Efficient and Secure Outsourcing of Differentially Private Data Publication

Jin Li[1,3](✉), Heng Ye[2], Wei Wang[2], Wenjing Lou[3], Y. Thomas Hou[4], Jiqiang Liu[2], and Rongxing Lu[5]

[1] School of Computer Science, Guangzhou University, Guangzhou, Guangdong, China
jinli71@gmail.com

[2] Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, 3 Shangyuancun, Beijing, China

[3] Department of Computer Science, Virginia Polytechnic Institute and State University, Falls Church, VA, USA

[4] Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA

[5] School of Computer Science, University of New Brunswick, Fredericton, NB, Canada

Abstract. While big data becomes a main impetus to the next generation of IT industry, big data privacy, as an unevadable topic in big data era, has received considerable attention in recent years. To deal with the privacy challenges, differential privacy has been widely discussed as one of the most popular privacy-enhancing techniques. However, with today's differential privacy techniques, it is impossible to generate a sanitized dataset that can suit different algorithms or applications regardless of the privacy budget. In other words, in order to adapt to various applications and privacy budgets, different kinds of noises have to be added, which inevitably incur enormous costs for both communication and storage. To address the above challenges, in this paper, we propose a novel scheme for outsourcing differential privacy in cloud computing, where an additive homomorphic encryption (e.g., Paillier encryption) is employed to compute noise for differential privacy by cloud servers to boost efficiency. The proposed scheme allows data providers to outsource their dataset sanitization procedure to cloud service providers with a low communication cost. In addition, the data providers can go offline after uploading their datasets and noise parameters, which is one of the critical requirements for a practical system. We present a detailed theoretical analysis of our proposed scheme, including proofs of differential privacy and security. Moreover, we also report an experimental evaluation on real UCI datasets, which confirms the effectiveness of the proposed scheme.

1 Introduction

There is a general consensus that we are currently in the era of big data, with tremendous amounts of information being collected by various organizations

J. Lopez et al. (Eds.): ESORICS 2018, LNCS 11099, pp. 187–206, 2018.
https://doi.org/10.1007/978-3-319-98989-1_10

and entities. Often, these data providers may wish to contribute their data to studies involving tasks such as statistical analysis, classification, and prediction. Because cloud service providers (CSPs) offer data providers (owners) great flexibility with respect to computation and storage capabilities, CSPs are presently the most popular avenue, through which data providers can share their data. However, the risk of leaking individuals' private information with the straightforward uploading of data providers' data (which may contain sensitive information such as medical or financial records, addresses and telephone numbers, or preferences of various kinds that the individuals may not want exposed) to CSPs is unacceptable. Usually, data providers protect their data's privacy by using of encryption, but the resulting encrypted data are known to be of poor utility. As an alternative, differential privacy (DP) has been considered as a useful technique not only for protecting the privacy of data but also for boosting the utility of data. As shown in Fig. 1, there are generally two main frameworks that can be used for achieving DP, i.e., the framework with interaction for release, and the framework with no interaction for publication. In this paper, we will focus on the latter, i.e., differentially private publication.

With current techniques, there does not exist an efficient method that allows data to be sanitized only once while still preserving the data's utility for all possible algorithms and applications. When data providers' data need to be shared for uses involving different algorithms, applications, or privacy budgets, different kinds of noise have to be added for privacy protection. Moreover, all of these different noisy datasets must be shared or published. Consequently, the communication overhead will be enormous if the number of different algorithms/applications/privacy budgets is large. Another challenging issue is that when data providers publish their data, public entities must exist somewhere that can store all of the different kinds of datasets with different types of noise, which also inevitably requires considerable storage space.

To overcome these challenges, we propose the notion of outsourcing differentially private data publication in this paper. With the advent of cloud computing, we know an increasing number of storage and computing tasks are moving from local resources to CSPs. In our scheme, the sanitized dataset generation procedure is outsourced to a CSP by the data providers. To protect the privacy of the data, we can use the effective method, i.e., cryptographic techniques, to encrypt the data before outsourcing. However, data sanitization requires the ability to easily modify the data to be sanitized, while encrypted data generally cannot provide such an ability. Using fully homomorphic encryption technique to support ciphertext manipulation is however inefficient and requires enormous amounts of storage space and communication bandwidth. Unlike other schemes, our differentially private data publication requires only addition operations, and such operations can be realized by using the additive homomorphic encryption, which would be much more efficient than the fully homomorphic encryption. Although this approach enables us to add noise to encrypted data, encryption still leads to poor utility or heavy consume of storage/computation in many respects. Therefore, in our scheme, it allows the data evaluator to decrypt the encrypted noisy data, thereby improving the utility of the data as well as the storage cost. In this way, the data providers are not required to be online when their data are requested, which is one of the critical requirements for a practical application system.

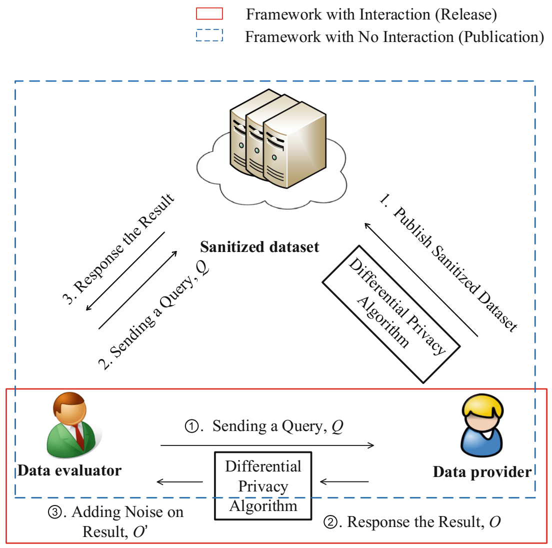

Framework with Interaction (Release)
Framework with No Interaction (Publication)
Sanitized dataset
3. Response the Result
2. Sending a Query, Q
1. Publish Sanitized Dataset
Differential Privacy Algorithm
Data evaluator
①. Sending a Query, Q
Data provider
Differential Privacy Algorithm
③. Adding Noise on Result, O'
②. Response the Result, O

Fig. 1. Frameworks for achieving differential privacy

1.1 Related Work

Differential privacy has been accepted as the main privacy paradigm in recent years, because it is based on purely mathematical calculations and provides a means of quantitative assessment. A large body of work on DP has accumulated due to its support for privacy preserving learning. There are some works which used the cryptographic methods to solve the privacy preserving for data utilization [32,33] before the first work of DP by Dwork in 2006 [9], and the Laplace mechanism for adding noise to achieve ϵ-DP was proposed in the seminal paper. Subsequently, McSherry designed the exponent mechanism [25] and identified the sequential and parallel properties of DP [24].

