Future Generation Computer Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Privacy-preserving machine learning with multiple data providers

Ping Li [a], Tong Li [b], Heng Ye [c], Jin Li [a,*], Xiaofeng Chen [d], Yang Xiang [e]

[a] *School of Computer Science, Guangzhou University, Guangzhou 510006, China*
[b] *College of Computer & Control Engineering, Nankai University, 300071, Tianjin, China*
[c] *Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, 3 Shangyuancun, Beijing 100044, China*
[d] *State Key Laboratory of Integrated Service Networks, Xidian University, 710126, Xi'an, China*
[e] *School of Information Technology, Deakin University, Melbourne Burwood, VIC 3125, Australia*

HIGHLIGHTS

- To protect data privacy, multiple parties encrypt their data under their own public key of double decryption algorithm, before outsourcing it to cloud for storing and processing.
- To improve the efficiency and accuracy of the computation, cloud transforms the encrypted data into noised data, such that the machine learning algorithm can be executed on this noised data with ϵ-differential privacy.
- The proposed scheme is proven to be secure in the security model.

ARTICLE INFO

Article history:
Received 20 November 2017
Received in revised form 30 March 2018
Accepted 23 April 2018
Available online xxxx

Keywords:
Differential privacy
Homomorphic encryption
Outsourcing computation
Machine learning

ABSTRACT

With the fast development of cloud computing, more and more data storage and computation are moved from the local to the cloud, especially the applications of machine learning and data analytics. However, the cloud servers are run by a third party and cannot be fully trusted by users. As a result, how to perform privacy-preserving machine learning over cloud data from different data providers becomes a challenge. Therefore, in this paper, we propose a novel scheme that protects the data sets of different providers and the data sets of cloud. To protect the privacy requirement of different providers, we use public-key encryption with a double decryption algorithm (DD-PKE) to encrypt their data sets with different public keys. To protect the privacy of data sets on the cloud, we use ϵ-differential privacy. Furthermore, the noises for the ϵ-differential privacy are added by the cloud server, instead of data providers, for different data analytics. Our scheme is proven to be secure in the security model. The experiments also demonstrate the efficiency of our protocol with different classical machine learning algorithms.

1. Introduction

With the fast development of cloud computing, more and more data and applications are moved from the local to cloud servers, including machine learning and other data analytics. However, the cloud computing platform cannot be fully trusted because it is run by a third party. Cloud users lose the control of their data after outsourcing their data to the cloud. To protect the privacy, the data are usually encrypted before they are uploaded to the cloud storage. However, the encryption techniques render the data utilization difficult.

Though there are some traditional techniques such as homomorphic cryptographic techniques to provide solutions for the data utilization over encrypted data, they are inefficient in practice. To address this challenge, another important notion of differential privacy has been proposed. It can not only protect the privacy, but also provides efficient data operations.

However, most of the previous mainly focus on the data from a single user. It is common that the data always from different data providers for machine learning. Therefore, how to perform machine learning over cloud data from multiple users become a new challenge. Traditional differential privacy technique and encryption methods are not practical for this environment. On one hand, the data from different users are encrypted with different public keys or noises, which makes the computation be difficult. On the other hand, data have to be proceeded in different ways for different applications, which makes both the communication overhead and computation overhead be huge.

* Corresponding author.
E-mail addresses: liping26@mail2.sysu.edu.cn (P. Li), litongziyi@mail.nankai.edu.cn (T. Li), heng.ye@bjtu.edu.cn (H. Ye), jinli71@gmail.com (J. Li), xfchen@xidian.edu.cn (X. Chen), yang@deakin.edu.au (Y. Xiang).

https://doi.org/10.1016/j.future.2018.04.076
0167-739X/© 2018 Elsevier B.V. All rights reserved.

Main idea. To tackle the above challenges, we propose a scheme named privacy-preserving machine learning under multiple keys (PMLM) to solve this problem. Since the secure multi-party computation (SMC) only supports the computation on the data encrypted under the *same public key* and the efficiency and accuracy of the computation need to be improved. Therefore, our PMLM scheme as an efficient solution is required that conducts the data encrypted under *different public keys* for different data providers and improves the efficiency and accuracy. Our novel technique based on a new public-key encryption with a double decryption algorithm (DD-PKE) and differential privacy. The DD-PKE is additively homomorphic scheme and holds two independent decryption algorithms which allows the outsourced data set to be transformed into randomized data. The differential privacy can be used to add statistical noises to the outsourced data set for data analyses and data computations.

