Secure Distributed Framework for Achieving ϵ-Differential Privacy

Dima Alhadidi, Noman Mohammed, Benjamin C.M. Fung,
and Mourad Debbabi

Concordia Institute for Information Systems Engineering,
Concordia University, Montreal, Quebec, Canada
{dm_alhad,no_moham,fung,debbabi}@encs.concordia.ca

Abstract. Privacy-preserving data publishing addresses the problem of disclosing sensitive data when mining for useful information. Among the existing privacy models, *ϵ-differential privacy* provides one of the strongest privacy guarantees. In this paper, we address the problem of private data publishing where data is horizontally divided among two parties over the same set of attributes. In particular, we present the first generalization-based algorithm for differentially private data release for horizontally-partitioned data between two parties in the semi-honest adversary model. The generalization algorithm correctly releases differentially-private data and protects the privacy of each party according to the definition of secure multi-party computation. To achieve this, we first present a two-party protocol for the exponential mechanism. This protocol can be used as a subprotocol by any other algorithm that requires exponential mechanism in a distributed setting. Experimental results on real-life data suggest that the proposed algorithm can effectively preserve information for a data mining task.

1 Introduction

Data can be horizontally-partitioned among different parties over the same set of attributes.These distributed data can be integrated for making better decisions and providing high-quality services. However, data integration should be conducted in a way that no more information than necessary should be revealed between the participating entities. At the same time, new knowledge that results from the integration process should not be misused by adversaries to reveal sensitive information that has not been available before the data integration. In this paper, we propose an algorithm to securely integrate sensitive data , which is horizontally divided among two parties over the same set of attributes, whereby the integrated data still retains the essential information for supporting data mining tasks. The following scenario further motivates the problem.

Consider a blood bank collects and examines the blood provided from donors and then distributes the blood to different hospitals. Periodically, hospitals are required to submit the blood transfusion information, together with the patient surgery data, to the blood bank for classification analysis [1]. Due to privacy concerns and privacy regulations, hospitals cannot provide any information about

S. Fischer-Hübner and M. Wright (Eds.): PETS 2012, LNCS 7384, pp. 120–139, 2012.

Table 1. Data Set D_1

ID	Class	Job	Sex	Age	Surgery
1	N	Janitor	M	34	Transgender
2	Y	Lawyer	F	58	Plastic
3	Y	Mover	M	58	Urology
4	N	Lawyer	M	24	Vascular
5	Y	Mover	M	34	Transgender
6	Y	Janitor	M	44	Plastic
7	Y	Doctor	F	44	Vascular

Table 2. Data Set D_2

ID	Class	Job	Sex	Age	Surgery
8	N	Doctor	M	58	Plastic
9	Y	Doctor	M	24	Urology
10	Y	Janitor	F	63	Vascular
11	Y	Mover	F	63	Plastic

individual medical records to the blood bank. Accordingly, there is a desideratum for an approach that allows anonymizing horizontally-partitioned data from different providers for data release. The resulted anonymizing data should not contain individually identifiable information and at the same time the data providers should not reveal their private data or the ownership of the data to each other.

Example 1. Suppose the first hospital P_1 and the second hospital P_2 own the data sets D_1 and D_2 as shown in Table 1 and Table 2, respectively. Each hospital has records for different individuals. The attribute *Class* contains the label Y or N, representing whether or not the patient has received blood transfusion. Both parties want to integrate their data and use the integrated data to build a classifier on the *Class* attribute. After the integration, the sensitive data of the patient #5 can be uniquely identified since he is the only 34-year mover in the data set. Moreover, we can infer that a 34-year male has performed a transgender surgery since both patients in the integrated data set has performed it.

In this context, Jurczyk and Xiong [2] have proposed an algorithm to securely integrate horizontally-partitioned data from multiple data owners. Mohammed *et al.* [1] have proposed a distributed algorithm to integrate horizontally-partitioned high-dimensional health care data. Their methods [1,2] adopt k-anonymity [3,4] or its extensions [5,6] as the underlying privacy principle. Recently, Wong *et al.* [7] and Zhang *et al.* [8] have shown that algorithms, which satisfy k-anonymity [3,4] or its extensions [5,6], are vulnerable to minimality attack and do not provide the claimed privacy guarantee. Although several fixes against minimality attack have been proposed [9], new attacks such as composition attack [10] and deFinetti attack [11] have emerged against algorithms that adopt k-anonymity or its extensions.

In this respect, differential privacy [12], which is a recently proposed privacy model, provides provable privacy guarantee and it is, by definition, immune against all these attacks. A differentially-private mechanism ensures that the probability of any output (released data) is equally likely from all nearly identical input data sets and thus guarantees that all outputs are insensitive to any individual's data. In other words, an individual's privacy is not at risk because of the participation in the data set. In this paper, we present the first generalization-based algorithm for differentially-private data release for horizontally-partitioned data between two parties in the semi-honest adversary model. We take the single-party algorithm

Table 3. Related Work - Summary

Algorithms	**Data Owner**			**Privacy Model**	
	Single	**Multi**		**Differential Privacy**	**Partition-based Privacy**
		Horizontally	**Vertically**		
LeFevre *et al.* [15], Fung *et al.* [16], etc	✓				✓
Xiao *et al.* [17], Mohammed *et al.* [13], etc.	✓			✓	
Jurczyk and Xiong [2], Mohammed *et al.* [1]		✓			✓
Jiang and Clifton [18], Mohammed *et al.* [19]			✓		✓
Our proposal		✓		✓	

for differential privacy that has been recently proposed by Mohammed *et al.* [13] as a basis and extend it to the two-party setting. The main contribution of our paper can be summarized as follows:

- We present a two-party protocol for the exponential mechanism for horizontally-partitioned data. We use this protocol as a subprotocol of our main algorithm.
- We present the first non-interactive two-party data publishing algorithm for horizontally-partitioned data which achieves differential privacy and satisfies the security definition of secure multiparty computation (SMC). In a non-interactive framework, a database owner first anonymizes the raw data and then releases the anonymized version for data analysis. This approach is also known as privacy-preserving data publishing (PPDP) [14].
- We experimentally show that the proposed algorithm can preserve information for classification analysis..

The rest of the paper is organized as follows. Section 2 presents related work. Section 3 overviews privacy and security models adopted in this paper.The two-party data publishing algorithm for horizontally-partitioned data is presented in Section 4. In Section 5, we describe the two-party protocol for the exponential mechanism. We discuss in Section 6 the correctness, the security and the efficiency of the two-party data publishing algorithm. Section 7 presents the experimental results, and estimates the computation and communication cost of the algorithm for a real data set. Section 8 answers some frequently raised questions. Finally, concluding remarks as well as a discussion of future work are presented in Section 9.