Generally, DP can be achieved via two main frameworks. In the first framework, the data evaluator's queries are responsed under a predetermined privacy budget ϵ; as shown in Fig. 1 with red box. In the framework, we can apply the Laplace [11], Privlet [35], Linear Query [21], and Batch Query [38] techniques, among others, to obtain different responses to these queries that satisfy ϵ-DP. However, this framework demands interaction between the data provider and the data evaluator. For this reason, in this paper, we will focus on the second framework, as depicted in Fig. 1 with the blue dashed box. The second framework not only allows a data provider to immediately publish his data after processing but also does not require any interaction. The main focus of research related to this framework is on how to design efficient and effective noise-adding algorithms to ensure DP while boosting data utility.

Typical publication methods include histogram publication [15,17,36,37], partition publication [5,7,26,28], contingency table publication [1,20], and sanitized dataset publication [8,10,12,14]. A histogram is an intuitive representation of the distribution of a set of data and can be used as a basis for other statistical queries or linear queries. However, histogram publication suffers from problems of redundant noise and inconsistency, meaning that different types of noise should be added for different uses. Partition publication can reduce the amount of noise that must be added. The foundation of partition publication is the careful design of an index structure to support the partitioning of the data. Using this index, the data provider can assign a privacy budget to each partition for noise addition before publication. However, determining how to assign a privacy budget is not a trivial task, and the partition index itself may leak some sensitive information; this potential risk is the core problem that remains to be solved for this method of publication. Often, data can be represented in the form of a contingency table. In fact, instead of publishing the contingency table itself for analysis, data are often published based on the statistical values of the combinations of certain variables, as represented by marginal tables. Directly adding noise to a contingency table introduces too much noise, whereas perturbing the marginal table may cause inconsistency. Qardaji proposed a noise-adding method in which the contingency table is divided into small pieces, called views [29]. This method can reduce the amount of noise introduced, but the questions of how to choose the parameters to be used for division and how to preserve the consistency between the views and the marginal tables remain challenging. The purpose of sanitized dataset publication is to ensure the protection of data privacy after the processing of the original dataset. The question on how to directly publish a sanitized dataset that satisfies DP while allowing the data evaluator to make any necessary inquiries is quite challenging. This method of dataset publication demands considerable calculation and thus is inefficient and difficult to realize. Kasiviswanathan and Blum proved that sanitized dataset publication is possible [3,19]; however, it requires an enormous number of records.

Although the DP model provides frameworks for data evaluators to analyze databases belonging to a single party, the initial frameworks did not consider a multi-party setting. Multi-party DP was first proposed by Manas in 2010 [27], based on the aggregation of multi-party datasets to train a classifier. Subsequently, many works involving multi-party DP publication have been reported, including multi-task learning [13], multi-party deep learning [30], classifier training on private and public datasets [18] and high-dimensional data publication [31]. These works, however, have not considered outsourced computing to relieve the computational burden on data providers.

Outsourced computing is a technique for securely outsourcing expensive computations to untrusted servers, which allows resource-constrained data providers to outsource their computational workloads to cloud servers with unlimited computational resources. Chaum first proposed the notion of wallets, secure hardware installed on a client's computer to perform expensive computations, in 1992 [4]. To protect data providers' privacy, Chevallier presented the first algorithm for the secure delegation of elliptic-curve pairings [6]. In addition, solutions for performing meaningful computations over encrypted data using fully homomorphic encryption have emerged, although they are known to be of poor practicality [2]. Meanwhile, cloud computing using attribute-based encryption appeared, which not only supports ciphertext operation, but also provides fine-grained access control. Some of works that concentrate on privacy preserving in cloud computing have been proposed recently [16,22,23]. Nevertheless, in this paper, we choose additive homomorphic encryption as our basic technique to perform outsourcing computing, which offers a perfect balance of efficiency and security.

1.2 Contributions

In this paper, we propose a novel outsourced DP scheme for cloud computing. Our contributions can be summarized as follows.

- We design an efficient outsourced DP approach using additive homomorphic encryption instead of fully homomorphic encryption. In such a way, data can be efficiently outsourced to a CSP for secure storage and DP supporting noise addition.
- In our scheme, the data provider is not required to be involved in subsequent noise computations and related processing.
- The security of the data against the CSP can be guaranteed under our proposed security model.

1.3 Organization

The rest of this paper is organized as follows. Some preliminary considerations are discussed in Sect. 2. In Sect. 3, the architecture of our scheme and our threat model are introduced. Then, we present the new scheme in Sect. 4. The implementation details and the evaluation of the experimental results are presented in Sect. 5. Finally, we conclude our work in Sect. 6. Also, the security analysis and some related concepts using in this paper are put in the appendix.

2 Preliminaries

2.1 Differential Privacy

DP was introduced by Dwork et al. as a technique for individual privacy protection in data publication. It provides a strong privacy guarantee, ensuring that the presence or absence of an individual will not significantly affect the final output of any function (Table 1).

Table 1. Symbol cross reference table

ϵ	Privacy budget	f	Algorithm for machine learning
m	Plaintext	c	Ciphertext
P	Data provider	$\|m\|$	Ciphertext of m
η	Noise	Δf	Global sensitivity
b	Parameters for distribution	sk_P/pk_P	Secret/public key of P
Sth	Vector of *sth*		

Definition 1. *(Differential privacy)*
A randomized function $\mathcal{A}$ with a well-defined probability density $\mathcal{P}$ satisfies ϵ-DP if, for any two neighboring datasets D_1 and D_2 that differ by only one record and for any $\mathcal{O} \in range(M)$,

$$\mathcal{P}(\mathcal{A}(D_1) = \mathcal{O}) \leq e^{\epsilon} \cdot \mathcal{P}(\mathcal{A}(D_2) = \mathcal{O}) \tag{1}$$

DP can usually be achieved via one of two standard mechanisms: the Laplace mechanism and the exponential mechanism. Both of them are based on the concept of the sensitivity of a function f. For ease of description, we consider only numeric values in this paper.

