7}$$

where

$$\mathsf{Lap}(b) \stackrel{\text{def}}{=} \mathsf{Lap}(x \mid b) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right). \tag{8}$$

Sample-and-Aggregate. The Sample-and-Aggregate technique provides a way to lower the global sensitivity and improve the parallel degree of algorithm and further results in a differentially private method of computing a function f. The basic mechanism is specified as shown in Algorithm 1.

4. Our Framework

In this section, we first introduce the entities involved in the PMDP framework and then provide an overview of the PMDP framework, followed by its details. Before presenting the details of our framework, we summarize the basic notations used throughout this paper in Notations.

4.1. Involved Entities. The PMDP framework involves the following entities:

(1) *Completely trusted authority*: the authority takes charge of producing secret keys for all legal system users. Since the authority can decrypt any ciphertext generated by any user, we thus suppose that it is completely trusted.

(2) *Semitrusted cloud server*: a cloud server has powerful resources of storage and computation that can be easily accessed when required by system users. In the framework it is in charge of performing evaluation part of FHE in a MPC protocol and is assumed to be semitrusted. Namely, the cloud server will honestly complete the given computation assignments but will try to learn the information of the outsourced data.

(3) *System users*: in the PMDP framework, several system users can compute the value of a public function on their private data. Each participant is associated with a unique identifier P_i $(i = 1, 2, \ldots, N)$ and holds the corresponding secure key issued by the authority. Let U be the collection of all parties. Each user is possible to be corrupted by adversaries, which results in the disclosure of the user's secret key.

(4) *Malicious adversaries*: in fact, malicious adversaries do not participate in the procedure of the PMDP framework. But they do exist when we consider the security of the framework. In this paper, only adversaries that can corrupt any subset of $t < N$ parties are considered. In the privacy analysis of our framework, we assume that adversaries have strong background knowledge.

4.2. Overview of the PMDP *Framework.* Roughly, the PMDP framework is a nontrivial and tight integration of MPC and DP proposed by using the Sample-and-Aggregate mechanism, and the enhanced framework sPMDP in Section 7 is also nontrivial and even tighter because it is based on PMDP and its noise addition part is conducted on the cloud in the evaluation stage of MPC.

The PMDP framework consists of the following six stages.

Stage 0. Initially, the authority sets up the system by generating public parameters and corresponding secret keys. Like most of other frameworks, the setup procedure is important to make it work correctly.

Stage 1. Each participant encrypts his/her private data with a multikey fully homomorphic encryption scheme and outsources the resulting ciphertext to the cloud server.

Stage 2. The cloud server identifies the parties involved in the multiparty computation of the corresponding ciphertexts and partitions them into some blocks.

Stage 3. The cloud server operates on the encrypted data with on-the-fly MPC and outputs the calculated results to their corresponding blocks in the form of ciphertext.

Stage 4. These parties in the same block decrypt the returned ciphertext. Furthermore, all these decrypted results from different blocks are aggregated into a final result in accordance with the partition and sample method in the second stage.

Stage 5. Finally, to ensure output privacy, a designated participant first runs a differential privacy mechanism on the final result and then publishes the designated result.

4.3. Details of the PMDP *Framework.* Now we present the details of the PMDP framework. As shown in Figure 1, the entire procedure is comprised of the following phases.

4.3.1. System Setup. Initially, the authority makes the framework specific by choosing appropriate algorithms for each part and presetting related parameters. Since our framework is mainly based on the on-the-fly MPC from multikey FHE, the following components are necessary:

(a) A collection of multikey fully homomorphic encryption schemes with semantical security

$$\mathscr{E} = \left\{\mathscr{E}^{(N)} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})\right\}_{N>0}. \tag{9}$$

(b) A NIZK argument system

$$\Omega^{\mathrm{enc}} = \left(\mathsf{Setup}^{\mathrm{enc}}, \mathsf{Prove}^{\mathrm{enc}}, \mathsf{Verify}^{\mathrm{enc}}, \mathsf{Sim}^{\mathrm{enc}}\right), \tag{10}$$

for the NP relation

$$R^{\mathrm{enc}} = \{((\mathsf{pk}, c), (x, s)) \mid c = \mathsf{Enc}(\mathsf{pk}, x; s)\}. \tag{11}$$

(c) An adaptively extractable SNARK system

$$\Phi = \left\{\mathsf{Setup}^{\Phi}, \mathsf{Prove}^{\Phi}, \mathsf{Verify}^{\Phi}, \mathsf{Ext}^{\Phi}\right\}, \tag{12}$$

for all of NP.

(d) A family of collision-resistant hash functions

$$\mathscr{H} = \{H_{\mathsf{hk}} : \{0,1\}^* \longrightarrow \{0,1\}^{\kappa}\}_{\mathsf{hk}}. \tag{13}$$

(e) An N-party MPC protocol $\Pi_{\mathrm{dec}}^{\mathrm{mul}}$ secure against malicious adversaries corrupting $t < N$ parties, for computing the family of decryption functions

$$\begin{aligned} &g_{c,\mathsf{pk}_1,\mathsf{ek}_1,\ldots,\mathsf{pk}_N,\mathsf{ek}_N}\left((\mathsf{sk}_1, r_1), \ldots, (\mathsf{sk}_N, r_N)\right) \\ &\quad \stackrel{\mathrm{def}}{=} \begin{cases} \mathsf{Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, c), & \text{if } \mathrm{P}_{\mathrm{versuc}}, \\ \perp, & \text{otherwise}, \end{cases} \end{aligned} \tag{14}$$

where

$$\mathrm{P}_{\mathrm{versuc}} = \left\{(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ek}_i) = \mathsf{Keygen}\left(1^{\kappa}; r_i\right), \ \forall i \in [N]\right\}. \tag{15}$$

