Information Sciences 444 (2018) 89–104

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Differentially private Naive Bayes learning over multiple data sources

Tong Li[a,b], Jin Li[a,*], Zheli Liu[b], Ping Li[a], Chunfu Jia[b,c]

[a] School of Computer Science, Guangzhou University, Guangzhou, China
[b] College of Computer and Control Engineering, Nankai University, Tianjin, China
[c] Information Security Evaluation Center of Civil Aviation, Civil Aviation University of China, Tianjin, China

ARTICLE INFO

Article history:
Received 21 October 2017
Revised 20 February 2018
Accepted 26 February 2018
Available online 27 February 2018

Keywords:
Privacy-preserving
Naive Bayes classification
Differential privacy

ABSTRACT

For meeting diverse requirements of data analysis, the machine learning classifier has been provided as a tool to evaluate data in many applications. Due to privacy concerns of preventing disclosing sensitive information, data owners often suppress their data for an untrusted trainer to train a classifier. Some existing work proposed privacy-preserving solutions for learning algorithms, which allow a trainer to build a classifier over the data from a single owner. However, they cannot be directly used in the multi-owner setting where each owner is not totally trusted for each other. In this paper, we propose a novel privacy-preserving Naive Bayes learning scheme with multiple data sources. The proposed scheme enables a trainer to train a Naive Bayes classifier over the dataset provided jointly by different data owners, without the help of a trusted curator. The training result can achieve ϵ-differential privacy while the training will not break the privacy of each owner. We implement the prototype of the scheme and conduct corresponding experiment.

1. Introduction

Nowadays, the machine learning classifier, as a concrete implementation of classification [5], is becoming an effective tool in data analysis, which has facilitated applications in many areas including economic prediction, risk assessment, and spam detection. Given a queried instance **x**, a trained classifier model **W** can be run to output a prediction that indicates the class which the instance belongs to. To make the prediction accuracy, it is desirable for a trainer to train the classifier over sufficient samples collected from various sources. Unfortunately, privacy and security concerns of personal information arise in recently years, that is, data owners can hard allow untrusted entities to get access to their sensitive data, which restricts trainer's centralization of sample datasets. For example, a medical researcher wants to build a classifier which can be used to classify symptoms according to patients' health records. In this scenario, the researcher is a trainer while health records can be seen as training samples that contains patients' individual sensitive information that should not be revealed by the researcher. It is urgent for this trainer to address a paradox between preserving privacy of patients and keeping the availability of samples.

As a solution, the notion of differential privacy [12,14] has been proposed to provide a privacy guarantee for an analyzed dataset, even in the situation that a trainer hold some prior knowledges about the dataset. Implementing an appropriate

* Corresponding author.
E-mail addresses: litongziyi@mail.nankai.edu.cn (T. Li), lijin@gzhu.edu.cn (J. Li), liuzheli@nankai.edu.cn (Z. Liu), liping26@mail2.sysu.edu.cn (P. Li), cfjia@nankai.edu.cn (C. Jia).

https://doi.org/10.1016/j.ins.2018.02.056
0020-0255/

private mechanism on analysis results of an algorithm can ensure that the algorithm will produce very similar outputs when working on two adjacent datasets. Thus, we can adopting similar methods for the privacy-preserving machine learning to build a classifier over a sample dataset without disclosing any single individual sample in the set. Aiming at a specific learning algorithm, some privacy-preserving schemes [1,48] are developed to achieve the goal of differential privacy.

However, the samples for training are usually collected from multiple data owners rather than a single source. In the multi-owner setting, owners are untrusted for each other, that is, they all try to reveal the data contents of other owners and hope to protect their own privacy. As a result, it is infeasible to directly depute the task of implementing private mechanisms to any owner. A trivial way to solve this training problem is to introduce a trusted curator whose tasks are collecting samples, training a model, and performing the mechanism. Unfortunately, such a curator usually cannot be established in a real world application. This motivates us to design a multi-owner learning scheme without the help of a trusted third party.

Although some powerful cryptographic tools (e.g., fully homomorphic encryption or secure multi-party computation protocol) can be used for these training computations, they are too heavy to be adopted for applications. To carry out multi-owner training results achieving differential privacy in a practical way, we should face challenges from two aspects. On one hand, the sensitivity of the learning function, which is a significant parameter in the mechanism, is related to the whole sample set. That means deriving it without knowing the set is very hard. On the other hand, collecting samples will certainly obtain some of their important characteristics such as counts. So, if there is an untrusted curator, how to prevent it revealing the characteristics is another problem.

1.1. Contribution

To address the problem above, in this paper, we propose a privacy-preserving machine learning scheme in the multi-owner setting for a simple but highly effective classification, the Naive Bayes (NB) classification. The proposed scheme enables a trainer, which is called data receiver, to build a NB classifier over the dataset contributed jointly by different data providers, without the help of a trusted curator. The NB classifier model is able to meet the requirements of ϵ-differential privacy.

In summary, we present the main contributions of this paper as follows.

- To achieve the goal of training, we design novel algorithms for aggregating each data provider's data and implementing differentially private training over these data. These algorithms do not involve heavy cryptographic tools and any trusted curator but can protect each provider's individual privacy.
- Different from existed works, the aggregation method that we design in the proposed scheme can hide some statistics information (e.g., the number of total samples). Furthermore, it makes the scheme guarantee the ownership privacy, which means others cannot know whether a provider holds a sample that contains a specific attribute value.
- We implement the prototype of our scheme and conduct experiments on the LAN server over datasets from UCI Machine Learning Repository [15]. The result shows that the proposed scheme is practical.

1.2. Organization

The rest of this paper proceeds as follows. In Section 2, related works are presented. Some preliminaries of this paper are briefly introduced in Section 3. In Section 4, we state the architecture of our scheme and give the threat model, requirements, and problems. The concrete construction of the scheme are proposed in Section 5. The security analysis for the proposed scheme are presented in Section 6. In Section 7, we present the implementation and evaluate the results of experiments. Finally, we make a conclusion in Section 8.

2. Related work

2.1. Privacy-preserving machine learning

There have been some existing works that address privacy problems for machine learning algorithms, which provide solutions for protecting the privacy of data providers. According to focused problems, the scenarios are mainly divided into two categories. One is classifying a data instance, and the other is training a classifier over the datasets. Our work pays attention on the latter one.

Classification. We briefly describe this category. In the privacy-preserving classification, Barni et al. [4] proposed the secure evaluation based on garbled circuits [22,52] for linear branching programs and neural networks. To deal with a Naive Bayes classifier, some literatures presented schemes for protecting the confidentiality of queried instances [25,46,53]. Based on comparison blocks and argmax blocks, Bos et al. [6] constructed a series of secure classification protocols which contains the NB classification.

Training. The system model of the training usually falls into either *collaborative* or *client-server* style. The addressed problem in the former is how to enable several data providers to collaboratively train a machine learning classifier over all their datasets but not break the privacy of each one. In this setting, each data provider also plays a role of the trainer. The existing works considered this issues for machine learning algorithms, such as the neural network [3,9,40,42,43,54], decision

trees [38], k-means clustering [24], and NB classification [16,47]. Some of these works introduce a trusted third party for authorizations, which is a usual method in the outsourcing storage system [10,11,28–31,33,34,44,49,51]. Alternatively, the latter system is to enable a trainer (i.e., the *server* who receives training results) to train the classifier over the datasets collected from data providers (i.e., the *clients*) without revealing the content of the datasets. In this setting, some general solutions based on the fully homomorphic encryption (FHE) technique [7,17,18] were presented by [19,50]. Nevertheless, the FHE scheme is currently inefficient for practical applications. Other works presented solutions for specific data analysis tasks and machine learning algorithms including statistics query [23,32,37], SVM [8,27], and deep learning [35,36,45].

We concern about The problem of the NB learning in a system architecture similar to a *client-server* one. Besides traditional privacy-preserving issues, we also address the problem in the view of the differential privacy.

2.2. Differential privacy in machine learning

Dwork et al. [14] have proposed the rigorous privacy definition of differential privacy, which is a significant work on privacy protection for data release. The differential privacy is defined in terms of that the access of a dataset is unrelated to the presence or absence of any individual. It has been extended to various applications of privacy-preserving data analysis such as recommender system [39] and access log statistics [26].

There have also been works that proposed differentially private solutions for data analysis tasks. The solution approaches are to implement a mechanism on the queried functions according to requirements of a specific task, and then give a way to mitigate the impact of the noise from the learning results [2]. The differentially private learning schemes follow the same line of the approaches to achieve quantifiable notion of privacy. These approaches include adding noises directly on the raw data [13] and conducting particular randomization mechanism [20] to query results. Recently, Abadi et al. [1] proposed a deep learning scheme with differential privacy, in which the mechanism is performed on the result of each learning round.

For the privacy-preserving NB classification with differential privacy, there are two existed work is most closely related to ours. One is the work [48] in which developed differential private scheme to training the NB classifier from horizontally partitioned data. However, this scheme orients to a single data provider. Without a trusted curator, it cannot be directly used in the multi-owner setting where each owner is not totally trusted for each other. The other work [21] proposed a scheme for the multi-owner setting. Nevertheless, different from ours, this work set statistics information such as the size of the dataset as public, and the trainer can easily know whether an data owner holds a sample contains a specific value. Both of these situations will break owner's privacy. Furthermore, the algorithm of the scheme for vertically partitioned data is trivial. Our work aims at enabling a trainer to training a NB classifier that achieves ϵ-differential privacy in the multi-owner setting, while guaranteeing statistic privacy and ownership privacy defined in Section 4.3.

3. Preliminaries

3.1. Naive Bayes learning

3.1.1. Classification

For a input vector $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$, the machine learning classification aims at classifying $\mathbf{x}$ into a class in a discrete possible category set$\{c_1, \ldots, c_m\}$. A classifier is defined by a model $\mathbf{W}$ and a classification function $C_{\mathbf{W}} : \mathbb{R}^d \to \{c_1, \ldots, c_m\}$. To predict the class which $\mathbf{x}$ belongs to, the function $C_{\mathbf{W}}(\cdot)$ should be evaluated on $\mathbf{x}$. The output $c_{i_0} \leftarrow C_{\mathbf{W}}(\mathbf{x})$ is the classification result.