Definition 2. *(Global sensitivity)*
Let f be a function that maps a database to a fixed-size vector of real numbers. For all neighboring databases D_l and D_2, the global sensitivity of f is defined as

$$\Delta(f) = \max_{D_1, D_2} \|f(D_1) - f(D_2)\|_1 \tag{2}$$

where $\|\cdot\|_1$ denotes the L_1 norm.

To publish data that satisfy ϵ-DP when a query function f is applied, the principal approach is to perturb the data by adding random noise based on Δf and the privacy budget ϵ. For example, for the Laplace mechanism, let Lap(λ) denote the Laplace probability distribution with mean zero and scale λ. The Laplace mechanism achieves DP by adding Laplace noise to an original dataset M.

Definition 3. *(Laplace mechanism)*
Let m be a record in database M ($m \in M$), and let η be a random variable such that $\eta \sim Lap(\Delta f/\epsilon)$. The Laplace mechanism is defined as follows:

$$m' = m + \eta.(\Sigma m = M) \tag{3}$$

3 Architecture

In this section, we formalize our system model, and identify the threat model and security requirements.

3.1 System Model

In our system model, four types of entities are involved in our basic scheme, namely, the trusted authority (TA), the data providers, the cloud service provider (CSP), and the data evaluator. The TA issues credentials for both data providers and data evaluators. The data providers possess data and would like to share those data for purposes such as classification or data analysis. The CSP provides the cloud storage service for the data providers. The data evaluator obtains the sanitized data from the CSP and performs the corresponding data analysis. Each data evaluator may acquire different part of dataset for different usage. However, the same data he got, the same noise the data has.

3.2 Threat Model and Security Requirements

In this work, both the data evaluator and the CSP are assumed to be *honest-but-curious*. The data evaluator needs to protect his trained model against the CSP. Moreover, the data evaluator will follow a specified protocol for building a correct model without obtaining incorrect results. The relationship between CSP and data evaluator is non-cooperative, which means they will not collude with each other (For example, CSP could be Google Inc. and data evaluator is Apple Inc., which is very common in our daily life). Otherwise, even they are honest, data evaluator could give CSP his secret key to get the original dataset. The privacy of the data providers' data needs to be protected against both the CSP and the data evaluator.

For the data owner, the security means that its privacy should be protected against the other entities even if they are curious about the underlying original data. Of course, the CSP and DP are not allowed to collude with each other in our security model.

4 Our Proposed Outsourced Differential Privacy Schemes

In this section, we present several basic outsourced differential privacy (ODP) schemes. During the description, we consider a public key encryption scheme with additive homomorphic properties, i.e., $(Setup, KeyGen, Enc, Dec, Add)$, will be applied in our system.

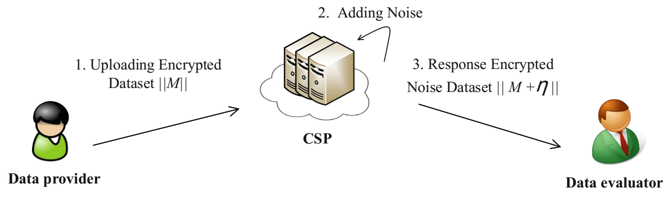

2. Adding Noise
1. Uploading Encrypted Dataset $||M||$
3. Response Encrypted Noise Dataset $|| M+\eta ||$
CSP
Data provider
Data evaluator

Fig. 2. Single-data-provider scheme

4.1 A Straightforward Scheme

In a straightforward scheme, the typical process of differentially private data publication can be summarized as follows:

- **Setup**. The data evaluator and data provider together establish the privacy budget ϵ.
- **Generation**. The data provider calculates Δf and then uses Δf and ϵ to generate a noisy dataset.
- **Uploading**. The data provider uploads the noisy dataset to the CSP.
- **Analysis**. The data evaluator obtains the noisy dataset from the CSP and analyzes this dataset.

Note that, for different data evaluators and/or different privacy budgets, the above steps should be executed repeatedly, which means more power and bandwidth will be consumed.

To overcome the disadvantages of the above scheme, we show how to provide efficient noise addition with the aid of cloud servers in next sections.

4.2 Our Scheme for Single Data Provider

Initially, we assume that there is a single data provider in the system (Fig. 2). Then, the single-data-provider scheme, as shown in Algorithm 1, is composed of the following steps:

- **Setup**. In this step, the TA generates the public parameters and key pairs involved in the system. Let (pk, sk) be the key pair for the data evaluator, where pk is the public key and sk is the secret key. Note that, the data provider should pre-calculate the possible set of function sensitivities, denoted by $\Delta \mathbf{F} = (\Delta f_1, \Delta f_2, \cdots, \Delta f_m)$, where f_i $(i = 1, \cdots, m)$ are the functions that the data evaluator might use in his evaluations.
- **Data uploading**. The data provider first receives the data evaluator's public key pk from the TA, encrypts his dataset $M = (m_1, m_2, \cdots, m_n)$ using the $Enc(pk, M)$ algorithm, and uploads the resulting ciphertexts $C = (\|m_1\|, \|m_2\|, \cdots, \|m_n\|)$ to the cloud server. Then, the data provider (and perhaps the data evaluator) determines the privacy budget ϵ. Furthermore, the parameter vector $\mathbf{b}$ for noise generation (e.g., for the Laplace mechanism, since the Laplace noise generation depends on $\frac{\Delta \mathbf{F}}{\epsilon}$, $\mathbf{b} = (b_1, b_2, \cdots, b_m) = (\frac{\Delta f_1}{\epsilon}, \frac{\Delta f_2}{\epsilon}, \cdots, \frac{\Delta f_m}{\epsilon})$) is also sent to the cloud server.
- **Noise addition**. After receiving the data from the data provider, the cloud server generates a noise component η (for example, in the Laplace noise mechanism, $b_i = \frac{\Delta f_i}{\epsilon}$ is used as the parameters to define the Laplace distributions from which to randomly draw noise) and encrypts the noise using $Enc(pk, \eta) = \|\eta\|$. Then, the cloud server uses the $Add(\|M\|, \|\eta\|)$ algorithm to add the noise to the data provider's data (for example, in the case of Paillier encryption, the server multiplies the data provider's encrypted data by the encrypted noise) and sends the resulting noisy data to the data evaluator.