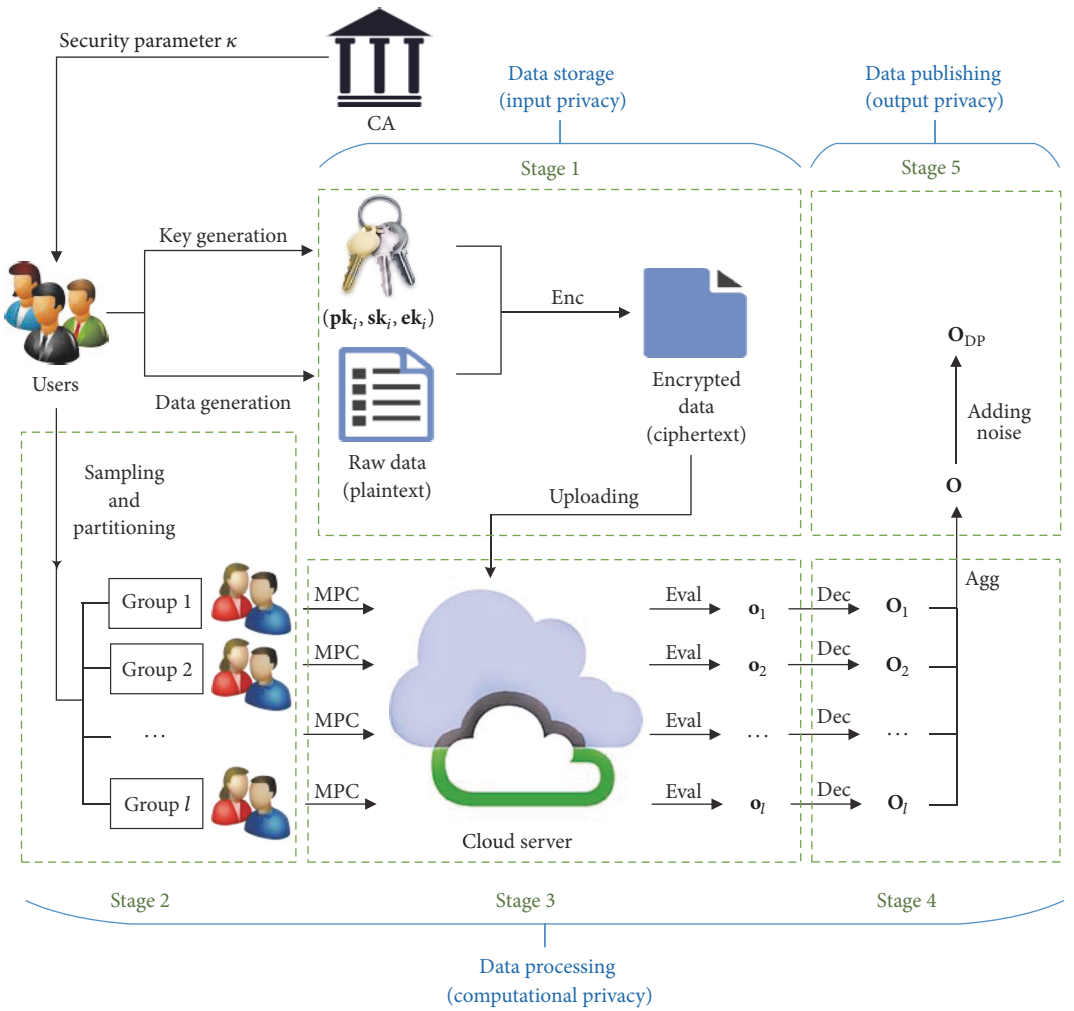

FIGURE 1: The PMDP Framework.

In addition, the method of sampling and partitioning $\Psi^{S\&P}$ should also be determined in this part, after which the aggregation mechanism Λ_{agg} would be explicit. Note that we use differential privacy in the last stage. Since there are several available mechanisms to achieve differential privacy and many variants of the original definition of ϵ-differential privacy, participants should pay attention to select an appropriate one in the light of the function to be computed on the cloud sever.

Moreover, the authority specializes the NIZK proof system Ω^{enc} according to a security parameter κ and send the resulting common reference string crs^{enc} to the cloud server and all parties. The privacy budget ϵ in differential privacy is also initialized.

4.3.2. Data Encryption and Uploading. In this phase all parties encrypt their data and upload them to the cloud server, before making the decision of performing which kind of multiparty computation on the outsourced data. Namely, this phase can be completed in an offline way, which reduces the communication delay of the framework. Specifically, to encrypt his/her data x_i, each party $P_i \in U$ runs the following algorithms:

$$\begin{aligned}(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ek}_i) &\longleftarrow \mathsf{Keygen}(1^\kappa; r_i),\\ c_i &\longleftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i; s_i),\\ \pi_i^{enc} &\longleftarrow \mathsf{Prove}^{enc}((\mathsf{pk}_i, c_i), (x_i, s_i)).\end{aligned} \tag{16}$$

In addition, P_i samples a hash key hk_i and computes a hash value of the ciphertext as $d_i = H_{\mathsf{hk}_i}(c_i)$. Moreover, P_i generates a tuple of verification reference string and private verification key $(\mathsf{vrs}_i, \mathsf{priv}_i) \leftarrow \mathsf{Setup}^{\Phi}(1^\kappa)$. After that, P_i sends the tuple $(\mathsf{pk}_i, \mathsf{ek}_i, c_i, \pi_i^{enc}, \mathsf{hk}_i, d_i, \mathsf{vrs}_i)$ to the cloud server and keeps the corresponding secret/private keys and random values as secret.

Note that after receiving the tuple from P_i, the cloud server will check whether the ciphertext c_i is well formed by verifying the associated proof π_i^{enc}.

4.3.3. Sampling and Partitioning. In this phase all participants agree on the mechanism of sampling and partitioning. So far,

there are a variety of sampling and partitioning mechanisms with different features for differential privacy when dealing with datasets, such as random sampling, uniform (fixed size) sampling, fraction sampling, Bernoulli sampling, random partitioning, and cell-based and kd-tree-based partitioning. Such mechanisms can not only decrease the sensitivity of the function to be computed, but also make the computation paralleled on the cloud server.

Typically, if sampling is not necessary, random partitioning is a common choice due to its simplicity, effectiveness, and practicability. This is the first step of the well-known Sample-and-Aggregate algorithm. By randomly partitioning the original dataset G into l disjoint subsets $\{G_j \mid 1 \le j \le l\}$ with almost the same size, we can perform the secure delegation of multiparty computation on each subset in the next phase. When the original dataset is partitioned, the corresponding collection of parties U is also partitioned into several subsets $\{U_j \mid 1 \le j \le l\}$. The parties belonging to the same U_j form a group.

It is suggested that the size of the blocks obtained from sampling and partitioning should be in the range of $(N^{0.5}, N^{0.6})$, since a too big size makes the sampling and partitioning operation meaningless, and a too small size will threaten the data privacy of participants.

4.3.4. Homomorphic Evaluation. In this phase the cloud server first represents the function F to be computed as a circuit C, and then performs the multiparty computation on each subset G_j $(1 \le j \le l)$.

Concretely, for each $j \in \{1, \ldots, l\}$, let the size of G_j be N, and the corresponding parties be $\{P_1, \ldots, P_N\}$. The cloud server computes

$$o_j \longleftarrow \mathsf{Eval}\left(C, \left(c_1, \mathsf{pk}_1, \mathsf{ek}_1\right), \ldots, \left(c_N, \mathsf{pk}_N, \mathsf{ek}_N\right)\right), \quad (17)$$

and produces succinct arguments $\{\varphi_i\}_{i\in[N]}$ for the following NP language:

$$\begin{aligned} L = \Big\{ & \left(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i\right) \mid \exists \left(\widetilde{c}_1, \widetilde{\pi}_1^{\mathrm{enc}}\right), \ldots, \left(\widetilde{c}_N, \widetilde{\pi}_N^{\mathrm{enc}}\right) \text{ s.t. } d_i \\ & = H_{\mathsf{hk}_i}\left(\widetilde{c}_i\right), \ \mathsf{Verify}^{\mathrm{enc}}\left(\left(\mathsf{pk}_i, \widetilde{c}_i\right), \widetilde{\pi}_i^{\mathrm{enc}}\right) = 1, \ o_j \\ & = \mathsf{Eval}\left(C, \left(\widetilde{c}_1, \mathsf{pk}_1, \mathsf{ek}_1\right), \ldots, \left(\widetilde{c}_N, \mathsf{pk}_N, \mathsf{ek}_N\right)\right)\Big\}. \end{aligned} \quad (18)$$

4.3.5. Decryption and Aggregation. In this phase all participants decrypt the evaluated ciphertext and further aggregate the resulting data.