2 Related Work

The primary goal of our study in this paper is to share data. In contrast, privacy preserving distributed data mining (PPDDM) [20] allows sharing of the computed result (e.g., a classifier), but completely prohibits sharing data. In PPDDM, multiple data owners want to compute a function based on their inputs without sharing their data with others. This function can be as simple as a count query or as complex as a data mining task such as classification, clustering, etc. However, compared to data mining result sharing, data sharing gives greater flexibility because recipients can perform their required analysis and data exploration, and apply different modeling methods and parameters.

Our approach allows anonymizing data from different sources for data release without exposing the sensitive information. Jiang and Clifton [18] have proposed Distributed k-Anonymity (DkA) framework to securely integrate two data tables while satisfying k-anonymity requirement. Mohammed *et al.* [19] have proposed an efficient anonymization algorithm to integrate data from multiple data owners. Unlike the distributed anonymization problem for horizontally-partitioned data studied in this paper, these methods [18,19] propose algorithms for vertically-partitioned data. Jurczyk and Xiong [2] have proposed an algorithm to securely integrate horizontally-partitioned data from multiple data owners. Mohammed *et al.* [1] have proposed a distributed algorithm to integrate horizontally-partitioned high-dimensional health care data. To the best of our knowledge, these are the only two methods [1,2] that generate an anonymous table for horizontally-partitioned data. However, both the methods adopt k-anonymity [3,4] or its extensions [5,6] as the underlying privacy principle; therefore, are vulnerable to the recently discovered privacy attacks [7,10,11].

Differential privacy [12] has received considerable attention recently as a substitute for partition-based privacy models for PPDP . However, most of the research on differential privacy so far concentrates on the interactive [12,21] and non-interactive [13,17] setting for the single-party scenario. Therefore, these techniques do not address the problem of privacy-preserving data sharing for classification analysis; the primary theme of this paper. Finally, Dwork *et al.* [22] have proposed a distributed interactive protocol for computing a function while guaranteeing differential privacy. Given a function, each party first computes the function on its own data and then perturbs the result appropriately such that the summation of all the perturbed results from all the parties generates a differentially private output. As mentioned already, interactive approach does not allow data sharing and therefore does not address the problem studied in this paper.

3 Background

In this section, we first present an overview of differential privacy. Then, we briefly discuss the security definition in the semi-honest adversary model. Additionally, we overview the required cryptographic primitives for the proposed algorithm.

3.1 Privacy Model

Differential privacy is a recent privacy definition that provides a strong privacy guarantee. It guarantees that an adversary learns nothing more about an individual, regardless of whether her record is present or absent in the data.

Definition 1. (ϵ-Differential Privacy) *[12] A randomized algorithm Ag is differentially private if for all data sets D and D' where their symmetric difference contains at most one record (i.e., $|D \triangle D'| \leq 1$), and for all possible anonymized data sets $\hat{D}$,*

$$Pr[Ag(D) = \hat{D}] \leq e^{\epsilon} \times Pr[Ag(D') = \hat{D}], \tag{1}$$

where the probabilities are over the randomness of the Ag.

A standard mechanism to achieve differential privacy is to add random noise to the true output of a function. The noise is calibrated according to the *sensitivity* of the function. The sensitivity of a function is the maximum difference of its outputs from two data sets that differ only in one record.

Definition 2. (Sensitivity) *[12] For any function $f : D \to \mathbb{R}^d$, the sensitivity of f is*

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

for all D, D' differing in at most one record.

For example, let f be the count function. The Δf is 1 because $f(D)$ can differ at most by 1 due to the addition or to the removal of a single record.

Dwork *et al.* [12] have proposed the Laplace mechanism. The mechanism takes a data set D, a function f, and the parameter λ that determines the magnitude of noise as inputs. It first computes the true output $f(D)$, and then perturbs the output by adding noise. The noise is generated according to a Laplace distribution with probability density function $\Pr(x|\lambda) = \frac{1}{2\lambda}\texttt{exp}(-|x|/\lambda)$; its variance is $2\lambda^2$ and its mean is 0. Laplace mechanism guarantees that perturbed output $f(\hat{D}) = f(D) + \texttt{Lap}(\Delta f/\epsilon)$ satisfies ϵ-differential privacy, where $\texttt{Lap}(\Delta f/\epsilon)$ is a random variable sampled from the Laplace distribution.

McSherry and Talwar [23] have proposed the exponential mechanism to achieve differential privacy whenever it makes no sense to add noise to outputs. The exponential mechanism can choose an output $t \in \mathcal{T}$ that is close to the optimum with respect to a utility function while preserving differential privacy. It takes as inputs a data set D, an output range $\mathcal{T}$, a privacy parameter ϵ, and a utility function $u : (D \times \mathcal{T}) \to \mathbb{R}$ that assigns a real valued score to every output $t \in \mathcal{T}$, where a higher score means better utility. The mechanism induces a probability distribution over the range $\mathcal{T}$ and then samples an output t. Let $\Delta u = \max_{\forall t,D,D'} |u(D,t) - u(D',t)|$ be the sensitivity of the utility function. The probability associated with each output is proportional to $\texttt{exp}(\frac{\epsilon u(D,t)}{2\Delta u})$; that is, the output with a higher score is exponentially more likely to be chosen.

3.2 Security Model

In this subsection, we briefly present the security definition in the semi-honest adversary model. Moreover, we overview the required cryptographic primitives.

Secure Multiparty Computation. In the semi-honest model, adversaries follow the protocol but may try to deduce additional information from the received messages. A protocol is private in a semi-honest environment if the view of each party during the execution of the protocol can be effectively simulated by a probabilistic polynomial-time algorithm knowing only the input and the output of that party [24]. Many of the protocols, as it is the case with the proposed algorithm in this paper, involve the composition of privacy-preserving subprotocols in which all intermediate outputs from one subprotocol are inputs to the next subprotocol. These intermediate outputs are either simulated given the final output and the local input for each party or computed as random shares. Using the composition theorem [24], it can be shown that if each subprotocol is privacy-preserving, then the resulting composition is also privacy-preserving.

Cryptographic Primitives. The required cryptographic primitives utilized in this paper are:

- Yao's Protocol [26]. It is a constant-round protocol for secure computation of any probabilistic polynomial-time function in the semi-honest model. More specifically, assume that we have two parties P_1 and P_2 with their inputs x and y, respectively. Both parties want to compute the value of the function $f(x, y)$. Then, P_1 needs to send P_2 an encrypted circuit computing $f(x, .)$. The received circuit is encrypted and accordingly P_2 learns nothing from this step. Afterwards, P_2 computes the output $f(x, y)$ by decrypting the circuit. This can be achieved by having P_2 obtaining a series of keys corresponding to its input y from P_1 such that the function $f(x, y)$ can be computed given these keys and the encrypted circuit. However, P_2 must obtain these keys from P_1 without revealing any information about y. This is done by using oblivious transfer protocol [24].
- Random Value Protocol (RVP) [27]. It describes how two parties can share a value $R \in \mathbb{Z}_Q$ where R has been chosen uniformly at random and $Q \in \mathbb{Z}_N$ is not known by either party, but is shared between them. More specifically, P_1 has $R_1 \in \mathbb{Z}_N$ and P_2 has $R_2 \in \mathbb{Z}_N$ such that $R = R_1 + R_2 \ mod \ N \in [0, Q-1]$ where N is the public key for the an additive homomorphic scheme.
- Oblivious Polynomial Evaluation (OPE) [28]. It is a protocol involving two parties, a sender whose input is a polynomial P, and a receiver whose input is a value α. At the end of the protocol, the receiver learns $P(\alpha)$ and the sender learns nothing.