– **Data analysis**. The data evaluator first decrypts the received ciphertexts using $Dec(sk, \|M + \eta\|)$ to obtain all of the noisy data. Based on these data, the data evaluator can successfully perform classification or apply other algorithms to the data.

Algorithm 1. Single data provider scheme for noise addition

```
Input: Data provider (D): clean dataset
    $M = (m_1, m_2, \cdots, m_n)$.
    Data evaluator (DE): possible functions to be used
    $(f_1, f_2, \cdots, f_m)$.
Output: DE: dataset with noise
    $M + \eta$.
1: D: $\Delta\mathbf{F} = (\Delta f_1, \Delta f_2, \cdots, \Delta f_m)$, calculates the sensitivities of the possible functions;
2: D: $\epsilon \leftarrow DE$, communicates with DE to establish an appropriate privacy budget;
3: D: $\mathbf{b} = \Delta\mathbf{F}/\epsilon$, calculates the parameter vector;
4: D: $\{\|M\|, \mathbf{b}\} \rightarrow CSP$, uploads the encrypted dataset and parameter vector;
5: D: $i_0 \leftarrow DE$, obtains which functions DE will use;
6: D: $i_0 \rightarrow CSP$, uploads the number of functions to be used;
7: for each $j \in [1, \cdots, n]$ do
8:     CSP: $\eta_j \sim Lap(b_{i_0})$, generates noise;
9:     CSP: $\|\eta_j\|_{pk} = Enc(\eta_j, pk)$, encrypts the noise;
10:    CSP: $\|m_j + \eta_j\|_{pk} = Add(\|m_j\|, \|\eta_j\|)$, calculates the noisy dataset;
11: end for
12: CSP: sends $\|M + \eta\|$ to DE;
13: DE: $M + \eta = Dec(\|M + \eta\|, sk)$, decrypts the ciphertexts;
14: return $M + \eta$.
```

4.3 A Multi-Data-Provider Scheme

Next, we present the multi-data-provider scheme (similar with Fig. 2, but more data providers). For the ease of description, we assume that each party holds a parallel data set, meaning that there is no overlap between any two parties' databases. Each party P_i hold a portion of D, denoted by D_i, such that $\sum D_i = D$. The scheme, as shown in Algorithm 2, consists of four steps.

– **Setup**. This step is similar to that for a single data provider. The difference is that there are k parties, denoted by $P_1, P_2, \cdots, P_k$, and each user P_i should pre-calculate a set of possible function sensitivities $\Delta\mathbf{F}_i$.
– **Data uploading**. Each data provider P_i encrypts the dataset $M_i = (m_{i1}, m_{i2}, \cdots, m_{in})$ using the $Enc(pk, M_i)$ algorithm. After the encryption, P_i uploads the ciphertexts $C_i = (\|m_{i1}\|, \|m_{i2}\|, \cdots, \|m_{in}\|)$ to the cloud

server. Then, the users determine the privacy budget ϵ. The parameter vectors $\mathbf{b}_i$ for noise generation are also sent to the cloud server.

- **Noise addition**. Using all the noise parameter vectors $\mathbf{b}_i$ $(i = 1, 2, \cdots, k)$, the cloud server generates the noise η_i for each P_i and encrypts the noise using $Enc(pk, \eta_i) = \|\eta_i\|$. Then, the cloud server uses the $Add(\|M_i\|, \|\eta_i\|)$ algorithm to add the noise to the data providers' data and sends the noisy data to the data evaluator.
- **Data analysis**. This step is the same as that for a single data provider.

Algorithm 2. Multiple-data-provider scheme for noise addition

```
Input: Data providers (D): clean dataset
    $\mathbf{M} = (M_1, M_2, \cdots, M_k)$.
    Data evaluator (DE): possible functions to be used
    $(f_1, f_2, \cdots, f_m)$.
Output: DE: dataset with noise
    $M + \eta$.
1: D: $\Delta\mathbf{F} = (\Delta\mathbf{F}_1, \Delta\mathbf{F}_2, \cdots, \Delta\mathbf{F}_k)$;
2: D: $\epsilon \leftarrow DE$;
3: D: $\mathbf{b} = \Delta\mathbf{F}/\epsilon$;
4: D: $\{\|M\|, \mathbf{b}\} \rightarrow CSP$;
5: CSP: generate appropriate noise $\eta$ under DE's require;
6: CSP: sends $\|M + \eta\|$ to DE;
7: DE: $M + \eta = Dec(\|M + \eta\|, sk)$;
8: return $M + \eta$.
```

4.4 Discussion: How to Add Noise Wisely and Efficiently

In the above schemes, we have assumed that the noise generation is a simple task performed by the CSP. Actually, other noise-adding schemes from other entities are also possible. In the following, we focus on discussing additional methods of noise generation for the single-data-provider schemes. Note that, similar methods can also be applied in the multi-data-provider schemes.

Noise Addition by the Data Provider. Noise can be generated by the data provider; see Fig. 3. In this method, the CSP will have no interaction with the parameter vector $\mathbf{b}$ or the generated noise η. The details of the procedure are shown in Algorithm 3.

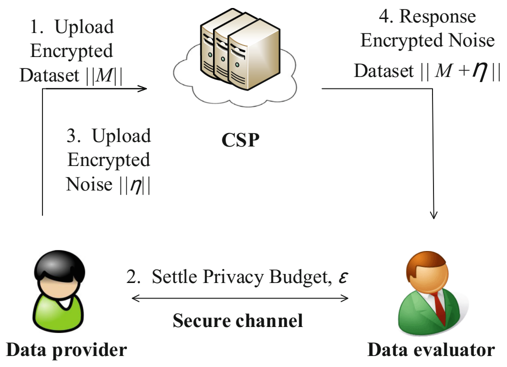

1. Upload Encrypted Dataset ||M||
4. Response Encrypted Noise Dataset || M + η ||
3. Upload Encrypted Noise ||η||
CSP
2. Settle Privacy Budget, ε
Secure channel
Data provider
Data evaluator