First, for $j = 1$ to l, each participant P_i belonging to the group G_j runs the algorithm $\mathsf{Verify}^{\Phi}(\{(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i)\}_{i\in[N]}, \varphi_i)$ to verify the validity of the argument φ_i. If the verification is successful for all parties in the same group, then the MPC protocol $\Pi_{\mathrm{dec}}^{\mathrm{mul}}$ is performed to compute

$$\begin{aligned} O_j &= g_{o_j, \mathsf{pk}_1, \mathsf{ek}_1, \ldots, \mathsf{pk}_N, \mathsf{ek}_N}\left(\left(\mathsf{sk}_1, r_1\right), \ldots, \left(\mathsf{sk}_N, r_N\right)\right) \\ &\stackrel{\mathrm{def}}{=} \begin{cases} \mathsf{Dec}\left(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, o_j\right), & \text{if } \mathrm{P}_{\mathrm{versuc}}, \\ \perp, & \text{otherwise.} \end{cases} \end{aligned} \quad (19)$$

Second, these participants from different groups merge their outputs into a common output by performing the aggregation mechanism

$$O \longleftarrow \Lambda_{\mathrm{agg}}\left(\left\{O_j, \ j = 1, \ldots, l\right\}\right). \quad (20)$$

To simplify the above aggregation procedure, the party with the most computing power in each group will be assigned as the agent of the group, and all these agents are designated to accomplish the aggregation work. Moreover, the aggregation procedure can also be outsourced to the cloud server by using a secure delegation protocol, for example, the cloud-assisted MPC from threshold FHE [5].

4.3.6. Privacy-Preserving Result Release. To preserve the output privacy well, in this phase, the noise generated by differential privacy mechanisms is added to the aggregated output. Specifically, the following issues should be synthetically considered by parties before sampling the noise.

First, since there are a few variants of ϵ-differential privacy, it is necessary and important to make a proper choice among them according to the participants' requirements. Particularly, the (ϵ, δ)-differential privacy is the most widely used one and is adaptable to the situation with less restriction of privacy loss. Personalized differential privacy provides more accurate control of the consumption of privacy budget at an individual level and is usually used in interactive queries. Concentrated differential privacy gives high probability bounds for cumulative loss of numerical computations and works well in the aspect of preserving group privacy. All in all, the choice should be based on the computation to be made on the cloud server and the desired privacy-preserving level.

Second, researchers have developed several differential privacy mechanisms, which can achieve the same DP definition in different application scenes. For example, the Laplace mechanism can achieve ϵ-DP for real valued queries, while the exponential mechanism is a ϵ-DP method of sampling from a discrete set of candidate outputs. Thus, if parties want to publish their average wage in a differentially private way, the Laplace mechanism is available; but if they want to release the result of their secure voting for a new leader, the exponential mechanism is a better option.

Third, by the definition of differential privacy, we know that the sensitivity of the function is an important parameter in the specific realization of differential privacy. Additionally, the computation outsourced to the cloud server and the used aggregation mechanism significantly affect the magnitude of sensitivity. Therefore, it is essential to pay attention to both of them.

The last issue needed to be considered is which party should be responsible for generating, handling, and adding the noise. It seems that any party can take on such a task. But this is built upon the assumption that all parties are at least semihonest without collusion in the result release phase. Otherwise, malicious participants may try to reduce the differential privacy of honest participants through sampling a noise that is smaller than the required magnitude, and

semihonest participants may collude to disclose the information of honest participants by sharing their input data and subtracting the noise term from the published result. In our framework we assume that all participants are honest, or at least semihonest without collusion in this phase. As a result, the output privacy can be guaranteed by selecting an agent of all parties to run the differential privacy mechanism on the aggregated result.

5. An Instantiation of the Framework

In this section, to illustrate the effectiveness of the PMDP framework, we present an instance of the PMDP framework and show how to use it to solve the problem that N parties securely compute and publish their average wage with the help of a semitrusted cloud sever. Specifically, this instantiated framework consists of the following stages.

Stage 0. The authority chooses the following cryptographic primitives:

(a) The family of multikey FHE schemes proposed by Stehlé and Steinfeld [36]

$$\mathscr{E}_{\mathrm{LA}} = \left\{\mathscr{E}^{(N)} = \left(\mathsf{Keygen}_{\mathrm{LA}}, \mathsf{Enc}_{\mathrm{LA}}, \mathsf{Dec}_{\mathrm{LA}}, \mathsf{Eval}_{\mathrm{LA}}\right)\right\}_{N>0}. \tag{21}$$

(b) The NIZK argument system constructed by Groth et al. [38]

$$\Omega_{\mathrm{LA}}^{\mathrm{enc}} = \left(\mathsf{Setup}_{\mathrm{LA}}^{\mathrm{enc}}, \mathsf{Prove}_{\mathrm{LA}}^{\mathrm{enc}}, \mathsf{Verify}_{\mathrm{LA}}^{\mathrm{enc}}, \mathsf{Sim}_{\mathrm{LA}}^{\mathrm{enc}}\right), \tag{22}$$

for the NP relation

$$R^{\mathrm{enc}} = \{((\mathsf{pk}, c), (x, s)) \mid c = \mathsf{Enc}(\mathsf{pk}, x; s)\}. \tag{23}$$

Let $\mathrm{crs}^{\mathrm{enc}}$ be the common reference string for $\Omega_{\mathrm{LA}}^{\mathrm{enc}}$.

(c) The adaptively extractable SNARK system presented by Bitansky et al. [39]

$$\Phi_{\mathrm{LA}} = \left\{\mathsf{Setup}_{\mathrm{LA}}^{\Phi}, \mathsf{Prove}_{\mathrm{LA}}^{\Phi}, \mathsf{Verify}_{\mathrm{LA}}^{\Phi}, \mathsf{Ext}_{\mathrm{LA}}^{\Phi}\right\}, \tag{24}$$

for all of NP.