4 Two-Party Differentially Private Data Release

In this section, we present our two-party algorithm for differentially-private data release for horizontally-partitioned data. To facilitate understanding the algorithm, we first present the notation that is used along this paper.

4.1 Notation and Preliminaries

Suppose two parties P_1 and P_2 own data table D_1 and D_2, respectively. Both the parties want to release an integrated anonymous data table $\hat{D}(A_1^{pr}, \ldots, A_d^{pr}, A^{cls})$ to the public for classification analysis. The attributes in D_1 and D_2 are classified into three categories: (1) An explicit identifier attribute A^i that explicitly identifies an individual, such as *SSN* and *Name*. These attributes are removed before releasing the data. (2) A class attribute A^{cls} that contains the class value, and the goal of the data miner is to build a classifier to accurately predict the value of this attribute. (3) A set of predictor attributes

Algorithm 1. Two-Party Algorithm

Input: Raw data set D_1, privacy budget ϵ, and number of specializations h
Output: Anonymized data set $\hat{D}$

```
Initialize $D_g$ with one record containing top most values;
Initialize $Cut_i$ to include the topmost value;
$\epsilon' \leftarrow \frac{\epsilon}{2(|A_n^{pr}|+2h)}$;
for $l = 1$ to $h$ do
    Determine winner candidate $w$ by DEM($D_1$,$D_2$,$\cup Cut_i$,$\epsilon'$);
    Specialize $w$ on $D_g$;
    Replace $w$ with $child(w)$ in $\cup Cut_i$;
end for
for each leaf node of $D_g$ do
    Compute the share $C_1$ of the true count $C$;
    Compute $X_1 = C_1 + \mathtt{Lap}(2/\epsilon)$;
    Exchange $X_1$ with $P_2$ to compute $(C + 2 \times \mathtt{Lap}(2/\epsilon))$;
end for
return Each leaf node with count $(C + 2 \times \mathtt{Lap}(2/\epsilon))$
```

$\mathcal{A}^{pr} = \{A_1^{pr}, \ldots, A_d^{pr}\}$, whose values are used to predict the class attribute. Given a table D_1 owned by P_1, a table D_2 owned by P_2 and a privacy parameter ϵ, our objective is to generate an integrated anonymized data table $\hat{D}$ such that (1) $\hat{D}$ satisfies ϵ-differential privacy and (2) the algorithm to generate $\hat{D}$ satisfies the security definition of the semi-honest adversary model.

We require the class attribute to be categorical. However, the values of the predictor attribute can be either numerical v_n or categorical v_c. Further, we require that for each predictor attribute A^{pr}, which is either numerical or categorial, a taxonomy tree is provided. We assume that there is no trusted third party who computes the output table $\hat{D}$ and the parties are semi-honest. Moreover, we assume that the two data sets include disjoint tuples and are defined on exactly the same schema.

4.2 Anonymization Algorithm

In this section, we present our distributed differentially-private anonymization algorithm based on generalization for two parties as shown in Algorithm 1. Algorithm 1 is executed by the party P_1 (same for the party P_2). The algorithm first generalizes the raw data and then adds noise to achieve ϵ-differential privacy.

Generalizing the Raw Data. The general idea is to anonymize the raw data by a sequence of specializations starting from the topmost general state. A specialization, written $v \rightarrow child(v)$ replaces the parent value v with its set of child values $child(v)$. The specialization process can be viewed as pushing the "cut" of each taxonomy tree downwards. A *cut* of the taxonomy tree for an attribute A_i^{pr}, denoted by Cut_i, contains exactly one value on each root-to-leaf path. Each party keeps a copy of the current $\cup Cut_i$ and a generalized table D_g, in addition to the private table D_1 or D_2. Initially, all values in $\mathcal{A}^{pr}$ are generalized to the

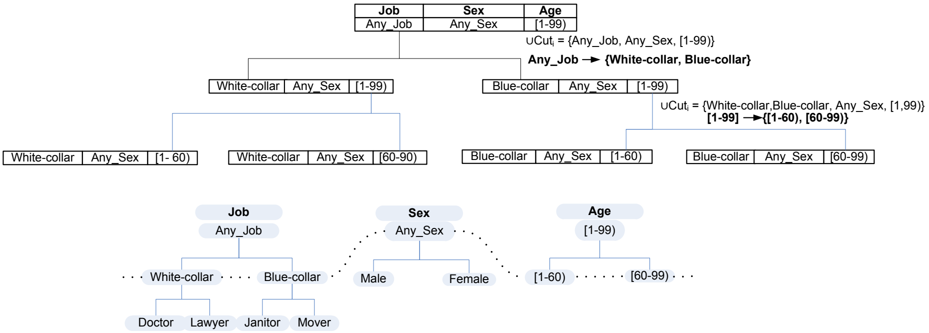

Fig. 1. Generalized Data Table (D_g)

topmost value in their taxonomy trees, and Cut_i contains the topmost value for each attribute A_i^{pr}. At each iteration, Algorithm 1 uses the distributed exponential mechanism to select a candidate for specialization (Line 5) depending on its score. This can be achieved by calling Algorithm 2 detailed in Section 5. Once a candidate is determined, both the parties specialize the winner candidate w on D_g (Line 6) by splitting their records into child partitions according to the provided taxonomy trees. Then, the parties update their local copy of $\cup Cut_i$ (Line 7). This process is repeated according to the number of specializations h.

Example 2. Consider Table 1 and Table 2 and the taxonomy trees presented at the bottom of Fig. 1. We do not show the class and the surgery attributes in Fig. 1 due to space limitation. Initially, D_g contains one root node representing all the records that are generalized to $\langle$*Any_Job*, *Any_Sex*, *[1-99)*$\rangle$. $\cup Cut_i$ is represented as $\{$*Any_Job, Any_Sex, [1-99)*$\}$ and includes the initial candidates. To find the winner candidate, both parties run DEM. Suppose that w is *Any_Job* $\rightarrow$ $\{$*White-collar*, *Blue-collar*$\}$. Both parties create two child nodes under the root node as shown in Fig. 1 and updates $\cup Cut_i$ to $\{$*White-collar*, *Blue-collar*, *Any_Sex*, *[1-99)*$\}$. Suppose that the next winning candidate is *[1-99)* $\rightarrow$ $\{$*[1-60), [60-99)*$\}$. Similarly, the two parties create further specialized partitions resulting the generalized table in Fig. 1.