Our PMLM scheme works as follows. First, we set up a public-key encryption with a double decryption algorithm (DD-PKE) to protect the data privacy of multiple data providers. During this phase, we do not take the differential privacy protection into consideration. We then use a cloud server to add different statistical noises to outsourced ciphertexts according to the different applications of the data analyst, and these noises are encrypted under a public key corresponding to the outsourced ciphertexts. Finally, the data analyst downloads this noise-added ciphertext data sets, decrypts it using his or her own master key and performs a machine learning task over this joint distribution with minimum error.

Our Contributions. In our PMLM scheme, we assume that the cloud server and data analyst are not collude with each other and that they are *semi-honest*. In all steps of PMLM scheme, the multiple users do not interact with each other. We show that our PMLM scheme is IND-CCA secure in the random oracle model.

In particular, the main contributions of this work are summarized as follows:

- In this work, the cloud server has the authority to add different statistical noises to the outsourced data set according to different queries of the data analyst rather than the data providers adding statistical noise by themselves with only one application.
- We use a DD-PKE cryptosystem to preserve the privacy of the data providers' data sets, which can be used to transform the encrypted data into a randomized data set without information leakage.
- In our PMLM scheme, the machine learning task is performed on a randomized data set with ϵ-differential privacy rather than on the encrypted data set. This process improves the computational efficiency and data analysis accuracy.

Organization of the Paper. The remainder of this paper is organized as follows. Section 2 provides a literature review over privacy-preserving machine learning based on differential privacy protection. Section 3 presents some notations and definitions on cryptographic primitives and differential privacy. In Section 4, we present the system model, the problem statement and the adversary model. In Section 5, we provide the PMLM scheme. Then, we present our simulation results in Section 6 and the security analysis in Section 7. Finally, the conclusions and directions for future work are presented in Section 8.

2. Related work

Machine learning is the process of programming computers to optimize a performance criterion using example data or prior experience. Because of its powerful ability to process large amounts of data, machine learning has been applied in various fields in recent years, including speaker recognition [1], image recognition [2,3] and signal processing [4]. To protect the data privacy in the machine learning model, two well-known lines of research should be considered in our work.

2.1. Homomorphic encryption in machine learning

There are many works considering the problem of privacy preserving for outsourced computation. Homomorphic encryption is one of the basic techniques, which can be also applied in machine learning. To protect the privacy of users' sensitive data, users only provide the encrypted data for data storing and data processing. For instance, Chen et al. [5] presented a privacy-preserving two-party distributed algorithm of back-propagation neural networks (BPNN) which allows a neural network to be trained without revealing the information about each of party. To preserve the privacy of input data and output result, they used a homomorphic scheme to keep the security. In their work, the BPNN conducts the vertically partitioned data, i.e., each party has a subset of feature vector. Due to their scheme only process vertically partitioned data, in the subsequent work, Bansal et al. [6] proposed a similar scheme for privacy-preserving BPNN over arbitrarily partitioned data between two parties. However, all works [5,6] cannot be applied to the multi-party scenario because directly extending them to the multi-party scenario will lead to the communication overhead.

Hence, Samet et al. [7] presented new privacy-preserving protocols for both the BPNN and extreme learning machine (ELM) algorithms with horizontally and vertically partitioned data among multiple parties. Graepel et al. [8] proposed secure machine learning scheme over encrypted data, they only trained two simple classifiers, linear means (LM) and fisher's linear discriminate (FLD). Dowlin et al. [9] proposed a scheme, called CryptoNets, which used an fully homomorphic encryption scheme of Bos et al. [10] to evaluate deep convolutional neural networks (CNN) with two convolutional layers and two fully connected layers. Hesamifard et al. [11] proposed a CryptoDL scheme, which is a solution to run deep NN algorithms on encrypted data and allow the parties to provide/ receive the service without having to reveal their sensitive data to the other parties. The main work of CryptoDL is combine the CNN with leveled homomorphic encryption (LHE). Gao et al. [12] considered a situation that a user requests a naive Bayes classifier server, both the user and the server do not want to reveal their private data to each other. Their key technique involves the use of a "double-blinding" technique, and they shown how to combine it with additively homomorphic encryptions and oblivious transfer to hide both parties' privacy. There are also many other solutions by using other outsourcing computation techniques, such as [13–20].