According to the Bayes decision rule, the classification function of a Naive Bayes classifier is to choose the class with the highest posterior probability. It should compute $i_0 \leftarrow \arg\max_{i\in[m]} p(C = c_i|\mathbf{X} = \mathbf{x}) = \arg\max_{i\in[m]} p(C = c_i, \mathbf{X} = \mathbf{x})$, where the factor $p(\mathbf{X} = \mathbf{x})$ with a fixed $\mathbf{x}$ is omitted.

Based on the attribute conditional independence assumption that each attribute (component) of $\mathbf{x}$ is conditionally independent, each class-conditional probability $p(C = c_i, \mathbf{X} = \mathbf{x})$ is equal to the factorization $p(C = c_i)\prod_{j=1}^{d} p(X_j = x_j|C = c_i)$.

Therefore, for a Naive Bayes classifier, the model $\mathbf{W}$ is composed of a set of probabilities:

- prior probability $\{p(C = c_1), \ldots, p(C = c_m)\}$ that represents the occurring probability of each class c_i;
- class-conditional probability $\{\{\{p(X_j = v|C = c_i)\}_{v\in V_j}\}_{j=1}^{d}\}_{i=1}^{m}$ that represents the probability of that the jth attribute of $\mathbf{x}$ is v when $\mathbf{x}$ belongs to class c_i, where V_j is the domain of X_j.

3.1.2. Training

The prior probabilities and the class-conditional probabilities in model $\mathbf{W}$ need to be trained from the sample dataset. Typically, the probabilities are estimated by simply counting the frequencies of samples in the set.

The NB learning is a supervised learning that each sample can be depicted as $\langle \mathbf{x}, c\rangle$ where $\mathbf{x}$ is the feature vector and c is the value denoting the class of $\mathbf{x}$. In the proposed scheme, the attributes of $\mathbf{x}$ are nominal. Let n be the total number of samples and n_i be the number of samples whose class label is c_i. Each probability can be obtained as follows:

- prior probability $p_i = p(C = c_i) = \frac{n_i}{n}$;

- class-conditional probability $p_{i,j}(v) = p(X_j = v|C = c_i) = \frac{n_{i,j}(v)}{n_i}$ where $n_{i,j}(v)$ is the number of samples whose jth attribute is v and class label is c_i.

3.2. Differential privacy

Differential Privacy provides a formal and quantifiable privacy guarantee irrespective of an adversary's background knowledge and available computational power. The differential Privacy is actually a condition on the data release mechanism but not on the dataset. A randomized algorithm is considered to be differentially private if for any pair of neighboring inputs, the probability of generating the same output is within a small multiple of each other for the entire output space. This means that for any two datasets which are close to one another, a differentially private algorithm will behave approximately the same on both datasets. This notion provides sufficient privacy protection for users regardless of the prior knowledges possessed by the adversaries.

Definition 1 (ϵ - differential privacy)**.** A randomization algorithm $\mathcal{M}$ satisfies ϵ - differential privacy if for any two neighboring datasets D_1 and D_2, and any output $D \in Range(\mathcal{M})$, we have $e^{-\epsilon} \leq \frac{Pr[\mathcal{M}(D_1)]=D}{Pr[\mathcal{M}(D_2)]=D} \leq e^{\epsilon}$.

Definition 2 (Sensitivity)**.** For a function $f : D \to \mathbb{R}^d$ over the input datasets, the sensitivity of f is $\delta f = \max||f(D_1) - f(D_2)||$ for any two neighboring datasets D_1 and D_2.

The lower sensitivity queries with, the better tolerate the data modifications from added noise is.

Definition 3 (The Laplace distribution)**.** The Laplace distribution (centred at 0) with scale b is the distribution with probability density function:

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

Laplace distribution with scale b can be simply denoted as $Lap(b)$.

Definition 4 (The Laplace mechanism)**.** Given any function $f : \mathbb{N}^{|\chi|} \to \mathbb{R}^k$, the Laplace mechanism is defined as:

$\mathcal{M}_L(x, f(\cdot), \epsilon) = f(x) + (Y_1, \ldots Y_k)$

where each noise Y_i is i.i.d. random variable drawn from $Lap(\delta f/\epsilon)$.

Theorem 1. *The Laplace mechanism preserves ϵ-differential privacy.*
The proof of Theorem 1 is presented in [14] which shows that releasing a function $f(\cdot)$ with Laplace mechanism $\mathcal{M}_L$ can satisfy privacy. In this paper, we add Laplace noises to training results with standard deviation $\delta f/\epsilon$.

3.3. Additive homomorphic encryption

Homomorphic encryption enables the computations of plaintexts to be performed on the corresponding ciphertexts without revealing the underlying plaintexts. A public-key encryption scheme $\mathcal{AHE}$ that supports addition operations is additively homomorphic. Given two encrypted messages $\mathcal{AHE}.Enc(a)$ and $\mathcal{AHE}.Enc(b)$ that are encrypted by using the same public key, there exists a public-key operation $\oplus$ such that $\mathcal{AHE}.Enc(a) \oplus \mathcal{AHE}.Enc(b)$ is the encryption of $a + b$. Let $\mathcal{AHE}.Enc(a)$ be a ciphertext of a plaintext a and c be a constant value. The multiplication ciphertext $\mathcal{AHE}.Enc(ca)$ can be implement by $\mathcal{AHE}.Enc(a) \oplus \ldots \oplus \mathcal{AHE}.Enc(a) = (\mathcal{AHE}.Enc(a))^c$. Considering the practicality, the AHE cryptosystem we choose in this paper is the Paillier cryptosystem [41].

3.4. Notations

In this paper, we denote a set as $\{\cdot\}$ and the integer set $\{1, \ldots, n\}$ as $[n]$. The simple combination of several elements is denoted as $\langle\cdot\rangle$ A column vector is denoted as a bold-type letter (e.g., $\mathbf{x}= (x_1, \ldots, x_d)$).

The plaintext space of the Paillier cryptosystem $\mathcal{P}$ is $\mathbb{Z}_N$ where N is the modulus. The corresponding ciphertext space is $\mathbb{Z}_{N^2}$. We use $[[a]]$ to represent the Paillier ciphertext encrypted from a. For distinguishing different public parameters, we add different constant subscripts to the encryption symbol (e.g., $[[\cdot]]_1$ is the ciphertext generated via $\mathcal{P}_1$). For computations of the Paillier ciphertext, we define some operators. The ciphertext addition of $[[a]]$ and $[[b]]$ is denoted as $[[a]] \cdot [[b]]$ that is $[[a + b]]$. The ciphertext multiplication of $[[a]]$ and a plaintext c is denoted as $[[a]]^c$ that is $[[ca]]$.

4. Problem statement

4.1. Architecture

Imagine a multi-owner application in practical. A medical researcher wants to build a Naive Bayes classifier which can be used to predict the health condition of a patient in holistic assessments. Since the NB classifier takes a patient's health record as the input, the researcher should take a paradox into consideration. On one hand, in order to train a classifier that makes the prediction more accurate, it is necessary for the researcher to collect records as raw as possible from patients. On the other hand, the researcher have the duty to keep the privacy of each patient when providing records. That means, a

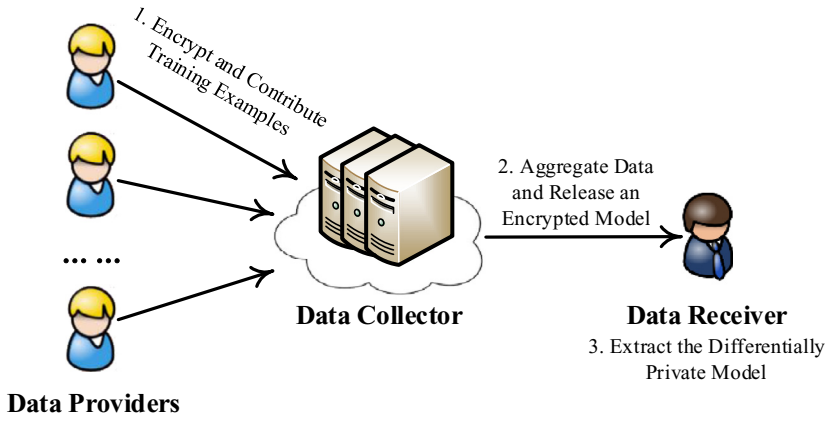


Fig. 1. Architecture of our scheme.

desirable secure scheme should be design to enable the researcher to perform training over the record sets without revealing too much information about any single individual in them.

In this application, health records are training samples, each patient can be seen as a data provider, and the researcher can be seen as a data receiver. As described in Section 3.2, to make the trained classifier differentially private via a private mechanism, the noises that is added to statistics (e.g., counts) should be related to the whole dataset aggregated from the provided datasets. However, for a provider, other providers are not totally trusted such that the contents of his/her own dataset must not be known by others. Thus, we introduce a untrusted data collector to undertake the aggregation task. Consequently, in the view of a provider, his/her privacy should be protected against the receiver, the collector, and other providers via some techniques.

An architecture that contains three entities is shown in Fig. 1. Each entity is described as follows.

- *Data provider*. The sample dataset is jointly owned by s data providers $\{O_1, \ldots, O_s\}$. A data provider O_k has a dataset partitioned from the whole database by some means. Each O_k encrypts its dataset D_k into e_k for protecting the confidentiality and generates the auxiliary information aux_k for aggregating data. Then, O_k contributes e_k and aux_k to the data collector *DC*.
- *Data collector*. The task of the data collector *DC* is to aggregate ciphertext $\{e_1, \ldots, e_s\}$ and perform differentially private mechanism secretly. Then, *DC* submits an encrypted model $E(\mathbf{W})$ to the data receiver *DR*.
- *Data receiver*. The data receiver *DR* can extract a NB classifier model $\mathbf{W}$ from a submission received from *DC*. The model $\mathbf{W}$ can meet the requirement of the differential privacy.

4.2. Threat model

Typically, we assume that the entities in our scheme are "honest-but-curious". That is, they will follow our proposed algorithms, but try to find out as much secret information as possible based on their possessions. Two categories of adversaries are considered, which are an external attacker and an internal attacker.