(d) The family of cryptographically collision-resistant hash functions HmacSHA256

$$\mathscr{H}^{\mathrm{I}} = \left\{H_{\mathsf{hk}}^{\mathrm{I}} : \{0,1\}^{*} \longrightarrow \{0,1\}^{256}\right\}_{\mathsf{hk}}. \tag{25}$$

(e) The cloud-assisted N-party MPC protocol proposed by Asharov et al. [5] for computing the family of decryption functions

$$g_{c,\mathsf{pk}_1,\mathsf{ek}_1,\ldots,\mathsf{pk}_N,\mathsf{ek}_N}^{\mathrm{LA}}\left((\mathsf{sk}_1, r_1), \ldots, (\mathsf{sk}_N, r_N)\right) \stackrel{\mathrm{def}}{=} \begin{cases} \mathsf{Dec}_{\mathrm{NTRU}}\left(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, c\right), & \text{if } \mathrm{P}_{\mathrm{versuc}}^{256}, \\ \perp, & \text{otherwise,} \end{cases} \tag{26}$$

$$\mathrm{P}_{\mathrm{versuc}}^{256} = \left\{\left(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ek}_i\right) = \mathsf{Keygen}\left(1^{256}; r_i\right),\ \forall i \in [N]\right\}.$$

Stage 1. Each party $P_i \in U$ first performs the following algorithms:

$$\begin{aligned} \left(h_i, f_i, \left(\boldsymbol{\gamma}^{(i)}, \boldsymbol{\zeta}^{(i)}\right)\right) &= \left(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ek}_i\right) \longleftarrow \mathsf{Keygen}_{\mathrm{LA}}\left(1^{\kappa}; r_i\right), \\ c_i &= \left[h_i s_i + 2e_i + x_i\right]_q \longleftarrow \mathsf{Enc}\left(\mathsf{pk}_i, x_i; s_i\right), \\ \pi_i^{\mathrm{enc}} &\longleftarrow \mathsf{Prove}^{\mathrm{enc}}\left(\left(\mathsf{pk}_i, c_i\right), \left(x_i, s_i\right)\right), \\ d_i &= \mathscr{H}_{\mathsf{hk}_i}^{\mathrm{I}}\left(c_i\right), \\ \left(\mathsf{vrs}_i, \mathsf{priv}_i\right) &\longleftarrow \mathsf{Setup}_{\mathrm{LA}}^{\Phi}\left(1^{256}\right), \end{aligned} \tag{27}$$

where s_i, e_i are randomly sampled from χ and x_i is P_i's wage (the new symbols $\boldsymbol{\gamma}^{(i)}, \boldsymbol{\zeta}^{(i)}$ are related to ek, which can be further referred to in [23]).

Then, P_i sends the tuple $(\mathsf{pk}_i, \mathsf{ek}_i, c_i, \pi_i^{\mathrm{enc}}, \mathsf{hk}_i, d_i, \mathsf{vrs}_i)$ to the cloud server, who will verify the correctness of the proof π_i^{enc}. These values (f_i, r_i), (hk_i, d_i), and priv_i are locally maintained by P_i.

Stage 2. After gathering the encrypted data from all parties, the cloud sever forms them as a ciphertext dataset G. Then, by calling the Sample-and-Aggregate algorithm [37], it randomly partitions G into $l = \lfloor N^{0.4} \rfloor$ disjoint subsets $\{G_j \mid 1 \le j \le l\}$, which are with almost the same size. Meanwhile, the collection of all parties U is also partitioned into several corresponding groups $\{U_j \mid 1 \le j \le l\}$.

Stage 3. Since the problem is to compute the average wage of all parties, the cloud server needs to conduct the mean value function

$$F_{\mathrm{avr}} = \frac{1}{N} \cdot \sum_{i=1}^{N} x_i, \tag{28}$$

which is represented as a circuit C_{avr}. To this end, for each subgroup $G_j = \{P_1, \ldots, P_{N_j}\}$ $(1 \le j \le l)$, the cloud server computes

$$o_j \longleftarrow \mathsf{Eval}_{\mathrm{LA}}\left(C_{\mathrm{avr}}, \left(c_1, h_1, \mathsf{ek}_1\right), \ldots, \left(c_{N_j}, h_{N_j}, \mathsf{ek}_{N_j}\right)\right), \tag{29}$$

and generates the succinct arguments $\{\varphi_i\}_{i \in [N_j]}$ for the following NP language:

$$\begin{aligned} L_j = \Big\{ &\left\{\left(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i\right)\right\}_{i \in N_j} \mid \exists \left(\widetilde{c}_1, \widetilde{\pi}_1^{\mathrm{enc}}\right), \ldots, \\ &\left(\widetilde{c}_{N_j}, \widetilde{\pi}_{N_j}^{\mathrm{enc}}\right) \text{ s.t. } d_i = H_{\mathsf{hk}_i}^{\mathrm{I}}\left(\widetilde{c}_i\right), \\ &\mathsf{Verify}^{\mathrm{enc}}\left(\left(\mathsf{pk}_i, \widetilde{c}_i\right), \widetilde{\pi}_i^{\mathrm{enc}}\right) = 1,\ o_j \\ &= \mathsf{Eval}\left(C, \left(\widetilde{c}_1, \mathsf{pk}_1, \mathsf{ek}_1\right), \ldots, \left(\widetilde{c}_{N_j}, \mathsf{pk}_{N_j}, \mathsf{ek}_{N_j}\right)\right)\Big\}. \end{aligned} \tag{30}$$

Stage 4. Fist, each party P_i belonging to U_j $(1 \le j \le l)$ runs the algorithm $\mathsf{Verify}_{\mathrm{LA}}^{\Phi}(\{(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i)\}_{i \in [N_j]}, \varphi_i)$ to verify the argument φ_i. If verification is successful for all parties in the

same group, they perform the cloud-assisted MPC protocol to compute the following function.

$$O_j = g^{\mathrm{LA}}_{o_j,h_1,\mathsf{ek}_1,\ldots,h_N,\mathsf{ek}_N}\left(\left(f_1, r_1\right), \ldots, \left(f_{N_j}, r_{N_j}\right)\right) \overset{\mathrm{def}}{=} \begin{cases} \left[f_1 \cdots f_{N_j} \cdot o_j\right]_q \pmod 2, & \text{if } \mathrm{P}^{256}_{\mathrm{versuc}}, \\ \perp, & \text{otherwise.} \end{cases} \tag{31}$$

Then, the cloud server broadcasts $(c, \varphi_1, \ldots, \varphi_{N_j})$ to all the parties $P_1, \ldots, P_{N_j}$ in the same group U_j together with the tuple $\{(h_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i)\}_{i\in[N_j]}$.

Then, the agent in the group G_j computes

$$O_j = \begin{cases} B_{\mathrm{left}}, & \text{if } O_j < B_{\mathrm{left}}, \\ O_j, & \text{if } B_{\mathrm{left}} \le O_j \le B_{\mathrm{right}}, \\ B_{\mathrm{right}}, & \text{if } O_j > B_{\mathrm{right}}, \end{cases} \tag{32}$$

where B_{left}, B_{right} are the left and right bounds in the Sample-and-Aggregate algorithm.

Furthermore, all agents calculate the following function with the help of on-the-fly MPC protocol:

$$O = \frac{1}{l} \cdot \sum_{j=1}^{l} O_j. \tag{33}$$

Stage 5. All parties vote for a publisher from all subgroup agents and authorize him/her to sample a noise

$$\eta_0 \longleftarrow \mathsf{Lap}\left(\frac{B_{\mathrm{right}} - B_{\mathrm{left}}}{l \cdot \epsilon}\right), \tag{34}$$

and publish the finial average wage of all parties as

$$O_{\mathrm{DP}} = O + \eta_0, \tag{35}$$

where ϵ is the privacy budget.

6. Performance Discussion and Security Analysis

In this section we first systematically analyze the security of the PMDP framework along with its application scenarios. Then, we roughly discuss the performance of the PMDP framework in terms of computation overhead and security property.