2.2. Differential privacy in machine learning

Differential privacy [21,22] is a popular approach to privacy protection for machine learning algorithms on data sets, including Bayesian inference, empirical risk minimization (ERM), stochastic gradient descent (SGD), and so on. The main idea of differential privacy in machine learning is to learn a simple rule automatically from the distributional information of the data set at hand without revealing too much about any single individual in the data set. In fact, we often want to perform privacy-preserving machine learning as accurately as possible, just like we perform non-privacy-preserving machine learning on the same number of examples.

Dwork [23] first considered the original definition of ϵ-differential privacy protection, where the parameter ϵ (> 0) is a real number and controls how much information is disclosed about an individual's data through a statistical analysis and computation. Subsequently, several variations on the formal definition of differential privacy, such as computational differential privacy (CDP) [24] and differentially private consensus algorithm [25] have been proposed. Friedman and Schuster [26] considered machine learning within the framework of differential privacy. Under the condition of privacy and algorithmic requirements, they focused on decision tree induction as a case study. Abadi et al. [27] developed new algorithm techniques for learning and a refined analysis of privacy costs under the framework of differential privacy. In [28,29], the authors interested how to build differential-privacy algorithm within the Naive Bayes framework.

3. Preliminaries

In this section, we present some notations, cryptographic primitives and differential privacy that will be used throughout this paper.

3.1. Notations

Let $\mathbb{N}$, $\mathbb{Z}$ and $\mathbb{R}$ be sets of all natural numbers, all real numbers and all integer numbers, respectively. We denote by $\mathbb{R}^n$ the n-dimensional real space and by $\mathbb{R}^+(\mathbb{Z}^+)$ the space of all positive real (integer) numbers. Let $[1, n]$ be a set from 1 to a natural number n. Let p, q be two primes, and let $N = qp^2$. We write $\mathbb{Z}_p$ as a set of $\{0, 1, \ldots, p-1\}$, and $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$. Let $\mathcal{X} = \{x \in \mathbb{Z}_{p^2}^* | x = 1 \bmod p\}$ be the p-Sylow subgroup of $\mathbb{Z}_{p^2}^*$. We use # to denote the order of a set or an element, such as $\mathbb{Z}_{p^2}$ is a cyclic group with order $p(p-1)$, i.e., $\#\mathbb{Z}_{p^2} = p(p-1)$, and the order of $\mathcal{X}$ is $\#\mathcal{X} = p$. We define a function $\mathcal{L}$ over $\mathcal{X}$ as follows:

$$\mathcal{L} : \mathcal{X} \to \mathbb{Z}_p$$
$$\mathcal{L}(x) := \frac{x-1}{p}. \tag{1}$$

From the definition of $\mathcal{L}$, we can obtain a homomorphic property from multiplication to addition as the following lemma:

Lemma 1 (*Isomorphism, [30]*). *For any $a, b \in \mathcal{X}$, it has*

$$\mathcal{L}(ab \bmod p^2) = \mathcal{L}(a) + \mathcal{L}(a) \bmod p. \tag{2}$$

Corollary 1 (*[30]*). *For any $x \in \mathcal{X}$ such that $\mathcal{L}(x) \neq 0 \bmod p$ and $y = x^m \bmod p^2$ for $m \in \mathbb{Z}_p$, it has*

$$m = \frac{\mathcal{L}(y)}{\mathcal{L}(x)} = \frac{y-1}{x-1} \bmod p. \tag{3}$$

Definition 1 (*Negligible Functions*). *We say that a function $neg : \mathbb{N} \to \mathbb{R}$ is negligible if for every positive polynomial $poly(\cdot)$ and for all sufficiently large n,*

$$neg(n) < \frac{1}{poly(n)}. \tag{4}$$

Definition 2 (*One-Way*). A function $f : \{0, 1\}^* \to \{0, 1\}^*$ is called one-way if there existed a polynomial time machine A is easy to output $f(x)$ on input x, and if each probabilistic polynomial time machine A' is hard to find an invert of input y under f, the successful probability of A' maybe only with negligible in the length of y.