Fig. 3. Noise addition by the data provider

Algorithm 3. Noise addition by the data provider

```
Input: Data provider (D): clean dataset
    $M = (m_1, m_2, \cdots, m_n)$.
    Data evaluator (DE): function to be used $f$.
Output: DE: dataset with noise
    $M + \eta$.
1: D: $\Delta f = \max_{M,M'} \|f(M) - f(M')\|_1$
2: D: $\epsilon \leftarrow DE$;
3: D: $b = \Delta f / \epsilon$;
4: D: $\|M\| \rightarrow CSP$;
5: D: sends the new generated noise in encrypted form $\|\eta\|_{pk}$ to the CSP;
6: CSP: $\|M + \eta\|_{pk} = Add(\|M\|, \|\eta\|)$;
7: DE: $M + \eta = Dec(\|M + \eta\|, sk)$;
8: return $M + \eta$.
```

Noise Addition by a Noise Server. Noise addition by the data provider can protect some sensitive parameters from the CSP. However, this scheme does not allow the data provider to go offline. Therefore, we propose a scheme with noise addition by a noise-generating server; see Fig. 4. We assume that there is another server, called the noise-generating server, in our system, i.e., the system now contains five types of entities: the trusted authority (TA), the data provider, the cloud service provider (CSP), the data evaluator and the noise-generating service provider (NSP). The main idea of this design is to separate the noise generation from the rest of the system. The details of the procedure are shown in Algorithm 4.

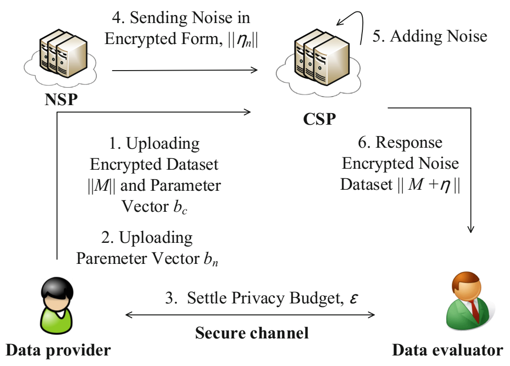

4. Sending Noise in Encrypted Form, $||\eta_n||$
5. Adding Noise
NSP
CSP
1. Uploading Encrypted Dataset $||M||$ and Parameter Vector b_c
2. Uploading Paremeter Vector b_n
6. Response Encrypted Noise Dataset $|| M+\eta ||$
3. Settle Privacy Budget, ε
Secure channel
Data provider
Data evaluator

Fig. 4. Noise addition by a noise server

Algorithm 4. Noise addition by a noise server

```
Input: Data provider (D): clean dataset
    $M = (m_1, m_2, \cdots, m_n)$.
    Data evaluator (DE): possible functions to be used
    $(f_1, f_2, \cdots, f_m)$.
Output: DE: dataset with noise
    $M + \eta$.
 1: D: $\mathbf{\Delta F} = (\Delta f_1, \Delta f_2, \cdots, \Delta f_m)$;
 2: D: $\epsilon \leftarrow DE$;
 3: D: $\epsilon = \epsilon_n + \epsilon_c$, divides privacy budget into two parts;
 4: D: $\mathbf{b}_n = \mathbf{\Delta F}/\epsilon_n$, $\mathbf{b}_c = \mathbf{\Delta F}/\epsilon_c$;
 5: D: $\mathbf{b}_n \rightarrow NSP$, uploads a parameter vector to the NSP;
 6: D: $\{\|M\|, \mathbf{b}_c\} \rightarrow CSP$;
 7: NSP: sends encrypted noise $\|\eta_n\|$ to the CSP;
 8: CSP: $\|m_j + \eta_j\|_{pk} = Add(\|m_j\|, \|\eta_{nj}\|, \|\eta_{cj}\|)$;
 9: DE: $M + \eta = Dec(\|M + \eta\|, sk)$;
10: return $M + \eta$.
```

4.5 Discussion: How to Treat High-Dimensional Data

The publication of high-dimensional data can support a wide spectrum of evaluation tasks. However, the problem of how to ensure the privacy of high-dimensional data is still challenging. Generally speaking, we can handle high-dimensional data as described below.

View-Based Dimension Reduction. High-dimensional data can be presented in the form of a contingency table [29]. For most types of data evaluations, a k-way marginal contingency table (generally, $k \leq 3$) can be used. However, directly adding noise to the complete contingency table (all of the data) to achieve data sanitization will lead to excessive noise. Therefore, determining how to add noise to obtain a high-utility k-way marginal table is the key to high-dimensional data publication. We use the following basic procedure to reduce the dimensionality of high-dimensional data before publication.

- **Build a contingency table**. In this step, the data provider builds a contingency table based on his high-dimensional dataset.
- **Generate views**. The data provider runs an algorithm to select the best parameters with which to generate views. (For example, suppose that we have an l-column, d-dimensional table, where $d = 8$ and $l = 100$; then, the data provider runs the algorithm to generate $d^2 + l$ sub-contingency tables, which are referred to as views.) The data provider can treat these views in the same way as his original data, following the scheme proposed above (setup, data uploading, noise generation, noise addition). A bloom filter should also be used to assign the attributes to the encrypted views.
- **Reconstruct**. The noisy k-way marginal table is reconstructed based on the encrypted views, and the bloom filter is used to check whether the views can be successfully used to reconstruct the k-way marginal table.

4.6 Discussion: How to Make the Scheme More Practical

In the schemes presented above, each data provider uses the data evaluator's public key to encrypt his own data, which means that the data provider cannot delete his local copy of his dataset to save storage space, because using the data evaluator's public key for encryption causes the data provider to lose the ability to decrypt the dataset stored by the CSP. Inspired by Wang's method [34], we propose our ODP scheme with proxy re-encryption technique; the details of the procedure are shown in Algorithm 5.

Algorithm 5. Using proxy re-encryption during uploading in multi-data-provider scheme

```
Input: Data providers (D): clean dataset
    $M = (M_1, M_2, \cdots, M_k)$.
    Data evaluator (DE): possible functions to be used
    $(f_1, f_2, \cdots, f_m)$.
Output: DE: dataset with noise
    $M + \eta$.
1: D: $\boldsymbol{\Delta F} = (\boldsymbol{\Delta F}_1, \boldsymbol{\Delta F}_2, \cdots, \boldsymbol{\Delta F}_k)$;
2: D: $\epsilon \leftarrow DE$;
3: D: $\mathbf{b} = \boldsymbol{\Delta F}/\epsilon$;
4: D: $\{\|M\|, \mathbf{b}\} \rightarrow CSP$;
5: CSP: $(rkey_{P_i} \rightarrow pk_{p_e}) \leftarrow$TA;
6: CSP: $\|M_i + \eta_i\|_{pk_{P_e}} = ReEnc(rkey_{P_i} \rightarrow pk_{p_e}, \|M_i + \eta_i\|_{pk_{P_i}})$;
7: DE: $M + \eta = Dec(\|M + \eta\|, sk_{P_e})$, decrypts the ciphertexts;
8: return $M + \eta$.
```

5 Evaluation

In this section, we evaluate the performance of our scheme in terms of functionality, computational overhead and communication overhead. All experiments were conducted on a PC with a 1.90 GHz AMD A4-3300M APU with Radeon(TM) HD Graphics and 6 GB of RAM. Also, using different kinds of noise addition method only differs nothing but in execution time during noise generation. For simplicity, we choose Laplace mechanism in this evaluation to prove our scheme's feasibility.