Adding Noisy Count. Each party computes the number of its records under each leaf node (Line 10). To have an exchange between the parties that is differentially-private, each party adds a Laplace noise to its count (Line 11) and sends the result to the other party (Line 12). The protocol ends up with two Laplace noises added to the count of each leaf (Line 14).

5 Two-Party Protocol for Exponential Mechanism

Exponential mechanism chooses a candidate that is close to optimum with respect to a utility function while preserving differential privacy. In the

Table 4. `MAX` score calculation for the candidate *Any_Job*

Max	Class		Job	Data Set
	Y	N		
5	3	1	Blue-collar	D_1
	2	1	White-collar	
3	2	0	Blue-collar	D_2
	1	1	White-collar	
8	5	1	Blue-collar	Integrated D_1 and D_2
	3	2	White-collar	

distributed setting, the same candidates are owned by two parties while records are horizontally-partitioned among them. Consequently, we need a private mechanism to compute the same output while ensuring that no extra information is leaked to any party. In this section, we present a two-party protocol for exponential mechanism in a distributed setting. We adopt the max utility function to compute the scores. For this reason, we illustrate first how this function is computed. Other utility functions can be adopted as discussed in Section 8.

5.1 Max Utility Function

To compute the score of each candidate, we adopt the max utility function [13].

$$\mathtt{Max}(D, v) = \sum_{a \in child(v)} \max_{c}(|T_D(a, c)|) \tag{3}$$

where the notation T denotes the set of transactions (records) and $|T_D(a, c)|$ denotes the number of records in D having the generalized value a and the class value c. Thus, $\mathtt{Max}(D, v)$ is the summation of the highest class frequencies over all child values. The sensitivity Δu of the `Max` function is 1 because the value $\mathtt{Max}(D, v)$ can vary at most by 1 due to a record change. The following example clarifies how to evaluate the max utility function.

Example 3. The maximum utility function of the candidate *Any_Job* of Table 1 is 5. Table 4 demonstrates how the value 5 is computed. For each possible child value of the candidate *Any_Job*, we compute the number of records having the class value Y and the class value N. Afterwards, we pick the maximum class frequency for each child value and sum them. In the same vein, the maximum utility function of the the candidate *Any_Job* of Table 2 is 3. If we integrate the two tables, the maximum utility function of the candidate *Any_Job* is 8. Note that, the maximum utility function of an integrated table is not the sum of the values of maximum utility function of each source data set.

5.2 Distributed Exponential Mechanism

The *distributed exponential mechanism (DEM)* presented in Algorithm 2 takes the followings as inputs:(1) Two raw data sets D_1 and D_2 owned by P_1 and P_2, respectively, (2) set of candidates $\{v_1, \ldots, v_k\}$, and (3) privacy budget ϵ. The

Algorithm 2. Distributed Exponential Mechanism (DEM)

Input: Raw data set D_1 owned by P_1, raw data set D_2 owned by P_2, a set of candidates $\{v_1, \ldots, v_k\}$ and privacy budget ϵ

Output: Winner w

```
 1: for each candidate $v_x$ where $x = 1$ to $k$ do
 2:   for (each possible value of $a_j$ of $v_x$ where $j = 1$ to $m$) do
 3:     for (each class value $c_i$ where $i = 1$ to $l$) do
 4:       $P_1$ computes $|T_{D_1}(a_j, c_i)|$;
 5:       $P_1$ computes $|T_{D_2}(a_j, c_i)|$;
 6:     end for
 7:   end for
 8:   $P_2$ generates a random share $\alpha_2$;
 9:   $(P_1 \leftarrow \alpha_1, P_2 \leftarrow \perp) \leftarrow \texttt{MAX}(|T_{D_1}(a_j, c_i)|_{i=1\ to\ l, j=1\ to\ m}, |T_{D_2}(a_j, c_i)|_{i=1\ to\ l, j=1\ to\ m}, \alpha_2)$;
10:   $P_1$ chooses a random share $\beta_x$ and defines the following polynomial $Q(z) = lcm(2!, \ldots, w!)\ .10^{sw}.\ \sum_{i=0}^{w} \frac{((\frac{\epsilon}{2\Delta \mathrm{u}})_s.10^s.(\alpha_1+z))^i}{10^{s(i-1)}.i!} - \beta_x$;
11:   $P_1$ and $P_2$ execute a private polynomial with $P_1$ inputting $Q(.)$ and $P_2$ inputting $\alpha_2$, in which $P_2$ obtains $\beta'_x = Q(\alpha_2)$.
12: end for
13: $(P_1 \leftarrow \gamma_1, P_2 \leftarrow \perp) \leftarrow \texttt{SUM}(\beta_{x,x=1\ to\ k}, \beta'_{x,x=1\ to\ k}, \gamma_2)$;
14: $P_1$ and $P_2$ execute RVP to compute random shares $R_1$ and $R_2$, where $(R_1 + R_2) \in \mathbb{Z}_{(\gamma_1+\gamma_2)}$;
15: $P_1$ and $P_2$ evaluates $x \leftarrow \texttt{COMPARISON}(R_1, R_2, \beta_{x,x=1\ to\ k}, \beta'_{x,x=1\ to\ k})$;
16: return $v_x$;
```

protocol outputs a winner candidate depending on its score using the exponential mechanism. The scores of the candidates can be calculated using different utility functions [13]. In this paper, we adopt the max utility function described previously to calculate the scores. Given the scores of all the candidates, exponential mechanism selects the candidate v having score u with the following probability where Δu is the sensitivity of the chosen utility function.

$$\frac{\texttt{exp}(\frac{\epsilon u}{2\Delta \mathrm{u}})}{\sum_{n=1}^{k} \texttt{exp}(\frac{\epsilon u_n}{2\Delta \mathrm{u}})} \tag{4}$$

Next, we detail the steps of the distributed exponential mechanism (DEM).

Computing Max Utility Function. To compute the max utility function for each candidate v_x, the parties P_1 and P_2 compute $|T_{D_1}(a_j, c_i)|$ and $|T_{D_1}(a_j, c_i)|$, respectively for every possible value a_j of v_x and for every possible value c_i of the class attribute (Lines 2 to 7). After that, the two parties engage in a secure circuit evaluation process using Yao's Protocol (Line 9). The values $|T_{D_1}(a_j, c_i)|_{i=1\ to\ l, j=1\ to\ m}$, $|T_{D_2}(a_j, c_i)|_{i=1\ to\ l, j=1\ to\ m}$ and α_2 are passed to the `MAX` circuit where α_2 is randomly generated by P_2. For each child value a_j of the candidate v_x, the circuit `MAX`, as shown in Algorithm 3, adds the corresponding values $|T_{D_1}(a_j, c_i)|$ and $|T_{D_2}(a_j, c_i)|$ for every possible value c_i of the