We use $|\cdot|$ to denote the size of data set D or the bit length of data x, and we use $\oplus$ to denote the addition mod 2 of the binary vectors. For a random variable or distribution S, let $s \leftarrow S$ denote that element s is selected uniformly at random from S according to its distribution. For a *probabilistic polynomial time* (PPT) algorithm A, we write $y \leftarrow A(x)$ if A output y on fixed input x according to A's distribution. Occasionally, we use $y \leftarrow A(x, r)$ to denote that y is computed by running the *deterministic time* (DT) algorithm A on input x and randomness r, which are chosen uniformly at random from some randomness space.

3.2. Diffie–Hellman and discrete logarithm problem over $\mathbb{Z}_N$

The Diffie–Hellman (DP) problem as a cryptographic primitive has been widely used in many cryptographic schemes. Let $\mathcal{P}(\kappa)$ be a set of all prime numbers with length κ. For any two distinct primes $p, q \in \mathcal{P}(\kappa)$, define $N = qp^2$. Let $\mathbb{G}_p = \{x \in \mathbb{Z}_N | \#(x^{p-1} \bmod p^2) = p\}$ be a set. The p-DH problem is defined below:

Definition 3 (*p-Diffie–Hellman, p-DH*). *Given three elements $a, b \leftarrow \mathbb{Z}_p^*$, $g \leftarrow \mathbb{G}_p$, and $(g^a \bmod N,\ g^b \bmod N)$, find $g^{ab} \bmod N$.*

From Definition 3, we know that the hardness of the Diffie–Hellman problem over $\mathbb{Z}_{qp^2}$ is based on the modulo size. If the size of exponent is $\kappa = 160$ bit, then it is sufficient for obtaining the current desired security on the DH problem. Therefore, the choice of κ should be not too small to be broken.

We use p-DL to denote the discrete logarithm (DL) problem over $\mathbb{Z}_N^*$, the formal definition is given below:

Definition 4 (*p-Discrete Logarithm, p-DL*). Given a set $\mathbb{G}_p$, an element $g \in \mathbb{G}_p$ and $g^a \bmod N$ for $a \in \mathbb{Z}_N$, find $a \bmod p$.

3.3. Public-Key encryption with a double decryption algorithm

Generally speaking, most of public-key encryption scheme generally has only one decryption algorithm. However, there are some special public-key encryption schemes that have a double decryption algorithm, denoted as DD-PKE. The formal definition of DD-PKE scheme is given as follows.

Definition 5 (*DD-PKE*). A public-key encryption scheme with a double decryption $\Pi = (\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{Enc}, \mathtt{uDec}, \mathtt{mDec})$ consists of the following PPT algorithms:

- $\mathtt{Setup}(1^\kappa)$. In setup algorithm, it takes the system security parameter κ as input and outputs a tuple (pp, msk), where pp is a public system parameter, which contains description of the plaintext space $\mathbb{P}$ and ciphertext space $\mathbb{C}$, and msk is the master secret key, which is only known to the master entity.
- $\mathtt{KeyGen}(pp)$. The key generation algorithm that generates the user's public key pk and secret key sk.
- $\mathtt{Enc}(pp, pk, m)$. The encryption algorithm takes the public system parameter pp, a user's public key pk and a message $m \in \mathbb{P}$ as input and outputs a ciphertext $c \in \mathbb{C}$.
- $\mathtt{uDec}(pp, sk, c)$. The user decryption algorithm takes public system parameter pp, the user's secret key sk and a ciphertext $c \in \mathbb{C}$ as input and returns a message $m \in \mathbb{P}$ or a special symbol $\perp$.
- $\mathtt{mDec}(pp, pk, msk, c)$. The master decryption algorithm takes the public system parameter pp, a user's public key pk, the master secret key msk and a ciphertext $c \in \mathbb{C}$ as input and returns a message $m \in \mathbb{P}$ or a special symbol $\perp$.

Please cite this article in press as: P. Li, et al., Privacy-preserving machine learning with multiple data providers, Future Generation Computer Systems (2018), https://doi.org/10.1016/j.future.2018.04.076.