An external adversary may obtain some information (e.g., an encrypted part of a NB classifier) in the communication via public channels. It can play a role of an external eavesdropper.

An internal adversary could refer to a data provider *O*, the data collector *DC*, or the data receiver *DR*. The goal of the type-i adversary *O* is to extract the information of partitioned datasets not owned by it, while the goal of the type-ii *DC* and type-iii adversary *DR* is to reveal the information of each partitioned dataset. In addition, we do not allow the collusion between *DC* and *DR*.

4.3. Requirements and problems

In this paper, the security goals which we aim to achieve is the privacy of the whole dataset.

- *ϵ-differential privacy*. It is the basic requirement of the proposed scheme. The result of "queried function" is the NB classifier model $\mathbf{W}$ obtained by the receiver *DR*, in which each probability (i.e., p'_i and $p'_{i,j}(v)$) should meet ϵ-differential privacy.
- *Statistics privacy*. A privacy-preserving scheme ensures that the content of a dataset is not revealed by entities other than it owner. Besides, the *statistics privacy* requires the statistics information of the dataset, including n, each n_i, and each $n_{i,j}(v)$, should not be disclosed by neither external adversaries nor internal adversaries.
- *Ownership privacy*. Furthermore, the adversaries are not allowed to learn whether a provider holds a sample that contains a specific attribute value and a specific class label.

There are some key problems should be taken into consideration when design the scheme.

- *Sensitivity deriving*. Results of the training function are a set of probabilities, and the function's sensitivity cannot be derived directly. Alternatively, since the sensitivity is related to the whole dataset which is jointly owned by the providers, the deriving without breaking the privacy of each one is a key problem.
- *Mechanism implementation*. After the sensitivity is carried out, how to privately implement Laplace mechanism on each probability should also be taken into consideration.
- *Aggregate method*. Aggregating provided data to a model **W** is undertaken by the untrusted collector *DC*. So, the aggregate method should be designed to meet all above requirements.

5. The proposed scheme

5.1. Overview

In this section, we give constructions of our privacy-preserving Naive Bayes learning scheme. Firstly, we reformulate some details of the partition in the scheme.

As is shown in Section 3, each sample can be depicted as $\langle \mathbf{x}, c\rangle$ where the feature vector $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ and the class label $c \in \{c_1, \ldots, c_m\}$. The partition of a whole database D can be divided into two categories: horizontal partition and vertical partition.

- *Horizontal partition*. All s data providers jointly hold n samples and the kth data provider O_k holds a subset D_k of D. Each sample in D is in the form as $\langle \mathbf{x}, c\rangle$. In addition, $D_1 \cup \ldots \cup D_s = D$ and $D_1 \cap \ldots \cap D_s = \emptyset$.
- *Vertical partition*. All s data providers jointly hold n samples, and each one has some attributes of these samples. In more detail, each vector $\mathbf{x}$ in D is partitioned into $\{\mathbf{x}_1, \ldots, \mathbf{x}_s\}$ owned by corresponding $O_k (k \in [s])$. Obviously, if class labels are know by each O_k, the solution of training is trivial. Therefore, in this setting, we set the sth provider O_s only has all the labels, while other $(s-1)$ provider jointly hold the n vectors.

In terms of the requirements in Section 4.3, we consider that the proposed scheme is expected to contain procedures: the *Initialization phase*, the *Preparation phase*, the *Aggregation phase*, and the *Extraction phase*. A overview of the procedures is described as follows.

- *Initialization phase*. In this phase, the *system*, which can be an honest dealer, initializes cryptographic systems and generates some public parameters. Each entity will receive some tokens from the *system* for follow-up tasks.
- *Preparation phase*. In this phase, each data provider O_k firstly encrypts his/her "partitioned" set, and then contributes it to the data collector *DC*. Some providers also submit some auxiliary information to *DC*.
- *Aggregation phase*. In this phase, the data collector *DC* aggregates each encrypted data. Then, it uses the auxiliary information to add noise in the aggregated ciphertexts, which functionally equivalent to perform the mechanism that implements differential privacy on the corresponding plaintexts. Next, *DC* releases an encrypted NB classifier model $E(\mathbf{W}')$ for the data receiver *DR*.
- *Extraction phase*. In this phase, using the received tokens the data receiver *DR* recover the trained model **W'** of a Naive Bayes Classifier from $E(\mathbf{W}')$.

Then, we present two constructions for horizontally partitioned dataset and vertically partitioned dataset respectively.

5.2. Sensitivity

As is depicted in Section 3.2, we choose the Laplacian mechanism to preserve the differential privacy in the trained NB classifier. Let $f(\cdot)$ be the queried function. The noise that is added to the result is y meets $y \sim Lap(\delta f/\epsilon)$. Each queried function and its sensitivity are derived as follows.

Functions. In this paper, we mainly consider the processing of categorical attributes. The description in Section 3.1 shows that all possible values for the given categorical attributes are public information (i.e., the jth nominal attribute is $x_j = v \in V_j$ where V_j has been known already). Intuitively, the outputs of training function are the prior probabilities and the class-conditional probabilities. As shown in [48], ϵ-differentially private counts can make the trained NB classifier be preserved ϵ-differential privacy. Hence, the private mechanism is implemented on the counts. For each prior probability $p_i = \frac{n_i}{n}$ and each class-conditional probability $p_{ij}(v) = \frac{n_{ij}(v)}{n_i}$, the corresponding output can be seen as the count n_i and $n_{ij}(v)$.

Sensitivity. As described above, the sensitivity computation can be done on the counts, and the mechanism is conducted on the counts. The presence of a sample can change 1 at most on the count n_i that indicates the number of samples with a class label c_i, which can result the same change on the count $n_{ij}(v)$. Therefore, the sensitivity of each n_i and $n_{ij}(v)$ is 1 for all attribute values and class values. According to *Theorem 1*, each noised count $n'_i = n_i + Lap(\delta f/\epsilon)$ (noised count $n'_{ij}(v) = n_{ij}(v) + Lap(\delta f/\epsilon)$) is ϵ-differentially private. As a result, the Laplace noise is sampled from $Lap(\delta f/\epsilon)$ (i.e., $Lap(1/\epsilon)$).

Encryption details. In this scheme, a noise $y \sim Lap(b)$ is represented as a float. Thus, considering the Paillier cryptosystem can only work with integers, a float number should be converted to an integer by multiplying it with a pre-concert public number l and ceiling ($\lceil \cdot \rceil$). We represent real numbers using IEEE 754 double precision floating point numbers with 52 bits

of precision and adopt the conversion method in [6]. As a result, the converted integer is in $\mathbb{Z}_N$, where N is the modulus for Paillier. The integer will not be in $\mathbb{Z}_N^* \simeq \mathbb{Z}_P^* \times \mathbb{Z}_Q^*$ with negligible probability $(1-\frac{1}{P})(1-\frac{1}{Q})$, where P and Q are the private primes. So Paillier plaintext space $\mathbb{Z}_N^*$ can be approximately seen as $\mathbb{Z}_N$ for convenience.

5.3. Construction for horizontally partitioned dataset

5.3.1. Main idea

Intuitively, using a additive homomorphic cryptosystem to encrypt provider's data can ensure the aggregation task is undertaken over the encrypted data. However, there are still two problems must be solved.

- Each probability in a NB classifier model $\mathbf{W}$ is carried out by a division operation. Since n and each n_i must be kept secret, such a operation cannot directly conducted by the collector *DC*.
- There is a paradox for the aggregation. That is, *DC* can know the noised count such as n'_i via decryption the corresponding aggregation result, but it is not allowed to decrypt each encryption submitted by providers before the aggregation.

For the first problem, we run two Paillier cryptosystems $\mathcal{P}_1$ and $\mathcal{P}_2$. $\mathcal{P}_1$ is used for the collector *DC* to aggregate noised data so that *DC* obtains each n'_i and $n'_{i,j}(v)$ in clear. Therefore, a division operation can be implemented by a multiplication operation over the ciphertext space of $\mathcal{P}_2$. For the second one, we set a series of factors $\{\tau_0, \ldots, \tau_s\}$ that each $\tau_k \in \mathbb{Z}_{N_1^2}$ and $tau_0 \cdot \ldots \cdot \tau_s = 1$ where N_1 is the modulus of $\mathcal{P}_1$. The factor τ_k is issued to the kth provider O_k, while τ_0 is issued to *DC*. As a result, *DC* can only extract the sum of counts, if each O_k multiples his/her encrypted count by τ_k.

5.3.2. Initialization

For the system initialization, according to the secure parameter 1^λ, *system* initializes the Paillier cryptosystems $\mathcal{P}_1$ and $\mathcal{P}_2$, where the key pairs are $\langle pk_1, sk_1\rangle$ and $\langle pk_2, sk_2\rangle$ respectively. Then, *system* generates two sets of factors $\{\tau_{1,0}, \ldots, \tau_{1,s+1}\}$ and $\{\tau_{2,0}, \ldots, \tau_{2,s}\}$. The sets meet that each $\tau_{1,k} \in \mathbb{Z}_{N_1^2}$, each $\tau_{2,k} \in \mathbb{Z}_{N_2^2}$, $\tau_{1,0} \cdot \ldots \cdot \tau_{1,s+1} = 1$, and $\tau_{2,0} \cdot \ldots \cdot \tau_{2,s} = 1$. The public parameters are published as public. The factor $\tau_{1,0}$, $\tau_{2,0}$, and sk_1 are issued to the collector *DC*, while sk_2 is issued to the receiver *DR*, The kth provider O_k will receive $\tau_{1,k}$ and $\tau_{2,k}$, while the token $\tau_{1,s+1}$ is give a member of providers for performing Laplace mechanism. The public factor l is determined in the initialization.

5.3.3. Preparation

Each data provider creates a count table for his/her dataset. Take the kth provider as an example. The partitioned dataset D_k is held by O_k, and contents of the table T_k are maintained as:

- $\{n_i^{(k)}\}_{i=1}^m$, where $n_i^{(k)}$ is the number of samples that contain a class label c_i ;
- $\{\{\{n_{i,j}^{(k)}(v)\}_{v\in V_j}\}_{j=1}^d\}_{i=1}^m$, where $n_{i,j}^{(k)}(v)$ is the number of samples of which jth attribute is $v \in V_j$ and class label is c_i.