5.1 Functionality

We used datasets acquired from the UCI machine learning repository, which can be downloaded from UCI[1], to evaluate our scheme's functionality. To evaluate the classifier performance, we reserved $\frac{1}{10}$ of each dataset to serve as a test dataset, and we chose $\epsilon = 0.1$ for applying the Laplace mechanism to our datasets. Figure 5 shows the accuracies of training KNN and Naive Bayes classifiers for Letter Recognition, EEG Eye State, CPU, and Glass. From the figure, we can see that training a classifier on the sanitized dataset instead of the original dataset exerts little influence on the classifier performance.

5.2 Computational Overhead

We use the Paillier scheme as our homomorphic encryption algorithm. To perform homomorphic addition on ciphertexts, we should treat each attribute as a plaintext input and calculate its related ciphertext. Thus, the total number of encryption operations is counts = records * attributes.

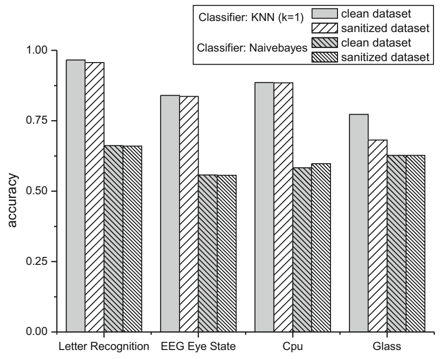


Fig. 5. Functionality

[1] http://archive.ics.uci.edu/ml/.

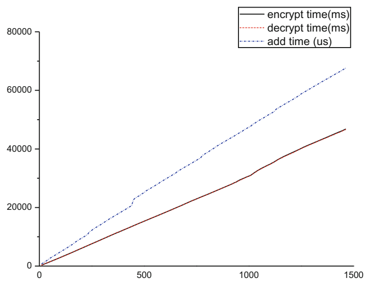


Fig. 6. Computational overhead

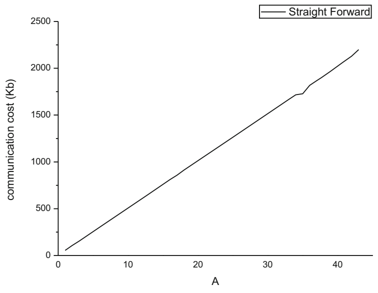


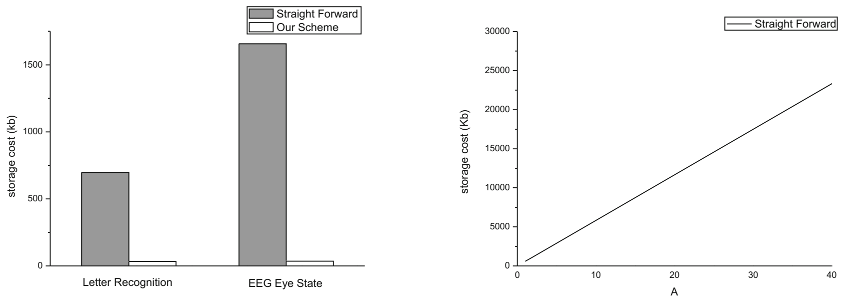
Fig. 7. Communication overhead

As seen in Fig. 6, it takes approximately 30 s to perform 1000 encryption/decryption operations but only 48 ms to perform 1000 addition operations on ciphertexts. In our scheme, data providers can pre-upload their data to the CSP, and the encryption can be executed in the data providers' spare time. Because the cost of ciphertext addition is low, our scheme is acceptable and feasible.

5.3 Communication Overhead

There are two phases that incur main communication costs, including data uploading and data analysis.

In the data uploading phase (see Fig. 7), the size of the message sent in our scheme is $ndc + \frac{k}{1000}$ kb, whereas the message size in the straightforward scheme is $kndp$ kb, where n, d, c, k, and p denote the size of the dataset, the number of attributes, the size of each ciphertext record, the number of types of noise to be added and the size of each plaintext record, respectively. As more different kinds of noise are added, the communication cost grows linearly in the straightforward scheme. Meanwhile, in our scheme, the size of the dataset increases to 88 kb after encryption when the original dataset size is 43 kb, indicating that encryption leads to a $\times 2$ increase. Once there is more than one type of noise to be added, our scheme ensures a lower communication cost. In the data analysis phase, the total size of the message sent by the CSP is $kndc$, compared with $kndp$ in the straightforward scheme. Generally speaking, therefore, the size of the message sent to the data evaluator in our scheme is approximately twice as large as that in the straightforward scheme. However, as shown in Fig. 8, the storage cost of our scheme is cheaper for both each data provider and the CSP. **Storage cost for a data provider** (see Fig. 8(a)): In our scheme, because the original dataset is uploaded, the data provider is required to store only the secret key for Paillier encryption (approximately 32 kb), which can be used to regain his data. By contrast, in the straightforward scheme, because of the lack of encryption (storage in plaintext poses privacy concerns), the data provider cannot reduce his storage cost for the original dataset. **Storage cost for the CSP** (see Fig. 8(b)): In our scheme, the CSP can store only one copy of the



(a) Data Provider's Storage Cost (b) CSP's Storage Cost

Fig. 8. Storage costs

original dataset in encrypted form and regenerate a noisy dataset every time a data evaluator requests it. However, to preserve confidentiality, the CSP cannot possess the unencrypted original dataset with which to generate noisy datasets using the straightforward scheme; therefore, in this scheme, the CSP must store every noisy dataset, which will lead to a linear increase in storage cost with the number of types of noise to be added.