6.1. Security Analysis. To simplify the security analysis of the PMDP framework, we separate participants into two categories, honest and semihonest, as in the security analysis of MPC protocol. On the other hand, the cloud server is always assumed to be untrusted. In addition, since the framework uses the differential privacy mechanism, we thus take the background knowledge attack into consideration. Below we present the security analysis of our framework in honest model and semihonest model, respectively. Since we extend the security models in traditional MPC protocols to those in our sPMDP framework, in the following security analysis we mainly focus on demonstrating that the output privacy cannot be violated since the input and computational privacy are already proven to be guaranteed by former works on MPC [23].

6.1.1. Honest Model. We first show that the PMDP framework does preserve the data privacy under the assumption that all participants are honest and do not attempt to get others' information. That is, each participant will honestly follow the framework procedure as required.

Specifically, in Stage 2 of the framework, each participant's data is encrypted with FHE and then outsourced to the cloud server. Thus, the data privacy in the storage phase is ensured by the security of the underlying FHE. In Stage 4, all data are computed in the form of ciphertext with on-the-fly MPC protocol based on multikey FHE, whose security and correctness have been proved before. Therefore, data privacy in the processing phase is also protected, which means that each participant will only know his/her input and the output of his/her group. The operations of sampling, partitioning, and aggregation in Stages 3 and 5 do not influence the data privacy or the correctness of intermediate results if chosen appropriately, though there may be accuracy loss. The aggregated result is transformed into differentially private form and published in Stage 6, which ensures the output privacy owing to the theoretical guarantee of differential privacy. To sum up, in the honest model, we can see that the PMDP framework preservers the data privacy of all participants throughout its lifecycle, without affecting the data usability.

Although the framework in the honest model is very simple and seems to be idealized, it can be deployed in particular applications. An actual example is that some medical facilities try to obtain meaningful insights on health according to analyses of their clients' health data. Obviously, these facilities have the need for health data storage, processing, and release. By calling our framework, these facilities can be regarded as participants, whose data is taken good advantage of without any risk of privacy disclosure. In this instance, the medical facilities only care about the features, objective laws, and tendency of the collectivity rather than information of individuals, so it complies with the definition of honest model.

6.1.2. Semihonest Model. In the semihonest model, a semihonest participant will follow the framework procedure but will also try to learn information about other parties. Moreover, a semihonest participant may collude with others, including other semihonest participants and external adversaries with background knowledge. In this work we assume that there is at least one honest party in the collection of all parties. We will show that the PMDP framework is secure only when there is no collusion in the semihonest model.

Firstly, we consider the noncolluding situation. For a semihonest participant $P_s \in U_s$, he/she knows his/her input $x_s \in G_s$, the local result of his/her group O_s, the aggregated global result of all groups O, the noise η_0, and the output O_{DP}. Besides, he/she knows the method of sampling, partitioning,

Table 1: Comparisons of security properties with other related works.

Related works	Delegation of storage	Input privacy	Computational privacy	Output privacy	Cloud-assisted	Distributed computing	Server type	Security model
GUPT [34]	×	×	×	√	√	×	Trusted	Noncolluding & Semihonest
DPSharemind [33]	×	√	√	√	√	√	Trusted	Noncolluding & Semihonest
DPStar [35]	×	√	√	√	√	×	Untrusted	Semihonest
PMDP	√	√	√	√	√	√	Untrusted	Noncolluding & Semihonest
sPMDP	√	√	√	√	√	√	Untrusted	Malicious

The security model is about the types of the participants of the works.

and aggregation, the function F computed on the cloud server, and who are in the same group U_s with him/her. If he/she does not collude with anyone, he/she can not infer anyone's input from what has been known, and the reason is not far to seek. From what has been known, P_s has the following equation.

$$F\left(\{x_i \mid x_i \in G_s\}\right) = O_s. \tag{36}$$

As we suggested before, the size of each group is at least $\lceil N^{0.5} \rceil$. Considering our framework is designed to meet the challenge of big data privacy, we can assume that $N \geq 10$, which implies that $|G_s| \geq 3$. For P_s, he/she only knows x_s, so there are at least 2 unknown elements in (36), meaning that it is unsolvable and P_s can not get others' inputs.

Now we consider the colluding situation. We present two possible attacks aiming at the PMDP framework from semihonest participants colluding with others.

Case 1. In each group U_j, there is only one honest party P^*; the others are semihonest and can collude with one another. Then, these semihonest parties can infer the input of P^* as follows. Recall that in the following equation:

$$F\left(x_1, x_2, \ldots, x_{P^*}, \ldots, x_N\right) = O_j, \tag{37}$$

the only unknown variable is x_{P^*}. It is not hard to solve (37) if the function F is not too complicated. If there is an external adversary who corrupts $N-1$ parties of an N-party group, he/she can conduct a similar attack.

Case 2. In each group U_j, there is only one semihonest party P°, and the others are honest. Suppose that the party P° is corrupted by an external adversary with strong background knowledge; for example, the adversary knows the input of each party in G_j except a party P^*. Then, the adversary can obtain O_j from P° and launch an attack by solving the equation in Case 1 to get x_{P^*}.

We note that an entity who wants to attack the framework must possess the inputs of at least $N-1$ parties in an N-party group. The condition is a little strict, but if the cloud server is corrupted and the sampling and partitioning method is decided by the cloud server, it will be much easier.

The above two cases suggest that the framework may suffer from potential attacks when some participants are semihonest. To avoid these attacks in the semihonest model, the framework needs to be reformed.

6.2. Performance Discussion. Since the PMDP framework involves several general security mechanisms, it is difficult to accurately evaluate its computation and computation costs. Thus, we show its efficiency by discussing the performance of the instance presented in prior section.

In the instance, each general party P_i, who is not assigned to be an agent, needs to generate his/her public/secret keys and perform encryption and NIZK operations in Stage 2. Moreover, P_i is also in charge of running the verification and decryption algorithms in Stage 5. Thus, P_i's computation cost is roughly the same as that of each party in an on-the-fly MPC protocol. According to López-Alt et al.'s [23] analysis, we know that the computation cost of each general party P_i is at most polylogarithmic in the circuit size $|C|$ and the total size of all inputs and polynomial in his/her own input x_i. For an assigned agent, he/she has an extra calculation task, that is, the aggregation operation. Mostly, the aggregation complexity is $\mathcal{O}(|U_j|)$ for the agent of group U_j. In addition, since we assume that the cloud server has sufficient computation resources, we thus omit its computation overhead.

In Table 1 we compare the PMDP framework with other related works from the aspect of security properties, including delegation of storage, privacy in different processes, the reliability of cloud server, security model, and whether they support distributing computation. We select three representative works for comparison, namely, the GUPT system [34], the scheme designed by Pettai and Laud [33], which we call DPSharemind, and the protocol from Bindschaedler et al. [35], which we call DPStar. The last row is about the sPMDP framework, which we will introduce later.