Algorithm 3. `MAX` Circuit

Input: $|T_{D_1}(a_j, c_i)|_{i=1\ to\ l, j=1\ to\ m}$, $|T_{D_2}(a_j, c_i)|_{i=1\ to\ l, j=1\ to\ m}$ and α_2
Output: α_1 to P_1,$\perp$ to P_2

```
$sum = 0$;
for $j = 1$ to $m$ do
    $max = 0$;
    for $i = 1$ to $l$ do
        $ss$= $|T_{D_1}(a_j, c_i)| + |T_{D_2}(a_j, c_i)|$;
        if $(ss > max)$ then
            $max = ss$;
        end if
    end for
    $sum = sum + max$;
end for
$\alpha_1 = sum$ - $\alpha_2$;
return $\alpha_1$,$\perp$;
```

Algorithm 4. `COMPARISON` Circuit

Input: Random shares R_1 and R_2, $\beta_{x,x=1\ to\ k}$, and $\beta'_{x,x=1\ to\ k}$
Output: Index x to P_1 and P_2

```
$L = 0$;
$R = R_1 + R_2$;
for $x = 1$ to $k$ do
    $\beta = \beta_x + \beta'_x$;
    $L = L + \beta$;
    if $(R \leq L)$ then
        return $x$;
    end if
end for
```

class attribute. It then computes the maximum value of the results. After that, the maximum values associated with each child value a_j should be summed to get the max utility function for the candidate v_x. To produce random shares of the max utility function, the circuit subtracts α_2, which is randomly generated by P_2, from the resulted score and outputs the result α_1 to P_1.

Computing Equation 4. The exponential function, $exp(x)$ can be defined using the following Taylor series:

$$1 + \frac{x}{1} + \frac{x^2}{2!} + \cdots + \frac{x^i}{i!} + \dots \tag{5}$$

To evaluate the nominator of Equation 4 for each v_x, we need to evaluate the expression $\texttt{exp}(\frac{\epsilon u}{2\Delta \mathrm{u}})$ which is equal to $\texttt{exp}(\frac{\epsilon(\alpha_1+\alpha_2)}{2\Delta \mathrm{u}})$. Given the aforementioned

Taylor series:

$$\texttt{exp}(\frac{\epsilon(\alpha_1 + \alpha_2)}{2\Delta\mathtt{u}}) = \sum_{i=0}^{w} \frac{(\frac{\epsilon(\alpha_1+\alpha_2)}{2\Delta\mathtt{u}})^i}{i!} \tag{6}$$

Hence, the next step involves computing shares of the Taylor series approximation. In fact, it computes shares of:

$$lcm(2!, \ldots, w!)\ .10^{s(w+1)}.\ \sum_{i=0}^{w} \frac{((\frac{\epsilon}{2\Delta\mathtt{u}})_s.(\alpha_1 + \alpha_2))^i}{i!}$$

where:

- $lcm(2!, \ldots, w!)$ is the lowest common multiple of {2!,. . . ,w!} and we multiply by it to ensure that there are no fractions.
- $(\frac{\epsilon}{2\Delta\mathtt{u}})_s$ refers to approximating the value of $\frac{\epsilon}{2\Delta\mathtt{u}}$ up to a predetermined number s after the decimal point. For example, if we assume $s = 4$ and $\epsilon = ln2$ then $(\frac{ln2}{2\times1})_4 = (0.3465)$. Note that, this approximation does not effect privacy guarantee since we are using less privacy budget. Also, the impact on the utility is insignificant. In Section 7, we experimentally show the accuracy for different privacy budgets.
- $10^{sw}.10^s$ is multiplied by the series to ensure that we end up with an integer result such that:

$$\begin{aligned}
&lcm(2!, \ldots, w!)\ .10^{sw}.10^s.\ \sum_{i=0}^{w} \frac{((\frac{\epsilon}{2\Delta\mathtt{u}})_s.(\alpha_1 + \alpha_2))^i}{i!} \\
&= lcm(2!, \ldots, w!)\ .10^{sw}.\ \sum_{i=0}^{w} 10^s.\frac{((\frac{\epsilon}{2\Delta\mathtt{u}})_s.(\alpha_1 + \alpha_2))^i}{i!} \\
&= lcm(2!, \ldots, w!)\ .10^{sw}.\ \sum_{i=0}^{w} \frac{((\frac{\epsilon}{2\Delta\mathtt{u}})_s.10^s.(\alpha_1 + \alpha_2))^i}{10^{s(i-1)}.i!}
\end{aligned}$$

Since s and w are known to both parties, the additional multiplicative factor $lcm(2!, \ldots, w!).10^{sw}.10^s$ is public and can be removed at the end (if desired). This equation is accurate up to an approximation error which depends on the value of w. Therefore, scaling is needed and consequently the accuracy of the exponential mechanism could be affected. However, if the scaling factor is very large, the total cost in terms of bits will increase. We experimentally measure the impact of scaling in Section 7 and show that the scaling has very little impact for the max utility function. The parties should agree on the number of the considered digits s after the decimal point. The higher accuracy (in terms of the number of the considered digits after the decimal point) we demand, the higher cost we pay (in terms of bits). That is for s decimal points, we need $log_2 10^s$ extra bits. These extra bits result additional computation and communication cost. More details are provided in Section 7. Note that restricting the values of

$\mathtt{exp}(\frac{\epsilon u}{2\Delta \mathrm{u}})$ to a finite range is completely natural as calculations performed on computers are handled in this manner due to memory constraints.

To evaluate the nominator of Equation 4 for each v_x in Algorithm 2, P_1 chooses a random share β_x and defines the following polynomial where s is a constant number (Line 10):

$$Q(z) = lcm(2!, \ldots, w!)\ .10^{sw}.\ \sum_{i=0}^{w} \frac{((\frac{\epsilon}{2\Delta \mathrm{u}})_s.10^s.(\alpha_1 + z))^i}{10^{s(i-1)}.i!} - \beta_x$$

Afterwards, P_1 and P_2 execute a private polynomial with P_1 inputting $Q(.)$ and P_2 inputting α_2, in which P_2 obtains $\beta'_x = Q(\alpha_2)$ (Line 11). To evaluate the denominator of Equation 4, the two parties execute the circuit `SUM` which takes as input the random shares β_x and β'_x for each candidate v_x and a random number γ_2 generated by P_2 (Line 13). The circuit computes the total sum of the results that come out because of adding the random shares β_x and β'_x for each candidate v_x. It then subtracts γ_2, which is randomly generated by P_2, from the value of the total sum and outputs the share γ_1 to P_1.

Once we compute the denominator and numerator of Equation 4, we can implement the exponential mechanism by first partitioning the interval [0,1] into segments according to the corresponding probability mass of each candidate. Next, we sample a random number uniformly in the range [0,1] and the partition in which the random number falls determines the winner candidate. However, this method involves computing a secure division (Equation 4). Unfortunately, we are not aware of any secure division scheme that fits our scenario where the nominator value is less than the denominator value. Alternatively, we solve this problem without a secure division protocol. We first partition the interval $[0, \sum_{x=1}^{k} \mathtt{exp}(\frac{\epsilon u_x}{2\Delta \mathrm{u}})]$ into k segments where $\sum_{x=1}^{k} \mathtt{exp}(\frac{\epsilon u_x}{2\Delta \mathrm{u}}) \approx \gamma_1 + \gamma_2$ and each segment corresponds to a candidate v_x has a subinterval of length equal to $\beta_x + \beta'_x$. We then sample a random number uniformly in the range $[0, \gamma_1 + \gamma_2]$ and the segment in which the random number falls determines the winner candidate.