According to the definition of DD-PKE, we know that for the key generation algorithm `KeyGen`, the user does not obtain the master secret key *msk* as input and *msk* is only kept by the master entity. Additionally, for the master decryption algorithm `mDec`, the master entity takes the user's public key *pk* as one of the inputs, which means that the `mDec` algorithm is dependent on the users' public key.

3.4. Differential privacy

Prior to defining the system model that we study, we will provide some notations. We assume a data set D with d attributes $\{\Gamma_1, \Gamma_2, \ldots, \Gamma_d\}$, the value domain of an attribute Γ_i by a real number $x_i \in \mathbb{R}$, and its size by $|D|$. Two data sets D and D' are said to be *neighbors* if they have the same cardinality and differ by at most one record. We use $D = \cup_{k=1}^{n} D_k$ to denote an aggregated data set, with sizes increasing according to the parameter k. Mechanism $\mathcal{M}$ is a randomized function mapping a data set D into an output in a range space, i.e., $\mathcal{M} : D \rightarrow Range(\mathcal{M})$, which is said to preserve differential privacy if any computational result from D and its neighbor D' will be statistically indistinguishable. Specifically, we provide a formal definition of differential privacy as follows:

Definition 6 (*ϵ-Differential Privacy, ϵ-DP [21]*). A random mechanism $\mathcal{M}$ is said to be ϵ-differential privacy if for any pair of neighboring data sets D and D' and for any possible anonymized data set O in output range space $Range(\mathcal{M})$,

$$Pr[\mathcal{M}(D) = O] \leq e^{\epsilon} \times Pr[\mathcal{M}(D') = O] \tag{5}$$

where the probability $Pr[\cdot]$ is taken over the randomness of mechanism $\mathcal{M}$ and also shows the risk of privacy disclosure.

In this definition, ϵ is a predefined privacy parameter for controlling the privacy budget, and it depends on the output of the statistical analysis and computation, the way in which the statistical analysis and computation are performed, and the information that the individual wants to hide. The smaller ϵ is, the stronger is the privacy protection. To achieve ϵ-DP, a private version of a function f needs to be constructed that maps a data set into numbers. These types of functions f are fundamental tools for statistical analysis and are called *numeric queries*. Typically, the numeric query has bounded *sensitivity*, and the maximum impact of a tuple on the output of f is called its *sensitivity*. The formal definition is given below.

Definition 7 (*Sensitivity*). Assume that f is a numeric query function that maps a data set D into a d-dimensional real space $\mathbb{R}^d$, i.e., $f : D \rightarrow \mathbb{R}^d$. For any pair of neighboring data sets D and D', the sensitivity f is defined as

$$\Delta f = \max_{D,D'} \|f(D) - f(D')\|_{L_1} \tag{6}$$

where $\|\cdot\|_{L_1}$ denotes the L_1 norm.

There are two standard mechanisms used to choose the statistical noise and achieve differential privacy: the Laplace mechanism and the exponential mechanism. Both of these mechanisms are based on the concept of the sensitivity of f. In this paper, we mainly consider the Laplace mechanism, which adds statistical noise drawn from a Laplace distribution to the data sets.

Theorem 1 (*Laplace Mechanism*). *Let $\sigma \in \mathbb{R}^+$, and f is a numeric query function that maps a domain D into a d-dimension real space $\mathbb{R}^d$, i.e., $f : D \rightarrow \mathbb{R}^d$. The computation $\mathcal{M}$*

$$\mathcal{M}(\mathbf{x}) = f(\mathbf{x}) + (Lap_1(\sigma), Lap_2(\sigma), \ldots, Lap_d(\sigma)) \tag{7}$$

provides ϵ-differential privacy, where the noise $Lap_i(\sigma)$ ($i \in [1, d]$) is drawn from the Laplace distribution with scaling parameter σ, whose density function is

$$p(\sigma) = \frac{1}{2\sigma} exp(-|x|/\sigma). \tag{8}$$

Here, the parameter $\sigma = \Delta f/\epsilon$ is controlled by the privacy budget ϵ and the function's sensitivity Δf.