From Table 1, we can see that overall PMDP overweighs the works in the first three rows. Firstly, PMDP enjoys the advantage in delegation of storage, which is important in the era of big data. This is because it employs FHE, while GUPT does not consider data storage and both DPSharemind and DPStar employ secret sharing, thus breaking the integrity of data. Consequently, all works but GUPT can provide lifelong data privacy guarantee, covering input privacy, computational privacy, and output privacy. All of the works can be cloud-assisted, but only PMDP and DPStar allow the cloud server to be untrusted. The security model is mainly about the participants of the works. We have shown that PMDP is secure under noncolluding semihonest model, and this is where it is weaker than DPStar. However, DPStar does not support distributed computing; only PMDP and DPSharemind do since they use sampling and aggregation. In order

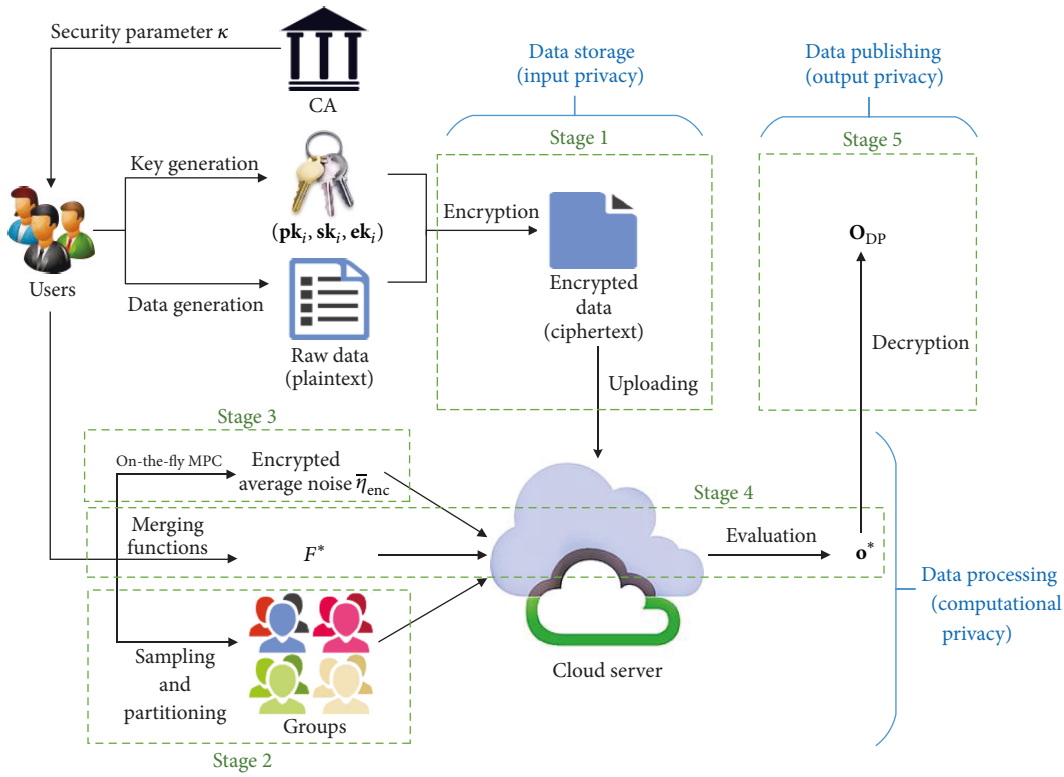

FIGURE 2: Security enhanced PMDP framework.

to make up for the shortcoming of PMDP in security model, we propose the sPMDP framework in the next section.

Before introducing sPMDP, we further illustrate the differences between PMDP and DPStar, which is a latest public work in addressing the privacy concerns related to multiparty computation. The fundamental difference is that DPStar uses Shamir secret sharing and homomorphic encryption, and PMDP uses on-the-fly MPC from multikey FHE. As a result, the security model of PMDP can be strengthened easily while that of DPStar can not. Besides, in [35] DPStar is firstly introduced as a summation protocol and extended to other queries including count queries, histograms, and linear combinations, meaning that it has to be readjusted and several new operations have to be added when facing different queries. By contrast, PMDP is a framework and all of its details have been explained; thus it can be easily applied to different queries. So the generality and usability of DPStar is not as well as PMDP. From what has been mentioned above, we can conclude that DPStar has good security properties and important practical significance, and our PMDP framework has advantage over it in delegation of storage, distributed computing, generality, and usability. As a reinforced version of PMDP, sPMDP also has advantage over DPStar in security model, and we prove this in the next section.

7. Security Enhanced PMDP Framework

The main reason that the PMDP framework suffers from the above attacks is that too much information is accessible to participants. If the intermediate result O_j of each group G_j is unknown to each one, then some attacks are not effective anymore. Furthermore, the aggregated result O and noise η_0 should also be not available to any participant; otherwise the adversary with strong background knowledge might be able to infer users' inputs. Motivated by the above observations, we propose a security enhanced PMDP framework (sPMDP for short).

Since the initialization mechanism and cryptography primitives used in the sPMDP framework are similar to that in PMDP, we briefly introduce its details. As shown in Figure 2, the sPMDP framework consists of the following stages.

Stage 0. Initialize the system as in the PMDP framework.

Stage 1. N parties encrypt their inputs with multikey FHE and upload the resulting ciphertexts to the cloud server.

Stage 2. Parties decide the method of sampling and partitioning and obtain groups $\{G_j \mid 1 \le j \le l^*\}$.

Stage 3. Each party samples a noise η_i as the agent does in the PMDP framework, and all parties calculate the ciphertext of the average noise of all parties with the help of an on-the-fly MPC protocol. Particularly, we use the protocol without the decryption, and the ciphertext of average noise will be used in the next stage. The average noise is denoted by $\overline{\eta}$ and its ciphertext is denoted by $\overline{\eta}_{enc}$. The value of $\overline{\eta}$ will not be known by any entity in the whole procedure.

Stage 4. All parties (or an agent of all parties) firstly merge the original function F to be computed on each group and the aggregation function for all groups Λ_{agg} into one function F', which is taken as inputs of all parties' inputs

$$\begin{aligned} F' &= \mathsf{Merge}\left(F, \Lambda_{\text{agg}}\right) \\ &= \Lambda_{\text{agg}}\left(\left\{F\left(G_j\right),\ j = 1, \ldots, l^*\right\}\right). \end{aligned} \tag{38}$$

Moreover, let F^* denote the sum of F' and $\overline{\eta}$:

$$F^* = F' + \overline{\eta}. \tag{39}$$

Then, the function F^* is represented as a circuit C^*, and the cloud server computes

$$o^* \longleftarrow \mathsf{Eval}\left(C^*, \left(c_1, \mathsf{pk}_1, \mathsf{ek}_1\right), \ldots, \left(c_N, \mathsf{pk}_N, \mathsf{ek}_N\right)\right). \tag{40}$$

The cloud server also produces succinct arguments $\{\varphi_i^*\}_{i\in[N]}$ for the NP language L^* similar to the language L in the original framework.