Picking a Random Number. The parties P_1 and P_2 need to pick a random number uniformly in the range $[0, \gamma_1 + \gamma_2]$, where $\gamma_1 + \gamma_2 \approx \sum_{x=1}^{k} \mathtt{exp}(\frac{\epsilon u_x}{2\Delta \mathrm{u}})$. This can be achieved by using the Random Value Protocol (RVP) [27] (Line 14). RVP takes γ_1 and γ_2 from the parties as input and outputs the random value shares R_1 and R_2 to the respective parties, where $R = R_1 + R_2$.

Example 4. Suppose the values of the expression $\mathtt{exp}(\frac{\epsilon u}{2\Delta \mathrm{u}})$ is approximated to 60, 150 and 90 for three candidates as shares. Both parties then pick a random number in the range $[0, 300]$ using the RVP where $300 = 50 + 150 + 90$.

Picking a Winner. The two parties engage again in a simple secure circuit evaluation process using Yao's Protocol [26] (Line 15). The circuit `COMPARISON` compares their random number R with the sum L. The winner v_x is the first candidate such that $R \leq L$ where $L = \sum_{r=1}^{x} (\beta_x + \beta'_x)$.

6 Analysis

We discuss in this section the correctness, security and efficiency of Algorithm 1.

Proposition 1. (Correctness) *Assuming both parties are semi-honest, Algorithm 1 releases ϵ-differentially private data when data records are divided horizontally among two parties over the same set of attributes.*

Proof. Algorithm 1 performs the same function as the single-party algorithm DiffGen [13] but in a distributed setting. DiffGen is ϵ-differentially private. Therefore, we prove the correctness of Algorithm 1 by just proving the steps that are different from DiffGen:

- Candidate selection. Algorithm 1 selects a candidate for specialization (Line 5) using Algorithm 2. Algorithm 2 selects a candidate v_w with probability $\propto \texttt{exp}(\frac{\epsilon u_w}{2\Delta \text{u}})$. The two parties compute cooperatively $\texttt{exp}(\frac{\epsilon u}{2\Delta \text{u}})$ for the candidates. Then the parties build an interval in the range $[0,\sum_{x=1}^{k} \texttt{exp}(\frac{\epsilon u_x}{2\Delta \text{u}})]$ and partition it among the candidates where each subinterval has a length equal to $\texttt{exp}(\frac{\epsilon u}{2\Delta \text{u}})$. Since, the random value lies uniformly between $[0,\sum_{x=1}^{k} \texttt{exp}(\frac{\epsilon u_x}{2\Delta \text{u}})]$ and a candidate is chosen according to this value, the probability of choosing any candidate is $\frac{\texttt{exp}(\frac{\epsilon u}{2\Delta \text{u}})}{\sum_{x=1}^{k} \texttt{exp}(\frac{\epsilon u_x}{2\Delta \text{u}})}$. Therefore, Algorithm 2 correctly implements exponential mechanism.
- Updating the tree D_g and $\cup Cut_i$. Each party has its own copy of D_g and $\cup Cut_i$. Each party updates these items exactly like DiffGen (Lines 6-7).
- Computing the noisy count. Algorithm 1 also outputs the noisy count of each leaf node (Line 14), where the noise is equal to $2 \times Lap(2/\epsilon)$. Thus, it guarantees $\frac{\epsilon}{2}$-differential privacy.

Since Algorithm 1 performs exactly the same sequence of operations as the single-party algorithm in a distributed setting where D_1 and D_2 are kept locally, it is also ϵ-differentially private. □

Proposition 2. (Security) *Algorithm 1 is secure under the semi-honest adversary model.*

Proof. The security of Algorithm 1 depends on the following steps where the parties exchange information:

- Algorithm *DEM* (Line 5): The privacy proof of DEM is as follows:
  - Circuit `MAX`: It can be evaluated securely [24]. Parties input their local counts $|T(a_j, c_i)|$ and receive the random share of the MAX value.
  - Oblivious Polynomial Evaluation: It has been proven to be secure [28].
  - Random Value Protocol (RVP): It has proven to be secure [27].
  - Circuits `SUM` and `COMPARISON`: Similarly, these circuits can be evaluated securely [24].

  Since, all the above protocols produce random shares and proved to be secure, DEM is also secure due to the composition theorem [24].

- *Exchanging noisy counts (Line 12)*: Each party initially adds Laplace noise to its local count and then exchange the noisy count with the other party. This does not violate differential privacy because the noisy count is already private according to Laplace mechanism [12].

Therefore, due to Composition Theorem [24], Algorithm 1 is secure. □

Proposition 3. (Complexity) *The encryption and the communication costs of Algorithm 1 are bounded by $O(hk \log R)$ and $O(hk \log RK)$, respectively.*

Proof. Distributed exponential mechanism (Algorithm 2) dominates the overall complexity of Algorithm 1. The complexity of DEM is computed as follows:

- Circuit `MAX`: This circuit is composed of simple *add* and *compare* operations and thus can be implemented by the number of gates linear to the input size of the circuit. The input includes $m \times l$ local counts $|T(a_j, c_i)|$ and these values are of size at most $\log |D|$. Hence, the encryption and the communication complexity of `MAX` are bounded by $O(ml \log |D|)$ and $O(ml \log |D|K)$, respectively, where K is the length of the key for a pseudorandom function [29]. The `MAX` protocol is called at most k times. Therefore, the encryption and the communication costs are $O(kml \log |D|)$ and $O(kml \log |D|K)$, respectively.
- Oblivious Polynomial Evaluation: This protocol involves the private evaluation of a polynomial of degree w. Thus, the encryption and the communication complexity are bounded by $O(w)$ and $O(we)$, where e is the length of an encrypted element [30]. This protocol is also called k times. Therefore, the encryption and the communication cost are $O(kw)$ and $O(kwe)$, respectively.
- Random Value Protocol (RVP): The costs of RVP are negligible and therefore they are ignored.
- Circuit `SUM` and `COMPARISON`: The analysis is similar to `MAX` circuit. The encryption and the communication complexity of both the circuits are bounded by $O(k \log R)$ and $O(k \log RK)$, where $R = \left\lfloor \texttt{exp}(\frac{\epsilon' u_x}{2\Delta \mathrm{u}}) \times 10^s \right\rfloor$.