Some counts in T_k could be 0. Using N_1 and $\tau_{1,k}$, O_k encrypt T_k into $e_{1,k} = \langle\{e_i^{(1,k)}\}_{i=1}^m, \{\{\{e_{i,j}^{(1,k)}(v)\}_{v\in V_j}\}_{j=1}^d\}_{i=1}^m\rangle = \langle\{\tau_{1,k}[[n_i^{(k)} \cdot l]]_1\}_{i=1}^m, \{\{\{\tau_{1,k}[[n_{i,j}^{(k)}(v) \cdot l]]_1\}_{v\in V_j}\}_{j=1}^d\}_{i=1}^m\rangle$. T_k is also encrypted into $e_{2,k} = \{e_i^{(2,k)}\}_{i=1}^m = \{\tau_{2,k}[[n_i^{(k)}]]_2\}_{i=1}^m$. by using N_2 and $\tau_{2,k}$. Then, O_k contributes $\langle e_{1,k}, e_{2,k}\rangle$ to the collector *DC*.

Next, the holder of $\tau_{1,s+1}$ generates ciphertexts of noises. Considering the sensitivities of queried functions (i.e., both n_i and $n_{ij}(v)$) are 1, the holder can maintain a noise table $Y = \langle\{y_i\}_{i=1}^m, \{\{\{y_{i,j}(v)\}_{v\in V_j}\}_{j=1}^d\}_{i=1}^m\rangle$ for $\langle\{n_i\}_{i=1}^m, \{\{\{n_{i,j}(v)\}_{v\in V_j}\}_{j=1}^d\}_{i=1}^m\rangle$. Each element in Y is independent and has same the distribution $Lap(1/\epsilon)$. The holder also encrypts Y into $e_Y = \langle\{e_i^{(Y)}\}_{i=1}^m, \{\{\{e_{i,j}^{(Y)}(v)\}_{v\in V_j}\}_{j=1}^d\}_{i=1}^m\rangle = \langle\{\tau_{1,s+1}[[y_i \cdot l]]_1\}_{i=1}^m, \{\{\{\tau_{1,s+1}[[y_{i,j}(v) \cdot l]]_1\}_{v\in V_j}\}_{j=1}^d\}_{i=1}^m\rangle$, where l is used to transform a noise to an integer. The encrypted noises e_Y, that can be seen as auxiliary information, are submitted to *DC*. Algorithm 1 and Algorithm 2 depicts the preparation algorithm run by O_k.

5.3.4. Aggregation

After receiving $\{e_{1,k}\}_{k=1}^s$, $\{e_{2,k}\}_{k=1}^s$, and e_Y, *DC* starts to train a differentially private NB classifier over the ciphertext (i.e., $e_{\mathbf{W}}$).

One task of the collector *DC* is to extract $\{n'_i\}_{i=1}{}^m$ and $\{\{\{n'_{i,j}(v)\}_{v\in V_j}\}_{j=1}{}^d\}_{i=1}{}^m$. Obviously, a total count equals the sum of corresponding s counts (e.g., $n_i = \sum_{k=1}^s n_i^{(k)}$). To the begin with, *DC* aggregates to obtain each encrypted noised count as follows.

- For each $[[n'_i \cdot l]]_1$, *DC* computes $\tau_{1,0} \cdot e_i^{(1,1)} \cdot \ldots \cdot e_i^{(1,s)} \cdot e_i^{(Y)} \mathbf{mod} N_1^2$;
- For each $[[n'_{i,j}(v) \cdot l]]_1$, *DC* computes $\tau_{1,0} \cdot e_{i,j}^{(1,1)}(v) \cdot \ldots \cdot e_{i,j}^{(1,s)}(v) \cdot e_{i,j}^{(Y)}(v) \mathbf{mod} N_1^2$.

According to the property of $\mathcal{P}_1$, the correctness of $[[n'_i \cdot l]]_1$ (same as $[[n'_{i,j}(v) \cdot l]]_1$) is shown as:

Algorithm 1 Preparation. (Part 1).

Input:

D_k, the kth horizontally partitioned dataset;
$\tau_{1,k}$ and $\tau_{2,k}$, the kth factor for $\mathcal{P}_1$ encryption and $\mathcal{P}_2$ encryption respectively;
$\tau_{1,s+1}$, the token for adding noises;
ϵ, the privacy parameter for differential privacy;
pk_1 and pk_2, the public keys of $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively;
and l, the integer transforming value

Output:

$e_{1,k}$, the $\mathcal{P}_1$ ciphertext of kth provider's counts;
$e_{2,k}$, the $\mathcal{P}_2$ ciphertext of kth provider's counts;
and e_Y, the auxiliary information that contains encrypted noises

Initialize table $T_k = \langle \{n_i^{(k)}\}_{i=1}^m, \{\{\{n_{i,j}^{(k)}(v)\}_{v\in V_j}\}_{j=1}^d\}_{i=1}^m \langle$ and set each element 0;
for each $\langle \mathbf{x}, c\rangle \in D_k$ **do**
  Set i to the identifier of class c ($c = c_i$);
  $n_i^{(k)} \leftarrow n_i^{(k)} + 1$;
  for each $j \in [d]$ **do**
    $n_{i,j}^{(k)}(x_j) \leftarrow n_{i,j}^{(k)}(x_j) + 1$;
  end for
end for
for each $i \in [m]$ **do**
  Using pk_1 to encrypt $n_i^{(k)}$ to $e_i^{(1,k)} \leftarrow \tau_{1,k} \cdot [[n_i^{(k)} \cdot l]]_1$;
  Using pk_2 to encrypt $n_i^{(k)}$ to $e_i^{(2,k)} \leftarrow \tau_{2,k} \cdot [[n_i^{(k)}]]_2$;
  for each $j \in [d]$ **do**
    for each $v \in V_j$(the range of x_j) **do**
      Using pk_1 to encrypt $n_{i,j}^{(k)}(v)$ to $e_{i,j}^{(1,k)}(v) \leftarrow \tau_{1,k} \cdot [[n_{i,j}^{(k)}(v) \cdot l]]_1$;
    end for
  end for
end for
$e_{1,k} \leftarrow \langle \{e_i^{(1,k)}\}_{i=1}^m, \{\{\{e_{i,j}^{(1,k)}(v)\}_{v\in V_j}\}_{j=1}^d\}_{i=1}^m \rangle$;
$e_{2,k} \leftarrow \{e_i^{(2,k)}\}_{i=1}^m$;

$$
\begin{aligned}
&\tau_{1,0} \cdot e_i^{(1,1)} \cdot \ldots \cdot e_i^{(1,s)} \cdot e_i^{(Y)} \\
&= \tau_{1,0} \cdot \prod_{k=1}^{s} (\tau_{1,k} \cdot [[n_i^{(k)} \cdot l]]_1) \cdot (\tau_{1,s+1} \cdot [[y_i \cdot l]]_1) \\
&= \prod_{k=0}^{s+1} \tau_{1,k} \cdot \left[\left[y_i \cdot l + \sum_{k=1}^{s} (n_i^{(k)} \cdot l) \right]\right]_1 \\
&= [[y_i \cdot l + n_i \cdot l]]_1 \\
&= [[n_i' \cdot l]]_1 \mathbf{mod} N_1^2.
\end{aligned}
$$

Thus, DC can decrypt the ciphertexts to obtain each n_i' and $n_{i,j}'(v)$.

The other task is to carry out the encrypted classifier model $e_{\mathbf{W}} = \langle \{([[n]]_2)^{r_i}\}_{i=1}^m, \{\{\{([[n_i]]_2)^{r_{i,j}(v)}\}_{v\in V_j}\}_{j=1}^d\}_{i=1}^m \rangle$, where each r_i is the reciprocal of corresponding n_i and each $r_{i,j}(v)$ is the reciprocal of corresponding $n_{i,j}(v)$.

Similar as the aggregation on $\{e_{1,k}\}_{k=1}^s$, DC generates $\mathcal{P}_2$ ciphertexts for the fractions of class-conditional probabilities and prior probabilities.

- For each $[[n_i]]_2$, DC computes $\tau_{2,0} \cdot e_i^{(2,1)} \cdot \ldots \cdot e_i^{(2,s)} \mathbf{mod} N_2^2$;
- For each $[[n]]_2$, DC computes $[[n_1]]_2 \cdot \ldots \cdot [[n_m]]_2 \mathbf{mod} N_2^2$.

The correctness of $[[n_i]]_2$ and $[[n]]_2$ can be derived by computing

$$
\begin{aligned}
&\prod_{i=1}^{m} (\tau_{2,0} \cdot e_i^{(2,1)} \cdot \ldots \cdot e_i^{(2,s)}) \\
&= \prod_{i=1}^{m} \left(\tau_{2,0} \cdot \prod_{k=1}^{s} (\tau_{2,k} \cdot [[n_i^{(k)}]]_2) \right)
\end{aligned}
$$

Algorithm 2 Preparation. (Part 2).

```
//Performing Laplace mechanism if the provider holds τ_{1,s+1}.
if τ_{1,s+1} is NULL then
    e_Y ← ∅;
else
    for each i ∈ [m] do
        y_i ←R Lap(1/ε);
        Using pk_1 to encrypt y_i to e_i^(Y) ← τ_{1,1+s} · [[y_i · l]]_1;
        for each j ∈ [d] do
            for each v ∈ V_j do
                y_{i,j}(v) ←R Lap(1/ε);
                Using pk_1 to encrypt y_{i,j}(v) to e_{i,j}^(Y)(v) ← τ_{1,1+s} · [[y_{i,j}(v) · l]]_1;
            end for
        end for
    end for
    e_Y ← ⟨{e_i^(Y)}_{i=1}^m, {{{e_{i,j}^(Y)(v)}_{v∈V_j}}_{j=1}^d}_{i=1}^m⟩;
end if
return e_{1,k}, e_{2,k}, and e_Y.
```

$$
\begin{aligned}
&= \prod_{i=1}^{m}\left(\prod_{k=0}^{s}\tau_{2,k}\cdot[[\sum_{k=1}^{s}(n_i^{(k)})]]_2\right)\\
&= \prod_{i=1}^{m}[[n_i]]_2\\
&= [[\sum_{i=1}^{m}n_i]]_2\\
&= [[n]]_2 \mathbf{mod} N_2^2.
\end{aligned}
$$

Then, *DC* computes each reciprocal. To enable multiplications over $\mathbb{Z}_{N_2^2}$, reciprocals are in the integer form as $r_i = l \cdot 1/n'_i$ and $r_{i,j}(v) = l \cdot 1/n'_{i,j}(v)$.