Stage 5. Each party P_i runs the algorithm

$$\mathsf{Verify}^{\Phi}\left(\left\{\left(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i\right)\right\}_{i\in[N]}, \varphi_i^*\right), \tag{41}$$

to verify the validity of the argument φ_i^*. If the verification is successful for all parties in the same group, they run an MPC protocol Π_{dec}^* to compute the function

$$\begin{aligned} O_{\text{DP}} &= g_{o^*, \mathsf{pk}_1, \mathsf{ek}_1, \ldots, \mathsf{pk}_N, \mathsf{ek}_N}\left(\left(\mathsf{sk}_1, r_1\right), \ldots, \left(\mathsf{sk}_N, r_N\right)\right) \\ &\overset{\text{def}}{=} \begin{cases} \mathsf{Dec}\left(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, o^*\right), & \text{if } \mathrm{P}_{\text{versuc}}, \\ \perp, & \text{otherwise}, \end{cases} \end{aligned} \tag{42}$$

$$\mathrm{P}_{\text{versuc}} = \left\{\left(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ek}_i\right) = \mathsf{Keygen}\left(1^{\kappa}; r_i\right),\ \forall i \in [N]\right\},$$

and O_{DP} is released as the final result.

As to the security of sPMDP against malicious adversaries, we prove it from two different perspectives. Firstly, we demonstrate the security by theoretical proof. The difference between PMDP and sPMDP is obvious. In sPMDP, we integrate the operations of sampling, aggregation, generating, and adding noise into the on-the-fly MPC scheme, thus letting the on-the-fly MPC scheme provide global security guarantee. The aim of merging functions is to turn the original evaluation function and the function of average noise addition into one function conducted by the cloud, thus protecting the amount of noise against malicious participants. Another advantage of merging functions besides enhancing security is that it optimizes the structure of PMDP framework, reducing participants assignments and achieving a better combination between different techniques. So sPMDP is a tighter combination between MPC protocol and differential privacy. It has been proved in [23] that the on-the-fly MPC scheme is secure against malicious adversary, which means that sPMDP is secure under malicious model. So the security property of sPMDP is better than PMDP and DPStar. It is remarkable that the computation of encrypted average noise also uses the on-the-fly MPC scheme, thus ensuring that no one will know about others' noise or the value of average noise.

Now we roughly explain the security of the sPMDP framework from another perspective. Note that each party in sPMDP only knows his/her own input x_i, noise η_i, and the final result O_{DP}. With all these values, each party can only get the equation

$$F^*\left(x_i, \eta_i,\ i = 1, \ldots, N\right) = O_{\text{DP}}; \tag{43}$$

that is to say

$$F'\left(x_i,\ i = 1, \ldots, N\right) + \frac{1}{N_U}\sum_{i=1}^{N_U}\eta_i = O_{\text{DP}}. \tag{44}$$

Because there is at least one honest party, w.l.o.g., we assume that party P^* is honest, who will not reveal his input x_{P^*} and noise η_{P^*} to others. Therefore there are at least two unknowns, namely, x_{P^*} and η_{P^*}, in (44) for parties other than P^*. From the perspective of solutions of the equation, the equation has infinitely many solutions when it has more than one unknown with no more conditions. Therefore the input and the noise of party P^* will always be unknown to others. Again, we see that in sPMDP the privacy of honest parties will be preserved.

As the cost of stronger security guarantees, the computation overhead of each party in sPMDP is larger than that in PMDP, and the additional part comes from the computation of the encrypted average noise in Stage 4. Overall, its computation cost also follows the on-the-fly MPC protocol.

8. Conclusion

In this work we focus on the problem of how to preserve multiparty data privacy throughout its lifecycle in cloud computing and propose a privacy-reserving framework named PMDP. This framework is built upon the on-the-fly MPC, sampling, partitioning, and aggregation mechanisms, and differential privacy, thus guaranteeing input privacy, computational privacy, and output privacy in cloud computing simultaneously, even if the cloud server is untrusted. The security analysis shows that the framework can achieve intended security goals in the honest model. To conquer those potential attacks in the semihonest model, we further present a security enhanced PMDP framework. The performance discussion indicates that the proposed framework owns advantages in security guarantees and thus is more desirable for secure multiparty data aggregation and publishing.

Notations

q: An odd prime number
R: A polynomial ring
χ: A distribution over a polynomial ring
κ: Security parameter
m, c: Plaintext and its ciphertext

$(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ek}_i)$: The public, secret, and evaluation key
o, O: Some intermediate results in a framework
ϵ: The privacy budget
Δf: The sensitivity of f_{DP}
$\mathscr{A}$: An algorithm satisfying ϵ-differential privacy
Range($\mathscr{A}$): All possible outputs of $\mathscr{A}$
D, D': A pair of neighboring datasets
O_{DP}: The differentially private output
f_{DP}: A function on dataset
$B_{\mathrm{left}}, B_{\mathrm{right}}$: Clipping range
P_i: The participant of the framework
N: The number of participants
x_i: The input of a participant
l, l^*: The number of blocks (groups)
T, T_i: The original dataset and its subsets
$\mathscr{C}$: The original dataset and its subsets
Φ: An adaptively extractable SNARK system
Ω^{enc}: A NIZK argument system
R^{enc}: NP relation
$\mathscr{H}$: A family of collision-resistant hash function
Π_{dec}: An N-party decryption MPC protocol
$g_{[\cdot]}$: Decryption functions
hk_i: Hash keys
d_i: The hash digest of the ciphertext
π_i^{enc}: A NIZK of the ciphertext
vrs_i: A verification reference string
priv_i: A private verification key
$\mathrm{P}_{\mathrm{versuc}}$: The verification of argument is successful
$\mathrm{crs}^{\mathrm{enc}}$: Security parameter
U, U_j: The set of parties and its subset
G, G_j: The set of all parties' data and its subset
P^*: An honest participant
P_s, P°: Semihonest participants
Λ_{agg}: The aggregation function
r_i: Random value
f, g, s_i, e_i: Polynomials sampled from some distribution
F: The function that multiparties want to compute
F': The mergence of F and aggregation function
F^*: The sum of F' and average noise
C, C^*: The circuit of F and F^*
η_i: The random noise following some distribution
$\overline{\eta}, \overline{\eta}_{\mathrm{enc}}$: Average noise and its ciphertext.

Conflicts of Interest

The authors declare that they have no conflicts of interest

Acknowledgments

This work was supported in part by the National Nature Science Foundation of China under Grant 61702549, Grant 61502527, and Grant 61379150 and in part by the Open Foundation of State Key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications) under Grant SKLNST-2016-2-22.

References

[1] M. Armbrust, A. Fox, R. Griffith et al., "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 12, pp. 2706–2716, 2016.

[3] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A Secure and Dynamic Multi-Keyword Ranked Search Scheme over Encrypted Cloud Data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 340–352, 2016.

[4] R. Cramer, I. DamgÅrd, and U. Maurer, "General secure multi-party computation from any linear secret-sharing scheme," in *Proceedings of the Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques*, pp. 316–334, Bruges, Belgium, 2000.

[5] G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs, "Multiparty computation with low communication, computation and interaction via threshold FHE," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 7237, pp. 483–501, 2012.

[6] D. Su, J. Cao, N. Li, E. Bertino, and H. Jin, "Differentially private K-Means clustering," in *Proceedings of the 6th ACM Conference on Data and Application Security and Privacy, CODASPY 2016*, pp. 26–37, USA, March 2016.

[7] D. Wang, H. Cheng, D. He, and P. Wang, "On the Challenges in Designing Identity-Based Privacy-Preserving Authentication Schemes for Mobile Devices," *IEEE Systems Journal*, pp. 1–10.