6 Conclusions

In this paper, we addressed the issues of inefficiency due to adding different types of noise to a dataset for differentially private publication. Specifically, to solve the challenge, we proposed an efficient and secure outsourced differential privacy scheme suitable for all DP algorithms with noise addition operations, including Laplace algorithm, in which the data providers incur reduced communication costs and are not required to be online when the access to their data is requested. We also showed how to use an independant noise server to deal with the differential privacy. The differential privacy for high-dimension data was also discussed. Finally, the experiment showed that the efficiency of our new scheme compared with the basic solutions. In future work, we will consider different kinds of noise-adding methods that can be executed over encrypted data. In addition, we are also interested in researching special encryption algorithms that permit smart and efficient noise addition.

Acknowledgement. This work was supported by Natural Science Foundation of Guangdong Province for Distinguished Young Scholars (2014A030306020), National Natural Science Foundation of China (No. 61472091) and National Natural Science Foundation for Outstanding Youth Foundation (No. 61722203). This work was also supported in part by US National Science Foundation under grants CNS-1446478 and CNS-1443889.

Appendix A: Cryptographic Tools

Additive Homomorphic Encryption. We say that an encryption scheme is additive homomorphic if it conforms to the following definition.

Definition 4. *(Additive homomorphic encryption)*
Let m_1 and m_2 be two plaintexts, let $\mathcal{A}$ be an encryption algorithm that outputs the corresponding ciphertexts $\|m_1\|$ and $\|m_2\|$, and let $\mathcal{B}$ be an operation performed on the two ciphertexts. For any two ciphertexts, additive homomorphic encryption has the following property:

$$\mathcal{B}(\|m_1\|, \|m_2\|) = \mathcal{B}(\mathcal{A}(m_1), \mathcal{A}(m_2)) = \|m_1 + m_2\| \quad (4)$$

In this paper, we discuss an encryption scheme that operates under a public key encryption system and consists of the following algorithms:

- **Setup**(1^l): A trusted authority (TA) uses a security parameter l to generate the public parameters (pp) and the main secret key (msk).
- **KeyGen**(msk, pp, uid): The TA uses a user's identity uid as input to generate a pair of keys (pk, sk) for that user.
- **Enc**(pk, m): The user uses his public key pk to encrypt a plaintext record m, generating the ciphertext $\|m\|$ as output.
- **Dec**$(sk, \|m\|)$: The user uses his secret key sk to decrypt a ciphertext record $\|m\|$ into the corresponding plaintext m.
- **Add**$(\|m_1\|,\|m_2\|)$: Two ciphertexts $\|m_1\|$ and $\|m_2\|$ are inputs, and the result $\|m_1 + m_2\|$ is output.

Due to the simplicity of the Paillier encryption scheme, we choose the Paillier scheme as our additive homomorphic encryption algorithm.

Proxy Re-encryption. Proxy re-encryption is based on the concept that an honest-but-curious proxy uses a re-encryption key to translate a ciphertext encrypted with the data owner's public key into another ciphertext that can be decrypted using another user's private key. The general structure of the proxy re-encryption process can be summarized as follows:

- **Setup**(1^l): The TA uses a security parameter l to generate the public parameters (pp) and the main secret key (msk).
- **KeyGen**(msk, pp, I_{de}): Using a data evaluator's identity I_{de}, msk and pp as input, the TA generates the data evaluator's keys $(pk_{I_{de}}, sk_{I_{de}})$.
- **PrivKeyGen**(pp, uid): The TA generates a pair of keys (pk_{uid}, sk_{uid}) for a user (a data owner) using that user's identity uid and pp.
- **ReKeyGen**$(sk_{uid}, pk_{I_{de}})$: The TA outputs the re-encryption key $rkey_{uid} \rightarrow pk_{I_{de}}$ using the data owner's secret key sk_{uid} and the data evaluator's public key $pk_{I_{de}}$.
- **Enc**(pk_{uid}, m): The data owner uses his public key pk_{uid} to encrypt a plaintext input m, generating the ciphertext $\|m\|$ as output.

- **ReEnc**($rkey_{uid} \rightarrow pk_{I_{de}}, \|m\|$): A cloud server calculates and outputs the re-encrypted ciphertext $\|m\|^R$, using as input the ciphertext $\|m\|$ and the re-encryption key $rkey_{uid} \rightarrow pk_{I_{de}}$.
- **Dec**($sk_{uid}, \|m\|$): The data owner uses his secret key sk_{uid} to decrypt the ciphertext $\|m\|$ into the plaintext m.
- **Dec**($sk_{I_{de}}, \|m\|^R$): The data evaluator uses his secret key $sk_{I_{de}}$ to decrypt the re-encrypted ciphertext $\|m\|^R$ to obtain the plaintext m.

Appendix B: Security Analysis

Clearly, our goals are to protect the data providers' data from any adversary (ADV). In this section, we only present the security proof about data transmission process and noise addition process. Note that, for the data storage process and decryption process, since they depend on the encryption algorithm applied, we will not discuss the security for the two processes here.

According to Fig. 2, the data flows are directed from the data provider to the CSP and from the CSP to the data evaluator. Generally speaking, an ADV may eavesdrop on the communication flow from the data provider to the CSP. However, even if the ADV obtains a message in this way he cannot learn anything from this message without the data evaluator's secret key sk. The data's security depends on the encryption algorithm used (e.g., Paillier). Therefore, we can assert that the ADV cannot access private data even if the communication flow is intercepted. Moreover, even in multi-data-provider schemes, the ADV can collude with some data providers, the ADV cannot reveal the private data of uncompromised data providers. Compromised data providers do not possess the data evaluator's secret key sk_{DE} using in common scheme nor different data providers' secret key sk_{DP} using in proxy encryption scheme, with which to decrypt other providers' messages.

We can treat the noise generated by the data provider, CSP or a noise server as the data provider's private data in transmission process. Thus, the ADV also cannot reveal private data during the noise transmission. Due to the noise addition method, even if the data provider, NSP or CSP discloses the part of the noise that it generates, the data evaluator cannot recover the original data because of the noise addition is performed by CSP and the order after adding will be permutated. It is reasonable to assume that (1) in common schemes, the CSP will not collude with the ADV. (2) in scheme with a NSP, the NSP and CSP will not both disclose their data at the same time and the NSP and CSP will not collude.