Finally, *DC* builds and releases $e_{\mathbf{W}}$, of which underlying messages are probabilities, to the receiver *DR*. Algorithm 3 depicts the aggregation.

5.3.5. Extraction

In this phase, the aim of the receiver *DR* is to extract $\mathbf{W}$ of a differential private NB classifier. As the owner of secret key sk_2, *DR* can easily decrypt each element in $e_{\mathbf{W}}$. Finally, the model $\mathbf{W} = \{p'_i\}_{i=1}^m, \{\{\{p'_{i,j}(v)\}_{v\in V_j}\}_{j=1}^d\}_{i=1}^m$ is recovered from the result of decrypting $\langle\{e_i^{(2)}\}_{i=1}^m, \{\{\{e_{i,j}^{(2)}(v)\}_{v\in V_j}\}_{j=1}^d\}_{i=1}^m\rangle$. The correctness of *extraction* is obvious. Take p'_i as an instance, the underlying plaintext of $e_i^{(2)}$ is actually l/p'_i since

$$
\begin{aligned}
e_i^{(2)} &= ([[n]]_2)^{r_i}\\
&= [[n\cdot r_i]]_2 = [[n\cdot l\cdot 1/n'_i]]_2 = [[l/p'_i]]_2.
\end{aligned}
$$

5.4. Construction for vertically partitioned dataset

5.4.1. Main idea

In fact, if an entity has the jth attribute of each sample and the corresponding class label, the class-conditional probability $p_{i,j}(v)$ can be trained locally. Thus, in the vertical partition setting, the loss of label is the only one obstacle for a provider to perform computation. Without loss of generality, we simply set that d data providers jointly hold n samples and the j-the provider O_j has the jth attribute of these samples for convenience. In addition, the $d+1$th provider O_{d+1}, who is called the label provider *LP*, only has all the labels.

A part of the solution of training is still trivial. That is, *LP* is able to count the number of any class, what results each n_i, n, and $p_i = n_i/n$ are subsequently obtained by *LP* directly. *LP* uses Laplace mechanism to add the noise to each prior probability p'_i and contributes it to the receiver *DR* via a secure communication protocol.

Algorithm 3 Aggregation.

Input:

$\{e_{1,k}\}_{k=1}^{s}$ and $\{e_{2,k}\}_{k=1}^{s}$,$\mathcal{P}_1$ ciphertext and $\mathcal{P}_2$ ciphertext of providers' counts;
e_Y, the auxiliary information that contains encrypted noises;
pk_1 and pk_2, the public keys of $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively;
sk_1, the secret key of $\mathcal{P}_1$;
$\tau_{1,0}$ and $\tau_{2,0}$, the decryption factors of a $\mathcal{P}_1$ encrypted sum and $\mathcal{P}_2$ encrypted sum respectively;
and l, the integer transforming value

Output:

$e_{\mathbf{W}}$, the encrypted model of a NB classifier

1: //Obtain each noised counts.
2: **for** each $i \in [n]$ **do**
3: $[[n_i' \cdot l]]_1 \leftarrow \tau_{1,0} \cdot e_i^{(1,1)} \cdot \ldots \cdot e_i^{(1,s)} \cdot e_i^{(Y)}$;
4: Recover n_i' by decrypting $[[n_i' \cdot l]]_1$;
5: $r_i \leftarrow l \cdot 1/n_i'$;
6: **for** each $j \in [d]$ **do**
7: **for** each $v \in V_j$ **do**
8: $[[n_{i,j}'(v) \cdot l]]_1 \leftarrow \tau_{1,0} \cdot e_{i,j}^{(1,1)}(v) \cdot \ldots \cdot e_{i,j}^{(1,s)}(v) \cdot e_{i,j}^{(Y)}(v)$;
9: Recover $n_{i,j}'(v)$ by decrypting $[[n_{i,j}'(v) \cdot l]]_1$;
10: $r_{i,j}(v) \leftarrow l \cdot 1/n_{i,j}'(v)$;
11: **end for**
12: **end for**
13: **end for**
14: //Obtain encrypted model.
15: $[[n_i]]_2 \leftarrow \tau_{2,0} \cdot e_i^{(2,1)} \cdot \ldots \cdot e_i^{(2,s)}$;
16: $[[n]]_2 \leftarrow [[n_1]]_2 \cdot \ldots \cdot [[n_m]]_2$;
17: **for** each $i \in [n]$ **do**
18: $e_i^{(2)} \leftarrow ([[n]]_2)^{r_i}$;
19: **for** each $j \in [d]$ **do**
20: **for** each $v \in V_j$ **do**
21: $e_{i,j}^{(2)}(v) \leftarrow ([[n_i]]_2)^{r_{i,j}(v)}$;
22: **end for**
23: **end for**
24: **end for**
25: $e_{\mathbf{W}} \leftarrow \langle \{e_i^{(2)}\}_{i=1}^{m}, \{\{\{e_{i,j}^{(2)}(v)\}_{v \in V_j}\}_{j=1}^{d}\}_{i=1}^{m} \rangle$;
26: **return** $e_{\mathbf{W}}$.

Therefore, what should we consider is to make *DR* receive each differentially private class-conditional probability $p_{i,j}'(v)$ without breaking providers' privacy. Inspired by the personal information retrieve (PIR), we devise a way to enable O_j to count $n_{i,j}(v)$ and add the corresponding noise for each $v \in V_j$.

5.4.2. Initialization

For the system initialization, according to the secure parameter 1^{λ}, *system* initializes the Paillier cryptosystems $\mathcal{P}_1$ and $\mathcal{P}_2$, where the key pairs are $\langle pk_1, sk_1 \rangle$ and $\langle pk_2, sk_2 \rangle$ respectively. The public keys and parameters are published as public. The secret key sk_1 is issued to the collector *DC*, while sk_2 is issued to the receiver *DR*, The public factor l for transforming a float to an integer is determined in the initialization.

5.4.3. Preparation

The label provider and each data provider undertake different tasks in the preparation.

The label provider *LP* firstly counts the total number n and each class number n_i, and the set $\{n_i\}_{i=1}^{m}$ is encrypted to $\{[[n_i]]_2\}_{i=1}^{m}$. Then, *LP* creates m encrypted class vectors for each $i \in [m]$, where the ith vector $\mathbf{t}_i = (t_1, \ldots, t_n)$. For each $k \in [n]$, t_k is carried out as follows: if the kth sample is labelled with c_i, let t_k be $[[1]]_1$ (i.e., the constant 1 encrypted with P_1); else let t_k be $[[0]]_1$. Next, via a secure communication protocol, *LP* sends the class vectors to each O_j, and sends $\{[[n_i]]_2\}_{i=1}^{m}$ to the collector *DC*.

Alternatively, the task of each data provider O_j is to prepare a series of noised class-conditional counts without disclosing the class of samples. We denote the vectors in the dataset as $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and the jth attribute of $\mathbf{x}_k$ as $x_j^{(k)}$. After receiving the m encrypted class vectors, each O_j generate a cipher set for $\{\{n_{i,j}'(v)\}_{v \in V_j}\}_{i=1}^{m}$. The procedure for the jth provider O_j

Algorithm 4 Preparation of data provider.

Input:
$\{x_j^{(1)}\ldots, x_j^{(n)}\}$, the jth vertically partitioned dataset;
$\{\mathbf{t}_1, \ldots, \mathbf{t}_m\}$, a set of encrypted class vectors;
ϵ, the privacy parameter for differential privacy;
pk_1, the public key of $\mathcal{P}_1$;
and l, the integer transforming value

Output:
e_j, the $\mathcal{P}_1$ ciphertext of jth differentially private class-conditional counts

//Obtain encrypted class-conditional probabilities
for each $i \in [m]$ **do**
  for each $v \in V_j$ **do**
    $[[n_{i,j}(v)]]_1 \leftarrow [[0]]_1$;
    for each $k \in [n]$ **do**
      if $x_j^{(k)} = v$ **then**
        $[[n_{i,j}(v)]]_1 \leftarrow [[n_{i,j}(v)]]_1 \cdot (t_k^{(i)})^l$;
      end if
    end for
  end for
end for
//Obtain encrypted noised class-conditional probabilities
for each $i \in [m]$ **do**
  for each $v \in V_j$ **do**
    $y_{i,j}(v) \overset{R}{\leftarrow} Lap(1/\epsilon)$;
    Using pk_1 to encrypt $y_{i,j}(v)$ to $[[y_{i,j}(v) \cdot l]]_1$;
    $[[n'_{i,j}(v)]]_1 \leftarrow [[n_{i,j}(v)]]_1 \cdot [[y_{i,j}(v) \cdot l]]_1$;
  end for
end for
$e_j \leftarrow \{\{[[l \cdot n'_{i,j}(v)]]_1\}_{v\in V_j}\}_{i=1}^m$;
return e_j.

is depicted Algorithm 4. The correctness of this algorithm can be easily derived. Then, the encrypted counts $e_j = \{\{[[l \cdot n'_{i,j}(v)]]_1\}_{v\in V_j}\}_{i=1}^m$ are submitted to *DC*.