[8] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006*, pp. 89–98, Alexandria, Va, USA, November 2006.

[9] K. Yang, Z. Liu, X. Jia, and X. S. Shen, "Time-Domain Attribute-Based Access Control for Cloud-Based Video Content Sharing: A Cryptographic Approach," *IEEE Transactions on Multimedia*, vol. 18, no. 5, pp. 940–950, 2016.

[10] J. Shen, D. Liu, J. Shen, Q. Liu, and X. Sun, "A secure cloud-assisted urban data sharing framework for ubiquitous-cities," *Pervasive and Mobile Computing*, vol. 41, pp. 219–230, 2017.

[11] D. Wang, D. He, P. Wang, and C.-H. Chu, "Anonymous Two-Factor Authentication in Distributed Systems: Certain Goals Are Beyond Attainment," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 4, pp. 428–442, 2015.

[12] D. Wang and P. Wang, "Two Birds with One Stone: Two-Factor Authentication with Security Beyond Conventional Bound," *IEEE Transactions on Dependable and Secure Computing*, pp. 1-1.

[13] Q. Jiang, S. Zeadally, J. Ma, and D. He, "Lightweight three-factor authentication and key agreement protocol for internet-integrated wireless sensor networks," *IEEE Access*, vol. 5, pp. 3376–3392, 2017.

[14] Q. Jiang, Z. Chen, B. Li, J. Shen, L. Yang, and J. Ma, "Security analysis and improvement of bio-hashing based three-factor authentication scheme for telecare medical information systems," *Journal of Ambient Intelligence and Humanized Computing*.

[15] Q. Jiang, J. Ma, C. Yang, X. Ma, J. Shen, and S. A. Chaudhry, "Efficient end-to-end authentication protocol for wearable

health monitoring systems," *Computers Electrical Engineering*, 2017.

[16] J. Ni, K. Zhang, X. Lin, and X. Shen, "Balancing Security and Efficiency for Smart Metering against Misbehaving Collectors," *IEEE Transactions on Smart Grid*, pp. 1-1.

[17] J. Ni, K. Zhang, X. Lin, and X. Shen, "Securing fog computing for internet of things applications: Challenges and solutions," *IEEE Communications Surveys and Tutorials*, vol. 99, p. 1, 2017.

[18] J. Shen, S. Chang, J. Shen, Q. Liu, and X. Sun, "A lightweight multi-layer authentication protocol for wireless body area networks," *Future Generation Computer Systems*, 2016.

[19] C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC '09*, pp. 169–178, usa, June 2009.

[20] X. Liu, R. H. Deng, K.-K. R. Choo, and J. Weng, "An efficient privacy-preserving outsourced calculation toolkit with multiple keys," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2401–2414, 2016.

[21] X. Liu, B. Qin, R. H. Deng, R. Lu, and J. Ma, "A Privacy-Preserving Outsourced Functional Computation Framework Across Large-Scale Multiple Encrypted Domains," *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3567–3579, 2016.

[22] S. Yin, X. Li, H. Gao, and O. Kaynak, "Efficient and privacy-preserving outsourced calculation of rational numbers," *IEEE Transactions on Industrial Electronics*, vol. PP, no. 99, p. 1, 2014.

[23] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," in *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012*, pp. 1219–1234, New York, NY, USA, May 2012.

[24] S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Outsourced symmetric private information retrieval," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS 2013*, pp. 875–887, Berlin, Germany, November 2013.

[25] C. Hazay and H. Zarosim, "The feasibility of outsourced database search in the plain model," in *Proceedings of the Security and Cryptography for Networks - 10th International Conference*, pp. 313–332, Amalfi, Italy, 2016.

[26] S. Ji, W. Li, N. Z. Gong, P. Mittal, and R. Beyah, "On your social network de-anonymizablity: Quantification and large scale evaluation with seed knowledge," in *Proceedings of 22nd Annual Network and Distributed System Security Symposium, NDSS 2015*, San Diego, Calif, USA, 2015.

[27] C. Dwork, "Differential privacy: A survey of results," in *Proceedings of the Theory and Applications of Models of Computation, 5th International Conference, TAMC 2008*, pp. 1–19, Xi'an, China, 2008.

[28] H. Li, J. Cui, X. Lin, and J. Ma, "Improving the utility in differential private histogram publishing: Theoretical study and practice," in *Proceedings of the 4th IEEE International Conference on Big Data, Big Data 2016*, pp. 1100–1109, Washington DC, USA, 2016.

[29] G. Yuan, Z. Zhang, M. Winslett, X. Xiao, Y. Yang, and Z. Hao, "Low-rank mechanism: optimizing batch queries under differential privacy," *Proceedings of the VLDB Endowment (Very Large Database Endowment)*, vol. 5, no. 11, pp. 1352–1363, 2012.

[30] W. Qardaji, W. Yang, and N. Li, "PriView: Practical differentially private release of marginal contingency tables," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, SIGMOD 2014*, pp. 1435–1446, Snowbird, UT, June 2014.

[31] L. Zhang, Y. Liu, R. Wang, X. Fu, and Q. Lin, "Efficient privacy-preserving classification construction model with differential privacy technology," *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, Article ID 7870511, pp. 170–178, 2017.

[32] C. Dwork, A. Nikolov, and K. Talwar, "Efficient Algorithms for Privately Releasing Marginals via Convex Relaxations," *Discrete & Computational Geometry*, vol. 53, no. 3, pp. 650–673, 2015.

[33] M. Pettai and P. Laud, "Combining differential privacy and secure multiparty computation," in *Proceedings of the 31st Annual Computer Security Applications Conference, ACSAC 2015*, pp. 421–430, USA, December 2015.

[34] P. Mohan, A. Thakurta, E. Shi, D. Song, and D. Culler, "GUPT: Privacy preserving data analysis made easy," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, SIGMOD '12*, pp. 349–360, Scottsdale, AZ, USA, May 2012.

[35] V. Bindschaedler, S. Rane, A. E. Brito, V. Rao, and E. Uzun, "Achieving differential privacy in secure multiparty data aggregation protocols on star networks," in *Seventh ACM on Conference on Data and Application Security and Privacy, CODASPY 2017*, pp. 115–125, Scottsdale, AZ, USA, 2017.

[36] D. Stehlé and R. Steinfeld, "Making NTRU as secure as worst-case problems over ideal lattices," in *Proceedings of the 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, vol. 6632 of *Advances in Cryptology - EUROCRYPT 2011*, pp. 27–47, Tallinn, Estonia, 2011.

[37] A. Smith, "Privacy-preserving statistical estimation with optimal convergence rates," in *Proceedings of STOC 2011*, pp. 813–821, 2011.

[38] J. Groth, R. Ostrovsky, and A. Sahai, "New techniques for noninteractive zero-knowledge," *Journal of the ACM*, vol. 59, no. 3, pp. 1–11, 2012.

[39] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "Recursive composition and bootstrapping for SNARKs and proof-carrying data," in *Proceedings of the 45th Annual ACM Symposium on Theory of Computing, STOC 2013*, pp. 111–120, usa, June 2013.