References

1. Barak, B., Chaudhuri, K., Dwork, C., Kale, S., Mcsherry, F., Talwar, K.: Privacy, accuracy, and consistency too: a holistic solution to contingency table release, pp. 273–282 (2007)

2. Benjamin, D., Atallah, M.J.: Private and cheating-free outsourcing of algebraic computations, pp. 240–245 (2008)
3. Blum, A., Ligett, K., Roth, A.: A learning theory approach to non-interactive database privacy, pp. 609–618 (2008)
4. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-48071-4_7
5. Chen, R., Mohammed, N., Fung, B.C.M., Desai, B.C., Xiong, L.: Publishing set-valued data via differential privacy. Proc. VLDB Endow. **4**, 1087–1098 (2011)
6. Chevallier-Mames, B., Coron, J.-S., McCullagh, N., Naccache, D., Scott, M.: Secure delegation of elliptic-curve pairing. In: Gollmann, D., Lanet, J.-L., Iguchi-Cartigny, J. (eds.) CARDIS 2010. LNCS, vol. 6035, pp. 24–35. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12510-2_3
7. Cormode, G., Procopiuc, C.M., Srivastava, D., Shen, E., Yu, T.: Differentially private spatial decompositions, pp. 20–31 (2012)
8. Dwork, C., Lei, J.: Differential privacy and robust statistics, pp. 371–380 (2009)
9. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_14
10. Dwork, C., Naor, M., Reingold, O., Rothblum, G.N., Vadhan, S.: On the complexity of differentially private data release: efficient algorithms and hardness results, pp. 381–390 (2009)
11. Dwork, C., Naor, M., Vadhan, S.: The privacy of the analyst and the power of the state, pp. 400–409 (2012)
12. Dwork, C., Rothblum, G.N., Vadhan, S.: Boosting and differential privacy, pp. 51–60 (2010)
13. Gupta, S.K., Rana, S., Venkatesh, S.: Differentially private multi-task learning. In: Chau, M., Wang, G.A., Chen, H. (eds.) PAISI 2016. LNCS, vol. 9650, pp. 101–113. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31863-9_8
14. Hardt, M., Rothblum, G.N., Servedio, R.A.: Private data release via learning thresholds, pp. 168–187 (2012)
15. Hay, M., Rastogi, V., Miklau, G., Suciu, D.: Boosting the accuracy of differentially private histograms through consistency. Proc. VLDB Endow. **3**, 1021–1032 (2010)
16. Huang, Z., Liu, S., Mao, X., Chen, K., Li, J.: Insight of the protection for data security under selective opening attacks. Inf. Sci. **412–413**, 223–241 (2017)
17. Jagadish, H.V., Koudas, N., Muthukrishnan, S., Poosala, V., Sevcik, K.C., Suel, T.: Optimal histograms with quality guarantees. Very Large Data Bases, 275–286 (2010)
18. Ji, Z., Jiang, X., Wang, S., Xiong, L., Ohnomachado, L.: Differentially private distributed logistic regression using private and public data. BMC Med. Genomics **7**(1), 1–10 (2014)
19. Kasiviswanathan, S.P., Lee, H.K., Nissim, K., Raskhodnikova, S., Smith, A.: What can we learn privately, pp. 531–540 (2008)
20. Kasiviswanathan, S.P., Rudelson, M., Smith, A., Ullman, J.: The price of privately releasing contingency tables and the spectra of random matrices with correlated rows, pp. 775–784 (2010)
21. Li, C., Hay, M., Rastogi, V., Miklau, G., Mcgregor, A.: Optimizing linear counting queries under differential privacy, pp. 123–134 (2010)
22. Li, P., Li, J., Huang, Z., Gao, C., Chen, W., Chen, K.: Privacy-preserving outsourced classification in cloud computing. Cluster Comput. 1–10 (2017)

23. Li, P., et al.: Multi-key privacy-preserving deep learning in cloud computing. Future Gener. Comput. Syst. **74**, 76–85 (2017)
24. Mcsherry, F.: Privacy integrated queries: an extensible platform for privacy-preserving data analysis. Commun. ACM **53**(9), 89–97 (2010)
25. Mcsherry, F., Talwar, K.: Mechanism design via differential privacy, pp. 94–103 (2007)
26. Mohammed, N., Chen, R., Fung, B.C.M., Yu, P.S.: Differentially private data release for data mining, pp. 493–501 (2011)
27. Pathak, M.A., Rane, S., Raj, B.: Multiparty differential privacy via aggregation of locally trained classifiers, pp. 1876–1884 (2010)
28. Qardaji, W., Yang, W., Li, N.: Differentially private grids for geospatial data, pp. 757–768 (2013)
29. Qardaji, W., Yang, W., Li, N.: Priview: practical differentially private release of marginal contingency tables, pp. 1435–1446 (2014)
30. Shokri, R., Shmatikov, V.: Privacy-preserving deep learning, pp. 1310–1321 (2015)
31. Su, S., Tang, P., Cheng, X., Chen, R., Wu, Z.: Differentially private multi-party high-dimensional data publishing, pp. 205–216 (2016)
32. Vaidya, J., Clifton, C.: Privacy preserving association rule mining in vertically partitioned data, pp. 639–644 (2002)
33. Vaidya, J., Clifton, C.: Privacy-preserving k-means clustering over vertically partitioned data, pp. 206–215 (2003)
34. Wang, B., Li, M., Chow, S.S.M., Li, H.: A tale of two clouds: computing on data encrypted under multiple keys. In: Communications and Network Security, pp. 337–345 (2014)
35. Xiao, X., Wang, G., Gehrke, J.: Differential privacy via wavelet transforms. IEEE Trans. Knowl. Data Eng. **23**(8), 1200–1214 (2011)
36. Xiao, Y., Xiong, L., Yuan, C.: Differentially private data release through multidimensional partitioning. In: Jonker, W., Petković, M. (eds.) SDM 2010. LNCS, vol. 6358, pp. 150–168. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15546-8_11
37. Xu, J., Zhang, Z., Xiao, X., Yang, Y., Yu, G., Winslett, M.: Differentially private histogram publication. Very Large Data Bases **22**(6), 797–822 (2013)
38. Yuan, G., Zhang, Z., Winslett, M., Xiao, X., Yang, Y., Hao, Z.: Low-rank mechanism: optimizing batch queries under differential privacy. Proc. VLDB Endow. **5**(11), 1352–1363 (2012)