Both the parties execute DEM (Algorithm 2) h times to select the winner candidates. Note that Lines 1-12 of Algorithm 2 are not executed in every iteration. Rather, these lines are only invoked once for each candidate. Hence, the overall encryption and communication costs are $O(\max\{kml \log |D|, kw, hk \log R\})$ and $O(\max\{kml \log |D|K, kwe, hk \log RK\})$, respectively. Since the value of R is usually very large, the encryption and communication costs can be defined as $O(hk \log R)$ and $O(hk \log RK)$, respectively. □

7 Performance Analysis

In this section, we evaluate the scaling impact on the data quality in terms of classification accuracy. Moreover, we estimate the computation and the communication costs of Algorithm 1. We employ the publicly available data set *Adult*; a real-life census data set that has been used for testing many anonymization

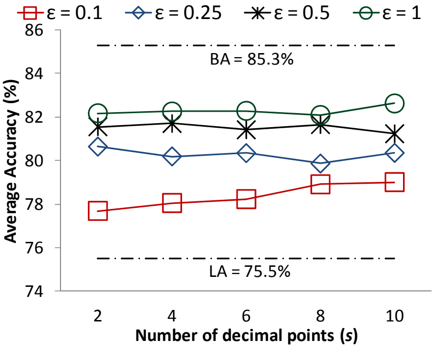

Fig. 2. Classification Accuracy for Different Scaling

algorithms [18,5,6]. It has 45, 222 census records with 6 numerical attributes, 8 categorical attributes, and a binary class column representing two income levels, ≤50K or >50K. All experiments are conducted on an Intel Core *i*7 2.7GHz PC with 12GB RAM.

7.1 Experiments

To evaluate the impact on classification quality, we divide the data into training and testing sets. First, we apply our algorithm to anonymize the training set and to determine the $\cup Cut_i$. Then, the same $\cup Cut_i$ is applied to the testing set to produce a generalized testing set. Next, we build a classifier on the anonymized training set and measure the Classification Accuracy (CA) on the generalized records of the testing set. For classification models, we use the well-known C4.5 classifier [31]. To better visualize the cost and benefit of our approach, we provide additional measures: (1) Baseline Accuracy (BA) is the classification accuracy measured on the raw data without anonymization; (2) BA - CA represents the cost in terms of classification quality for achieving a given ϵ-differential privacy requirement; (3) Lower bound Accuracy (LA) is the accuracy on the raw data with all attributes (except for the *class* attribute) removed and (4) CA - LA represents the benefit of our method over the naive non-disclosure approach.

Fig. 2 depicts the classification accuracy CA for the utility function `Max` where the privacy budget $\epsilon \in \{0.1, 0.25, 0.5, 1\}$ and the number of considered digits after the decimal point $2 \leq s \leq 10$ (i.e., scaling as described in Section 5). The BA and LA are 85.3% and 75.5%, respectively, as shown in the figure by the dotted lines. We use 2/3 of the records (i.e., 30,162) to build the classifier and measure the accuracy on the remaining 1/3 of the records (i.e., 15060). For each experiment, we execute 10 runs and average the results over the runs. The number of specializations h is 10 for all the experiments. For $\epsilon = 1$ and $s = 10$, BA - CA is around 2.6% whereas CA - LA is 7.1%. For $\epsilon = 0.5$, BA - CA spans from 3.6% to 4%, whereas CA - LA spans from 5.7% to 6.2%. However, as ϵ decreases to 0.1, CA quickly decreases to about 79% (highest point), the cost increases to about 6.5%, and the benefit decreases to about 3.3%. The experimental result demonstrates that the classification accuracy is insensitive to the scaling (the number of considered digits after the decimal points) for the Max function. This is because the value of $\mathtt{exp}(\frac{\epsilon'}{2\Delta\mathrm{u}}\mathrm{u})$ is large due to the score of the Max function which is usually a large integer. Therefore, scaling has hardly any impact on the data utility.

7.2 Cost Estimates

Most of the computation and the communication of Algorithm 1 take place during the execution of the DEM (Line 5). The runtime of the other steps is less than 30 seconds for *Adult* dataset. Hence, we only elaborate the runtime of the DEM. As discussed in Section 6, the computation and the communication complexity of the distributed exponential mechanism are dominated by the cost of the SUM (Line 13) and COMPARISON (Line 15) circuits. In the following, we provide an estimate for the computation and the communication costs of evaluating the SUM and COMPARISON circuit. Here, we assume that P_1 encodes and P_2 evaluates the encrypted circuit. The roles of P_1 and P_2 can be swapped.

Computation. The cost of an encryption is denoted by C_m which is 0.02 second for 1024-bit numbers on a Pentium III processor [28]. For both the circuits, P_2 needs to execute a 1-out-of-2 oblivious transfer protocol to get the corresponding encryption key for its input bits. This is the major computational overhead of the distributed exponential mechanism. The computation cost of an oblivious transfer protocol is roughly equal to the cost of a modular exponentiation, which is C_m. Therefore, the computation overhead is equal to the number of input bits of P_2 times C_m. Each input of the circuit is bounded by $\lceil \log_2 R \rceil$ bits, where $R = \left\lfloor \mathtt{exp}(\frac{\epsilon'}{2\Delta\mathrm{u}} u(D, v_i)) \times 10^s \right\rfloor$.

$$\begin{aligned}\lceil \log_2 R \rceil &= \left\lceil \log_2 \left(\left\lfloor \mathtt{exp}(\frac{\epsilon'}{2\Delta\mathrm{u}}) \times 10^s \right\rfloor \right) \right\rceil \\ &= \left\lceil \frac{\frac{\epsilon'}{2\Delta\mathrm{u}}}{\ln 2} + \log_2 10^s \right\rceil \\ &= \left\lceil \frac{\frac{\epsilon'}{2\Delta\mathrm{u}}}{\ln 2} + (3.3219 \times s) \right\rceil\end{aligned}$$

Here, $\Delta\mathrm{u} = 1$, $\epsilon' = \frac{1}{2(6+2\times 10)} = 0.02$, $u(D, v_i)$ is bounded by the number of the records $|D| = 30162$ for Max function, and $s = 10$ suffices the desired accuracy. Hence, we have $\lceil \log_2 R \rceil = 469$ bits. The input size of P_2 is $O(k \log R)$ bits, where the constant is fairly small. Here, k is the total number of candidates which is 24 at most for *Adult* data set. Thus, the cost is $k \times \lceil \log_2 R \rceil \times C_m$

$= 24 \times 469 \times 0.02s \approx 225$ seconds. As mentioned in Section 6, there are at most h invocations of these circuits. Here, h is the number of specializations which is set to 10. Hence, the total computational cost is $h \times 225 \approx 37.5$ mins .

Communication. P_1 needs to send a table of size $4K$ for each gate of the `SUM` and `COMPARISON` circuit, where we assume the key size K is 128 bits. This is the major communication overhead of the distributed exponential mechanism. Since these circuits only use addition and comparison operations, the total number of gates needed to implement these circuits are $O(k \log R)$. Thus, the number of gates, $T_g \approx 24 \times 469 = 11256$. Therefore, the communication cost of sending the tables is $h \times 4K \times T_g \approx 5.76 \times 10^7$ bits, which takes approximately 37.3 seconds using a T1 line with 1.544 Mbits/second bandwidth.