5.4.4. Aggregation and extraction

Aggregation. After receiving $\{[[n_i]]_2\}_{i=1}^m$ and $\{e_j\}_{j=1}^d$, the receiver *DC* starts to aggregate differentially private class-conditional probabilities $\{\{\{[[l \cdot p'_{i,j}(v)]]_2\}_{v\in V_j}\}_{j=1}^d\}_{i=1}^m$ over the ciphertext. Our trick is similar as the one in the horizontally partitioned setting. To the begin with, *DC* uses sk_1 to decrypt $\{e_j\}_{j=1}^d$ to a clear set, and obtains $\{\{\{n'_{i,j}(v)\}_{v\in V_j}\}_{j=1}^d\}_{i=1}^m$. Then, *DC* computes the reciprocal of each $n'_{i,j}(v)$ in float. To make sure that multiplications are over $\mathbb{Z}_{N_2^2}$, we set the integer reciprocals as $r_{i,j}(v) = l \cdot 1/n'_{i,j}(v)$. Finally, *DC* releases $\{\{\{([[n_i]]_2)^{r_{i,j}(v)}\}_{v\in V_j}\}_{j=1}^d\}_{i=1}^m$, of which the underlying messages are class-conditional probabilities, to the receiver *DR*.

Extraction. In this phase, the aim of the receiver *DR* is to extract class-conditional probabilities in a differential private NB classifier, since prior probabilities have been contributed by the label provider *LP*. As the owner of secret key sk_2, *DR* can easily obtain the probabilities by decrypting each element in $\{\{\{([[n_i]]_2)^{r_{i,j}(v)}\}_{v\in V_j}\}_{j=1}^d\}_{i=1}^m$ and then recovering it to a float. Thus, $\mathbf{W} = \{p'_i\}_{i=1}^m, \{\{\{p'_{i,j}(v)\}_{v\in V_j}\}_{j=1}^d\}_{i=1}^m$ is built. The correctness deriving is same as in Section 5.3.5.

6. Security analysis

In this section, we make intuitively security analysis based on assumptions that contain the security property of ϵ-differential privacy, the secure communication protocol, and the semantic security of the Paillier cryptosystem.

An external adversary can only observe the communication among the provider *O*, the collector *DC*, and the receiver *DR* during the training. Therefore, the internal adversaries have more attack power than the external adversaries and thus we only need to consider privacy and security against the internal attackers.

In Section 4.2, we present three types of internal adversaries that do not collude with each other. According to the security requirements in Section 4.3, the *statistics privacy* should be protected against the type-i adversary $\mathcal{A}_1$, type-ii adversary $\mathcal{A}_2$, and type-iii adversary $\mathcal{A}_3$, who try to reveal n, each n_i, and each $n_{i,j}(v)$.

6.1. Statistics privacy

6.1.1. Construction for horizontally partitioned dataset

In the constructions for horizontally partitioned dataset, other than the eavesdropped information, what $\mathcal{A}_1$ (i.e., a provider O_k) learns are its own partitioned dataset D_k, private factors $\langle \tau_{1,k}, \tau_{2,k} \rangle$, and the public information. We assume that $\mathcal{A}_1$ undertakes encrypting Laplace noises and it will also hold $\tau_{1,s+1}$. In the view of $\mathcal{A}_1$, the outputs of Algorithms 1 and 3 can be captured, which results that $\mathcal{A}_1$ has $\{e_{1,1}, \ldots, e_{1,s}\}$, $\{e_{2,1}, \ldots, e_{2,s}\}$, and $e_{\mathbf{W}}$. On one hand, due to the semantic security of $\mathcal{P}_2$, $\mathcal{A}_1$ cannot recover any p'_i or $p'_{i,j}(v)$ from $e_{\mathbf{W}}$. On the other hand, although $\mathcal{A}_1$ can locally aggregate $e_i^{(1,1)} \cdot \ldots \cdot e_i^{(1,s)} \cdot \tau_{1,s+1}$ to $[[n_i \cdot l]]_1 \cdot \tau_{1,0}^{-1}$, it still cannot extract n_i without knowing $\tau_{1,0}$ and the decryption key sk_1 (the same as $n_{i,j}(v)$). Therefore, the statistics privacy of the dataset can be protected against $\mathcal{A}_1$.

The adversary $\mathcal{A}_2$ plays a role of *DC*, which views $\{e_{1,k}\}_{k=1}^{s}$, $\{e_{2,k}\}_{k=1}^{s}$, and e_Y. Since the factors $\langle \tau_{1,1}, \ldots, \tau_{1,s} \rangle$ and $\langle \tau_{2,1}, \ldots, \tau_{2,s} \rangle$ are chosen randomly in $\mathbb{Z}_{N_1^2}$ and $\mathbb{Z}_{N_2^2}$ respectively, the elements in $e_{1,k}$ and $e_{2,k}$ are not legal Paillier ciphertexts. As a result, without knowing the factors, $\mathcal{A}_2$ can only get the aggregation results rather than the underlying contents of any $e_{1,k}$ or $e_{2,k}$. However, the clear counts $\{n'_i\}$ and $\{n'_{i,j}(v)\}$ are ϵ-differentially private, while the cipher counts $[[n_i]]_2$ and $[[n_{i,j}(v)]]_2$ are encoded via $\mathcal{P}_2$ encryption which cannot be decrypted without sk_2. Thus, the statistics privacy is also protected against $\mathcal{A}_2$.

Similar as $\mathcal{A}_2$, even the receiver $\mathcal{A}_3$ can capture $\{e_{1,k}\}_{k=1}^{s}$, $\{e_{2,k}\}_{k=1}^{s}$, and e_Y, it still cannot disclose the underlying content of any individual due to missing the random factors. Alternatively, the training result, which is the decryption of $e_{\mathbf{W}}$, achieves ϵ-differential privacy and does not contain the information of statistics counts. Hence, the privacy can be protected against $\mathcal{A}_3$.

6.1.2. Construction for vertically partitioned dataset

In this construction, consider the format of partitioned sets the total number of samples in the whole dataset (i.e., n) is "public" for each provider O_j. Besides, the label holder *LP* has already known each count n_i from its own class label set, and an honest-but-curious *LP* can be regarded as a special adversary $\mathcal{A}'_1$.

Considering the label holder *LP* sends the class vectors to each O_j, and sends $\{[[n_i]]_2\}_{i=1}^{m}$ to *DC*. via a secure communication protocol, such sent ciphertexts cannot be captured by entities other than their receiver. As a result, $\mathcal{A}_2$ can only obtain ϵ-differentially private counts $\{\{n'_{i,j}(v)\}_{v \in V_j}\}_{i=1}^{m}$ by decrypting $e_j = \{\{[[l \cdot n'_{i,j}(v)]]_1\}_{v \in V_j}\}_{i=1}^{m}$, while $\mathcal{A}_3$ only obtains the trained NB classifier with ϵ-differentially private probabilities.

For $\mathcal{A}_1$ (i.e, the provider O_j), $[[0]]_1$ and $[[1]]_1$ are indistinguishable due to the semantic security of $\mathcal{P}_1$. That is, $\mathcal{A}_1$ cannot determine the underlying text of any component of a class vector $\mathbf{t}_i = (t_1, \ldots, t_n)$, which makes it have no way to infer $\{n_i\}_{i=1}^{m}$ and $\{\{n_{i,j}(v)\}_{v \in V_j}\}_{i=1}^{m}$. Alternatively, since $\mathcal{A}'_1$ can capture $\{e_j\}_{j=1}^{d}$ and $\{\{\{([[n_i]]_2)^{r_{i,j}(v)}\}_{v \in V_j}\}_{j=1}^{d}\}_{i=1}^{m}$ but does not hold corresponding secret keys, it cannot reveal $\{\{n_{i,j}(v)\}_{v \in V_j}\}_{i=1}^{m}$.

Therefore, the statistics privacy of the dataset can be protected in this construction.

6.2. Ownership privacy

In the constructions for vertically partitioned dataset, each provider O_j has the jth attribute values of all samples in the whole dataset. That means, if the statistics privacy is achieved in this construction, the ownership of samples that contain a specific attribute value is hidden for $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$. Hence, the ownership privacy is equivalent to the statistics privacy in this setting.

In the other setting, the ownership privacy is based on the semantic security of the Paillier cryptosystem. Take $\mathcal{A}_1$ (is not the provider O_k) as an example. In its view, $n_i^{(k)} = 0$ means the kth provider O_k does not hold a sample of which class label is c_i, and $n_{i,j}^{(k)}(v) = 0$ means the kth provider O_k does not hold a sample of which jth attribute is a specific value v. $\mathcal{A}_1$ try to learn them, but the captured $e_{1,k}$ and $e_{2,k}$ do not belong to $\mathcal{A}_1$. As a result, $\mathcal{A}_1$ cannot distinguish whether $e_i^{(1,k)}$ ($e_i^{(2,k)}$) in $e_{1,k}$ ($e_{2,k}$, respectively) is the encryption of $n_i^{(k)} = 0$ or not. Therefore, this adversary cannot learn whether another provider holds a sample contains a specific attribute value and class label.

On the similar condition, $\mathcal{A}_2$ and $\mathcal{A}_3$ do not have another way to conduct distinguishing, since each $e_{1,k}$ and $e_{2,k}$ cannot be decrypted directly.

Therefore, the ownership privacy can be protected here.

7. Implementation and evaluation

7.1. Implementation details

Our prototype is implemented in C++ Language on two different machine in the LAN, and they are connected with 1 Gbps Ethernet network. The one acts as the user U that is equipped with an Inter(R) Core(TM) i7-2600 3.40 GHz CPU,

Table 1
Parameters of secure Naive Bayes classifiers.

Dataset	Attribute number	Class number	Set size
Balance Scale	4	3	625
Breast Cancer Original	9	2	699
SPECT Heart	22	2	307
Gene Sequences	60	3	3190

4 GB RAM and installed with Ubuntu 16.04 64-Bit Version. The other acts as the remote server *RS* that is equipped with an Intel(R) Xeon(R) E5-2630 v3 2.40 GHz CPU, 16 GB RAM and installed with Ubuntu Server 16.04 64-Bit Version.

In the implementation, the training and testing datasets are chosen from UCI Machine Learning Repository. We choose *Balance Scale, Breast Cancer Wisconsin (Original), SPECT Heart*, and *Splice-Junction Gene Sequences* for training and testing Naive Bayes classifiers. The parameters are shown in Table 1. The classifiers are trained using *scikit-learn* non-privately.