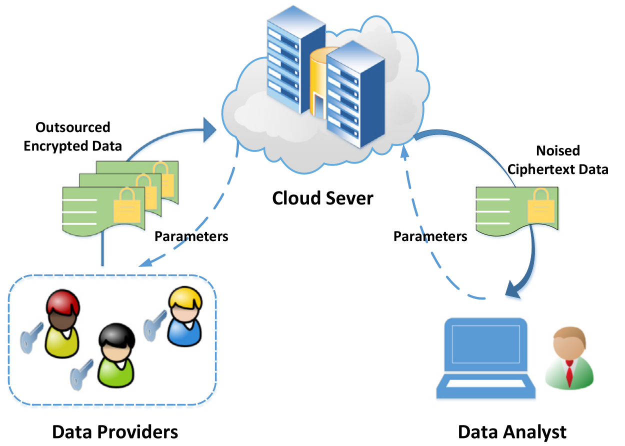


Fig. 1. System model under consideration.

4. System and adversary models

In this section, we present the definitions of our system model, problem statement and the adversary model.

4.1. System model

Our system consists of a data provider set $\mathcal{DP}$, a data analyst $\mathcal{DA}$ and a cloud server $\mathcal{C}$ (see Fig. 1).

- $\mathcal{DP}$ is a set of data providers, i.e., $\mathcal{DP} = \{P_1, P_2, \ldots, P_n\}$. Each data provider $P_i \in \mathcal{DP}$ uses its own public key pk_i to encrypt its sensitive data set D_i ($i \in [1, n]$) before outsourcing to $\mathcal{C}$.
- $\mathcal{C}$ as a semi-honest entity holds a data center, which provides unlimited storage space and powerful computation abilities for cloud users in this system. Furthermore, $\mathcal{C}$ can aggregate the combined data sets from the various cloud users and publish the data sets according to the task of $\mathcal{DA}$, such as query, classification and computation. Noting that $\mathcal{C}$ owns the data sets encrypted with *different public keys*.
- $\mathcal{DA}$ trains a machine learning model on the published data such that no data sets of participants are disclosed and no information is leaked about any single data set from the trained machine learning model.

4.2. Problem statement

In this paper, we consider the following problem: *Assume that each data provider $P_i \in \mathcal{DP}$ keeps data set $D_i = \{(\mathbf{x}_j^i, \mathbf{y}_j^i) \subset \mathbf{X} \times \mathbf{Y} : j \in [1, p_i], i \in [1, n]\}$. Each data D_i ($i \in [1, n]$) is of size p_i* with data vector $\mathbf{x}_j^i \in \mathbb{R}^d$, *and the corresponding binary label $y_j^i \in \mathbf{Y} := \{0, 1\}$. Due to privacy concerns, data providers $P_1, P_2, \ldots, P_n$ encrypt their local data sets before uploading to $\mathcal{C}$ for data storing and data processing. Based on these encryptions under different public* *keys, $\mathcal{C}$ generates a synthetic data set, in which different statistical noises are added according to different applications. We aim to release this synthetic data with ϵ-DP and to perform a privacy-preserving machine learning model on these synthetic data.*

In this scenario, we should consider the following challenges:

- To reduce the cost of key management, data providers $P_1, P_2, \ldots, P_n$ should be able to generate their own public and secret keys without communicating with $\mathcal{DA}$.
- Since $\mathcal{DA}$ holds the master secret key (independent of the data providers' individual secret keys) that allows any encrypted data set stored on the cloud to be decrypted. Hence, the interaction protocol between $\mathcal{C}$ and $\mathcal{DA}$ needs to do 'mask' processing.
- In order to support the computation over the ciphertext space, the used encryption scheme should be have the property of homomorphic or some malleability.

4.3. Adversary model

In this work, we assume that data provider $P_i \in \mathcal{DP}$ $(i \in [1, n])$, $\mathcal{C}$ and $\mathcal{DA}$ are semi-honest but untrusted. Additionally, we assume that there is no collusion between $\mathcal{C}$ and $\mathcal{DA}$, between any two data providers or between any data provider and $\mathcal{DA}$. Based on this security assumption, we present an active adversary $\mathcal{A}$ in our scheme. The aim of $\mathcal{A}$ is to obtain the plaintext of $\mathcal{DP}$'s data with the following abilities:

1. $\mathcal{A}$ is able to collude with $\mathcal{DA}$ to obtain plaintexts of all ciphertext data downloaded from $\mathcal{C}$ by running an interactive protocol.
2. $\mathcal{A}$ may corrupt $\mathcal{C}$ to guess the plaintexts of all ciphertext data outsourced from $P_i \in \mathcal{DP}$ $(i \in [1, n])$ and all data sent from $\mathcal{DA}$ by performing an interactive protocol.
3. $\mathcal{A}$ may corrupt some data providers of $\mathcal{DP}$ to generate plaintext information of other data providers' ciphertexts.