Remark. Our estimation ignores the computational cost of evaluating the circuit and the communication cost of the oblivious transfer protocol. The evaluation of the circuit involves decrypting a constant number of ciphertexts (symmetric encryption) for every gate which is very efficient compared to oblivious transfer (modular exponentiations) since the number of gates of the circuit is linear to the number of input bits. Also, the communication cost of the obvious transfer protocol is negligible compared to the cost of sending the tables.

8 Discussion

Is differential privacy good enough? What changes are required if there are more than two parties? Can the algorithm be easily adapted to accommodate a different utility function? In this section, we provide answers to these questions.

Differential privacy. Differential privacy is a strong privacy definition. However, Kifer and Machanavajjhala [32] have shown that if the records are not independent or an adversary has access to aggregate level background knowledge about the data, then privacy attack is possible. In our application scenario, each record is independent of each other and we assume that no deterministic statistics of the raw database have ever been released. Hence, differential privacy is appropriate for our problem.

More than two parties. The proposed algorithm is only applicable for the two-party scenario because the distributed exponential algorithm, and the other primitives (e.g., random value protocol) are limited to two-party scenario. The proposed algorithm can be extended for more than two parties by modifying all the subprotocols while keeping the general top-down structure of the algorithm as it is.

Other Utility functions. For each new utility function, we only need to devise an algorithm to calculate the utility function. Hence, we only have to change Algorithm 3 to adapt our approach for other utility function.

9 Conclusion

In this paper, we have presented a two-party differentially-private data release algorithm for horizontally-partitioned data for the non-interactive setting. We have shown that the proposed algorithm is differentially private and secure under the security definition of semi-honest adversary model. Moreover, we have experimentally evaluated the data utility of the algorithm. An intersecting research direction, as a future work, is devising different heuristics for different data mining tasks.

Acknowledgments. We sincerely thank the reviewers for their valuable comments. The research described in this paper is part of the project in cloud computing security and privacy with Ericsson Canada and Alcatel Lucent, funded by an NSERC Strategic Grant and NSERC Canada Graduate Scholarships.

References

1. Mohammed, N., Fung, B.C.M., Hung, P.C.K., Lee, C.: Centralized and distributed anonymization for high-dimensional healthcare data. ACM Transactions on Knowledge Discovery from Data (TKDD) 4(4), 18:1–18:33 (2010)
2. Jurczyk, P., Xiong, L.: Distributed anonymization: Achieving privacy for both data subjects and data providers. In: Proceedings of the Annual IFIP WG 11.3 Working Conference on Data and Applications Security, DBSec (2009)
3. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Transaction on Knowledge and Data Engineering (TKDE) (2001)
4. Sweeney, L.: k-anonymity: A model for protecting privacy. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems (2002)
5. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: ℓ-diversity: Privacy beyond k-anonymity. ACM Transactions on Knowledge Discovery from Data (TKDD) (2007)
6. Wang, K., Fung, B.C.M., Yu, P.S.: Handicapping attacker's confidence: An alternative to k-anonymization. Knowledge and Information Systems (KAIS) 11(3), 345–368 (2007)
7. Wong, R.C.W., Fu, A.W.C., Wang, K., Pei, J.: Minimality attack in privacy preserving data publishing. In: Proceedings of the International Conference on Very Large Data Bases (VLDB) (2007)
8. Zhang, L., Jajodia, S., Brodsky, A.: Information disclosure under realistic assumptions: Privacy versus optimality. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS) (2007)
9. Cormode, G., Srivastava, D., Li, N., Li, T.: Minimizing minimality and maximizing utility: Analyzing methodbased attacks on anonymized data. In: Proceedings of the International Conference on Very Large Data Bases (VLDB) (2010)
10. Ganta, S.R., Kasiviswanathan, S., Smith, A.: Composition attacks and auxiliary information in data privacy. In: Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD) (2008)
11. Kifer, D.: Attacks on privacy and de finetti's theorem. In: Proceedings of the ACM Conference on Management of Data (SIGMOD) (2009)

12. 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)
13. Mohammed, N., Chen, R., Fung, B.C.M., Yu, P.S.: Differentially private data release for data mining. In: Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD) (2011)
14. Fung, B.C.M., Wang, K., Chen, R., Yu, P.S.: Privacy-preserving data publishing: A survey of recent developments. ACM Computing Surveys 42(4), 1–53 (2010)
15. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Mondrian multidimensional k-anonymity. In: Proceedings of the IEEE International Conference on Data Engineering (ICDE) (2006)
16. Fung, B.C.M., Wang, K., Yu, P.S.: Anonymizing classification data for privacy preservation. IEEE Transactions on Knowledge and Data Engineering (TKDE) 19(5), 711–725 (2007)
17. Xiao, X., Wang, G., Gehrke, J.: Differential privacy via wavelet transforms. In: Proceedings of the International Conference on Data Engineering (ICDE) (March 2010)
18. Jiang, W., Clifton, C.: A secure distributed framework for achieving k-anonymity. Very Large Data Bases Journal (VLDBJ) 15(4), 316–333 (2006)
19. Mohammed, N., Fung, B.C.M., Debbabi, M.: Anonymity meets game theory: secure data integration with malicious participants. Very Large Data Bases Journal (VLDBJ) 20(4), 567–588 (2011)
20. Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., Zhu, M.Y.: Tools for privacy preserving distributed data mining. ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD) Explorations Newsletter 4(2), 28–34 (2002)
21. Roth, A., Roughgarden, T.: Interactive privacy via the median mechanism. In: Proceedings of the ACM Symposium on Theory of Computing (STOC) (2010)
22. Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., Naor, M.: Our Data, Ourselves: Privacy Via Distributed Noise Generation. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 486–503. Springer, Heidelberg (2006)
23. McSherry, F., Talwar, K.: Mechanism design via differential privacy. In: Proceedings of the IEEE Symposium on Foundations of Computer Science (2007)
24. Goldreich, O.: Foundations of Cryptography, vol. 2. Cambridge University Press (2001)
25. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
26. Yao, A.C.: Protocols for secure computations. In: Proc. of the IEEE Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS) (1982)
27. Bunn, P., Ostrovsky, R.: Secure two-party k-means clustering. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS), 486–497 (2007)
28. Naor, M., Pinkas, B.: Efficient oblivious transfer protocol. In: Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) (2001)
29. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game - a completeness theorem for protocols with honest majority. In: Proceedings of the ACM Symposium on the Theory of Computing (STOC) (1987)
30. Lindell, Y., Pinkas, B.: Privacy preserving data mining. Journal of Cryptology 15(3), 177–206 (2002)
31. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
32. Kifer, D., Machanavajjhala, A.: No free lunch in data privacy. In: Proceedings of the ACM Conference on Management of Data (SIGMOD) (2011)