We adopt *GMP* library to implement cryptographic operations. Considering security and efficiency, the modulus N of the Paillier cryptosystem is set as 1024-bit. Real numbers are represented by IEEE 754 double precision floating point numbers with 52 bits of precision, and we choose the big integer $l = 2^{80}$ for our number conversion.

7.2. Evaluation

For the dataset partitioned by different means, we perform experiments in terms of the computational cost of *Preparation* phase, *Aggregation* phase, and *Extraction* phase, respectively. The existed works are proposed either just for the single data owner or without the protection of the statistics privacy. Thus, these works cannot be used for solving the problem we focus on and compare to ours directly. In the experiment, we set $\epsilon = 0.1$ for the differential privacy.

7.2.1. Horizontally partitioned dataset

.

The whole dataset is partitioned (as evenly as possible) into several part each of which is held by a provider. The number of parts varies from 2 to 16. Before submitting his/her private data to the collector *DC* for NB learning, each provider $O_k (k \in [s])$ runs Algorithm 1. Assume that the first provider O_1 has the duty to perform Laplace mechanism. Fig. 2a and b show that the data encryption time for each provider is not affected by the number of samples in the partitioned set (the number of providers as well), while the time of adding noises does not vary either. This is because each provider has to encrypt all partitioned prior counts and class-conditional counts to make them obvious no matter how many samples he/she holds. The preparation has an one-time cost and can be pre-computed off-line.

After collecting all encryptions from providers, the performance of *DC* in the aggregation is depicted in Fig. 2c. In the algorithm, for each $e_{1,k}$ and $e_{2,k}$, *DC* aggregates it with the corresponding blinding factor. Although the time spent on this operation is determined by the number of providers theoretically, its trend is not apparent when the number is small. Besides undertaking aggregation tasks, *DC* should also compute each noised prior count and class-condition count, and then derive the corresponding encrypted probability. In Fig. 2(c), the time cost is roughly linear with the sum of each $|S_j|$ (the size of attribute set S_j), and thus we can learn that the Paillier decryption and the multi-plus operation over Paillier ciphertexts cause most of the cost. The performance of the receiver *DR* in the extraction is depicted in Fig. 2(d). Since the aim of *DR* is to decrypt $e_{\mathbf{W}}$, its time cost has the similar result as *DC*'s.

7.2.2. Vertically partitioned dataset

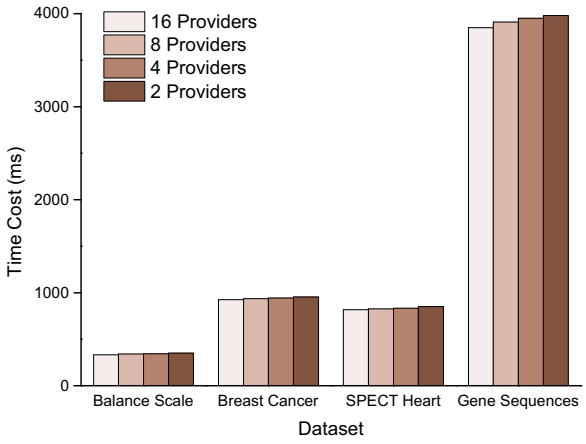
.

Recall that the partition of attribute is trivial in the vertical partition setting. Thus, we simply set that there is only one data provider O (not the label provider *LP*) in the system. Considering the computation of differentially private prior probabilities is trivial for *LP*, our evaluation is only conduct for class-conditional probabilities.

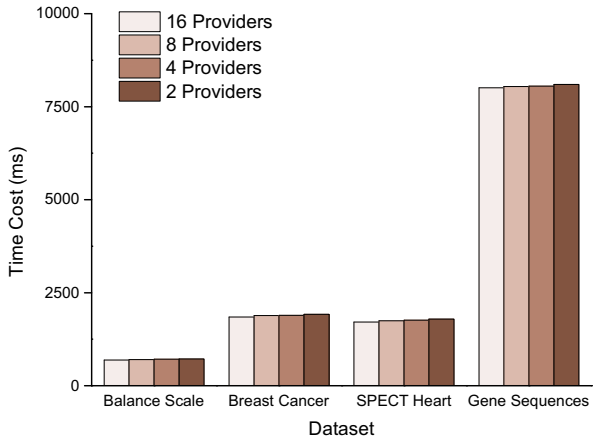
Fig. 3(a) describes the performance of O and *LP*. For the generation of class vectors, *LP* has to go through the whole dataset and modify $\mathcal{P}_1$ ciphertext for each sample. Alternatively, O uses these vectors to obviously compute each class-conditional count and then performs the private mechanism. As a result, the time cost is affected by the size of dataset, the number of classes, and the sum of each $|S_j|$.

Similar as in the horizontal partition setting, the necessary work of *DC* is to decrypt $\mathcal{P}_1$ ciphertexts of noised counts and create $\mathcal{P}_2$ ciphertexts of noised probabilities, while the work of *DR* is to decrypt $e_{\mathbf{W}}$. From Fig. 3(b), we can see the time spent on the aggregation of *DC* and the extraction of *DR*. The cost is roughly determined by both the number of classes and the sum of each $|S_j|$.

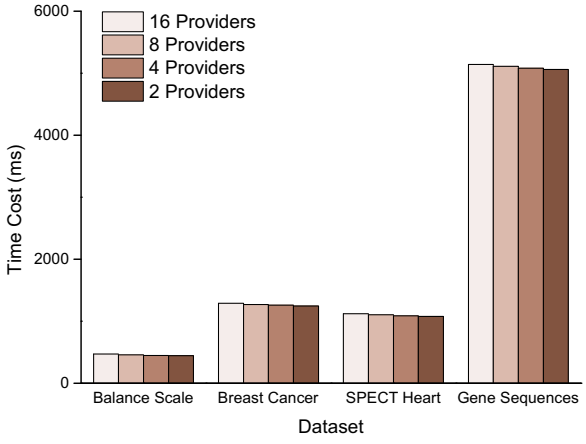
The experimental results in the two setting show that the algorithms run in at most a few seconds in once training. Therefore, we believe the proposed scheme to be practical for privacy-preserving applications.



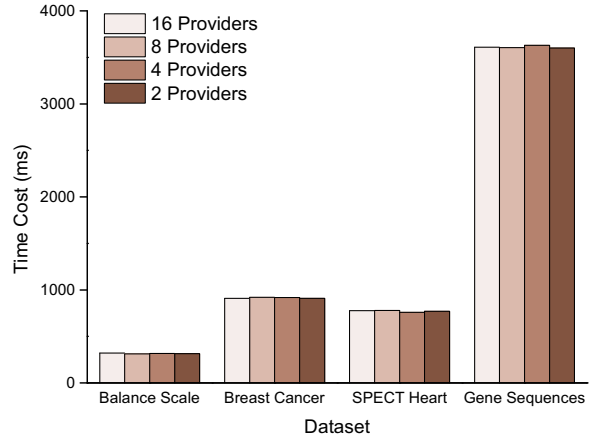
(a) Preparation of O_k



(b) Preparation of O_1

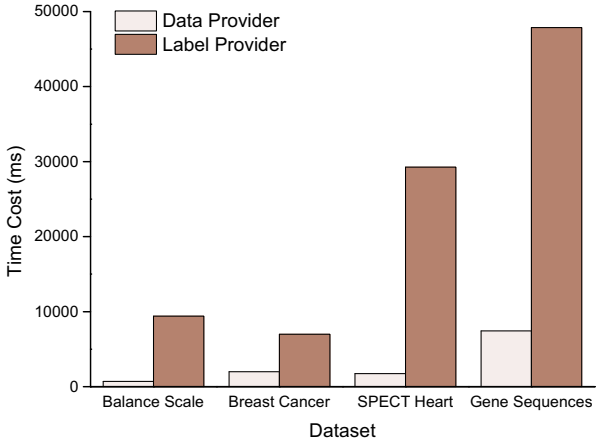


(c) Aggregation of DC

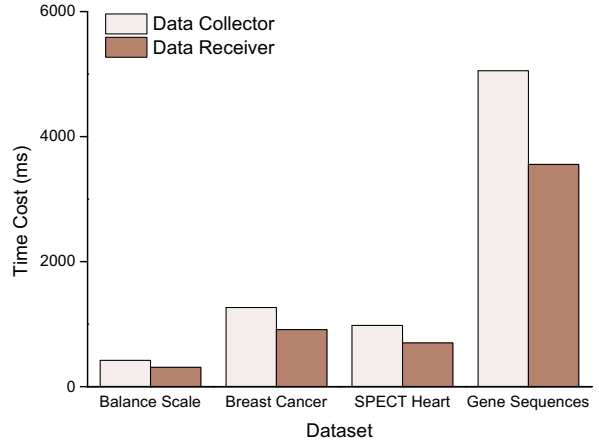


(d) Extraction of DR

Fig. 2. Computational time cost.



(a) Preparation



(b) Aggregation and Extraction

Fig. 3. Computational time cost.

8. Conclusion

In this paper, we propose a scheme for privacy-preserving Naive Bayes learning over data contributed by multiple providers. In the proposed scheme, the designed algorithms can aggregating each provider's data and implementing the private mechanism on training results privately. Thus, a data receiver is allow train a NB classifier which achieves ϵ-differential privacy without breaking the privacy of each provider. Moreover, the proposed scheme can also guarantee both statistics privacy and ownership privacy, and is applicable for either horizontally or vertically partitioned dataset. Finally, our experimental results over datasets from UCI Machine Learning Repository show that the proposed scheme is practical for applications. In the future, we plan to enhance the proposed scheme based on a weaker assumption where some collusions are allowed or adversaries have abilities of forgery and tampering.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61772291 and 61472091), the Natural Science Foundation of Tianjin, China (No. 17JCZDJC30500), the Open Project Foundation of Information Security Evaluation Center of Civil Aviation, Civil Aviation University of China (NO. CAAC-ISECCA-201702), and National Natural Science Foundation for Outstanding Youth Foundation (No. 61722203).

References

[1] M. Abadi, A. Chu, I. Goodfellow, H.B. McMahan, I. Mironov, K. Talwar, L. Zhang, Deep learning with differential privacy, in: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, ACM, 2016, pp. 308–318.
[2] R. Agrawal, R. Srikant, Privacy-preserving data mining, in: Proceedings of the ACM Sigmod Record, 29, ACM, 2000, pp. 439–450.
[3] A. Bansal, T. Chen, S. Zhong, Privacy preserving back-propagation neural network learning over arbitrarily partitioned data, Neural Comput. Appl. 20 (1) (2011) 143–150.
[4] M. Barni, P. Failla, V. Kolesnikov, R. Lazzeretti, A.-R. Sadeghi, T. Schneider, Secure evaluation of private linear branching programs with medical applications, in: Proceedings of the European Symposium on Research in Computer Security, Springer, 2009, pp. 424–439.
[5] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
[6] R. Bost, R.A. Popa, S. Tu, S. Goldwasser, Machine learning classification over encrypted data., IACR Cryptol. ePrint Arch. 2014 (2014) 331.
[7] Z. Brakerski, V. Vaikuntanathan, Efficient fully homomorphic encryption from (standard) LWE, SIAM J. Comput. 43 (2) (2014) 831–871.
[8] K. Chen, L. Liu, Privacy preserving data classification with rotation perturbation, in: Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM), IEEE, 2005, pp. 4–pp.
[9] T. Chen, S. Zhong, Privacy-preserving backpropagation neural network learning, IEEE Trans. Neural Netw. 20 (10) (2009) 1554–1564.
[10] X. Chen, J. Li, J. Ma, Q. Tang, W. Lou, New algorithms for secure outsourcing of modular exponentiations, IEEE Trans. Parallel Distrib. Syst. 25 (9) (2014) 2386–2396.
[11] X. Chen, J. Li, J. Weng, J. Ma, W. Lou, Verifiable computation over large database with incremental updates, IEEE Trans. Comput. 65 (10) (2016) 3184–3195.
[12] C. Dwork, Differential privacy: a survey of results, in: Proceedings of the International Conference on Theory and Applications of Models of Computation, Springer, 2008, pp. 1–19.
[13] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, M. Naor, Our data, ourselves: privacy via distributed noise generation., in: Proceedings of the Eurocrypt, 4004, Springer, 2006, pp. 486–503.
[14] C. Dwork, A. Roth, et al., The algorithmic foundations of differential privacy, Found. Trends® Theor. Comput. Sci. 9 (3–4) (2014) 211–407.
[15] A. Frank, A. Asuncion, et al., Uci machine learning repository, 2010, https://archive.ics.uci.edu/ml/index.php.
[16] C.-z. Gao, Q. Cheng, X. Li, S. Xia, Cloud-assisted privacy- preserving profile-matching scheme under multiple keys in mobile social network, Cluster Comput. (2018), doi:10.1007/s10586-017-1649-y.
[17] C. Gentry, A fully homomorphic encryption scheme, Ph.D. thesis, Stanford University, 2009.
[18] C. Gentry, S. Halevi, Implementing Gentrys fully-homomorphic encryption scheme, in: Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2011, pp. 129–148.
[19] T. Graepel, K. Lauter, M. Naehrig, Ml confidential: Machine learning on encrypted data, in: Proceedings of the International Conference on Information Security and Cryptology, Springer, 2012, pp. 1–21.
[20] Y. Hong, J. Vaidya, H. Lu, M. Wu, Differentially private search log sanitization with optimal output utility, in: Proceedings of the 15th International Conference on Extending Database Technology, ACM, 2012, pp. 50–61.
[21] M. Huai, L. Huang, W. Yang, L. Li, M. Qi, Privacy-preserving Naive Bayes classification, in: Proceedings of the International Conference on Knowledge Science, Engineering and Management, Springer, 2015, pp. 627–638.
[22] Y. Huang, D. Evans, J. Katz, L. Malka, Faster secure two-party computation using garbled circuits., in: Proceedings of the USENIX Security Symposium, 201, 2011.
[23] Z. Huang, S. Liu, X. Mao, K. Chen, J. Li, Insight of the protection for data security under selective opening attacks, Inf. Sci. 412–413 (2017) 223–241.
[24] G. Jagannathan, R.N. Wright, Privacy-preserving distributed k-means clustering over arbitrarily partitioned data, in: Proceedings of the Eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, ACM, 2005, pp. 593–599.
[25] M. Kantarcıoglu, J. Vaidya, C. Clifton, Privacy preserving naive bayes classifier for horizontally partitioned data, in: Proceedings of the IEEE ICDM Workshop on Privacy Preserving Data Mining, 2003, pp. 3–9.
[26] A. Korolova, K. Kenthapadi, N. Mishra, A. Ntoulas, Releasing search queries and clicks privately, in: Proceedings of the 18th International Conference on World Wide Web, ACM, 2009, pp. 171–180.
[27] S. Laur, H. Lipmaa, T. Mielikäinen, Cryptographically private support vector machines, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2006, pp. 618–624.
[28] B. Li, Y. Huang, Z. Liu, J. Li, Z. Tian, S.-M. Yiu, Hybridoram: practical oblivious cloud storage with constant bandwidth, Inf. Sci. (2018), doi:10.1016/j.ins.2018.02.019.
[29] J. Li, X. Chen, M. Li, J. Li, P.P. Lee, W. Lou, Secure deduplication with efficient and reliable convergent key management, IEEE Trans. Parallel Distrib. Syst. 25 (6) (2014) 1615–1625.
[30] J. Li, J. Li, X. Chen, C. Jia, W. Lou, Identity-based encryption with outsourced revocation in cloud computing, IEEE Trans. Comput. 64 (2) (2015) 425–437.
[31] J. Li, Y.K. Li, X. Chen, P.P. Lee, W. Lou, A hybrid cloud approach for secure authorized deduplication, IEEE Trans. Parallel Distrib. Syst. 26 (5) (2015) 1206–1216.
[32] J. Li, Z. Liu, X. Chen, F. Xhafa, X. Tan, D.S. Wong, L-Encdb: a lightweight framework for privacy-preserving data queries in cloud computing, Knowl. Based Syst. 79 (2015) 18–26.

[33] J. Li, L. Sun, Q. Yan, W. Srisa-an, H. Ye, Significant permission identification for machine learning based android malware detection, IEEE Trans. Ind. Inf. (2017), doi:10.1109/TII.2017.2789219.
[34] J. Li, Y. Zhang, X. Chen, Y. Xiang, Secure attribute-based data sharing for resource-limited users in cloud computing, Comput. Secur. 72 (2018) 1–12.
[35] P. Li, J. Li, Z. Huang, C.-Z. Gao, W.-B. Chen, K. Chen, Privacy-preserving outsourced classification in cloud computing, Cluster Comput (2017), doi:10.1007/s10586-017-0849-9.
[36] P. Li, J. Li, Z. Huang, T. Li, C.-Z. Gao, S.-M. Yiu, K. Chen, Multi-key privacy-preserving deep learning in cloud computing, Future Gener. Comput. Syst. 74 (2017) 76–85.
[37] T. Li, Z. Liu, J. Li, C. Jia, K.-C. Li, CDPS: a cryptographic data publishing system, J. Comput. Syst. Sci. 89 (2016) 80–91.
[38] Y. Lindell, B. Pinkas, Privacy preserving data mining, in: Proceedings of the Annual International Cryptology Conference, Springer, 2000, pp. 36–54.
[39] F. McSherry, I. Mironov, Differentially private recommender systems: building privacy into the net, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2009, pp. 627–636.
[40] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, M. Costa, Oblivious multi-party machine learning on trusted processors, in: Proceedings of the USENIX Security, 16, 2016, pp. 619–636.
[41] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 1999, pp. 223–238.
[42] S. Samet, A. Miri, Privacy-preserving back-propagation and extreme learning machine algorithms, Data Knowl. Eng. 79 (2012) 40–61.
[43] N. Schlitter, A protocol for privacy preserving neural network learning on horizontally partitioned data, in: Proceedings of the PSD, 2008.
[44] J. Shen, Z. Gui, S. Ji, J. Shen, H. Tan, Y. Tang, Cloud-aided lightweight certificateless authentication protocol with anonymity for wireless body area networks, J. Netw. Comput. Appl. (2018), doi:10.1016/j.jnca.2018.01.003.
[45] R. Shokri, V. Shmatikov, Privacy-preserving deep learning, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, ACM, 2015, pp. 1310–1321.
[46] J. Vaidya, C. Clifton, Privacy preserving naive Bayes classifier for vertically partitioned data, in: Proceedings of the 2004 SIAM International Conference on Data Mining, SIAM, 2004, pp. 522–526.
[47] J. Vaidya, M. Kantarcıoğlu, C. Clifton, Privacy-preserving naive Bayes classification, Int. J. Very Large Data Bases 17 (4) (2008) 879–898.
[48] J. Vaidya, B. Shafiq, A. Basu, Y. Hong, Differentially private naive bayes classification, in: Proceedings of the IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), vol. 01, IEEE Computer Society, 2013, pp. 571–576.
[49] H. Wang, Z. Zheng, L. Wu, P. Li, New directly revocable attribute-based encryption scheme and its application in cloud storage environment, Cluster Comput. 20 (3) (2017) 2385–2392.
[50] P. Xie, M. Bilenko, T. Finley, R. Gilad-Bachrach, K. Lauter, M. Naehrig, Crypto-nets: neural networks over encrypted data, 2014, https://arxiv.org/pdf/1412.6181.pdf.
[51] J. Xu, L. Wei, Y. Zhang, A. Wang, F. Zhou, C.-z. Gao, Dynamic fully homomorphic encryption-based Merkle tree for lightweight streaming authenticated data structures, J. Netw. Comput. Appl. (2018), doi:10.1016/j.jnca.2018.01.014.
[52] A.C. Yao, Protocols for secure computations, in: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (SFCS), IEEE, 1982, pp. 160–164.
[53] X. Yi, Y. Zhang, Privacy-preserving Naive Bayes classification on distributed data via semi-trusted mixers, Inf. Syst. 34 (3) (2009) 371–380.
[54] J. Yuan, S. Yu, Privacy preserving back-propagation neural network learning made practical with cloud computing, IEEE Trans. Parallel Distrib. Syst. 25 (1) (2014) 212–221.