5. Our solution

In this section, we first present the main steps of our solution. We then describe in detail the construction of our solution based on the DD-PKE cryptosystem $\Pi_1 = (\texttt{Setup}, \texttt{KeyGen}, \texttt{Enc}, \texttt{uDec}, \texttt{mDec})$ and ϵ-DP. The DD-PKE cryptosystem Π_1 is based on the application of basic scheme in [31] to achieve CCA security in the random oracle model [32] by using the generic transformation proposed in [33].

1. **Initialization**. In this step, $\mathcal{DA}$ runs a `Setup` algorithm to set up the DD-PKE system and distributes the public system parameter pp to $\mathcal{C}$.
2. **DataUploading**. After obtaining the public system parameter pp sent from $\mathcal{C}$, data providers generate their own public/secret keys using the algorithm `KeyGen` and upload the data encrypted under their own associated public key to $\mathcal{C}$.
3. **NoiseAdding**. In this phase, according to the differential application or queries of $\mathcal{DA}$, cloud server $\mathcal{C}$ adds differential Laplace noise to these outsourced ciphertexts. Here, the Laplace noises are encrypted under the public key corresponding to outsourced ciphertexts. Later, $\mathcal{C}$ publishes these noise-added ciphertexts to $\mathcal{DA}$.
4. **Machine Learning-based ϵ-DP**. After downloading the noise-added ciphertexts from $\mathcal{C}$, $\mathcal{DA}$ can decrypt these ciphertexts using the `mDec` algorithm since he has the master private key msk. Then, $\mathcal{DA}$ keeps a synthetic data set with added noise. Based on this new data set, $\mathcal{DA}$ can learn a machine learning model with ϵ-DP.

5.1. Initialization

We stress that $\mathcal{DA}$ and $\mathcal{C}$ are semi-honest and are not colluding with each other. To set up the DD-PKE cryptosystem Π_1 and distribute the public system parameters to $\mathcal{C}$, $\mathcal{DA}$ runs the following algorithm `Setup` (illustrated in Algorithm 1):

Algorithm 1 Set up of DD-PKE cryptosystem Π_1

Input: a security parameter $\kappa \in \mathbb{N}$, two prime numbers p, q with κ bits length $(2^{\kappa-1} < p, q < 2^{\kappa})$, let $N = qp^2$. The cryptosystem uses a symmetric encryption scheme $SE = (Enc, Dec)$ with keys of length $s + (\kappa - 1)$ and also uses two hash functions $\mathcal{H} : \{0, 1\}^* \to \mathbb{Z}_{2^{\kappa-1}}$ and $\mathcal{G} : \mathbb{Z}_N \to \{0, 1\}^s \times \{0, 1\}^{\kappa-1}$, where $s \in \mathbb{Z}^+$.
Output: (pp, msk)
1: $\mathcal{DA}$ selects a random element $g \in \mathbb{Z}_N^*$ such that the order of $g_p := g^{p-1} \bmod p^2$ is p;
2: The public system parameters are $pp = (N, g, \mathcal{H}, \mathcal{G})$;
3: The master secret key is $msk = (p, q)$.

Here, we define the plaintext space as $\mathbb{P} := \{0, 1\}^s$ and the ciphertext space as $\mathbb{C} := \mathbb{Z}_N \times \{0, 1\}^s$. We say a encryption scheme $SE = (Enc, Dec)$ is a symmetric encryption, if the encryption mechanism Enc and decryption mechanism Dec are both deterministic, and the private key and public key are the same. For the efficiency, we use one-time padding as a symmetric encryption scheme SE, that is $Enc(\tau, x \parallel r) = \tau \oplus (x \parallel r)$. By running Algorithm 1, $\mathcal{DA}$ obtains the public system parameters pp and master secret key msk. Later, $\mathcal{DA}$ sends pp to $\mathcal{C}$ and keeps msk. $\mathcal{C}$ sends pp to n data providers $P_1, P_2, \ldots, P_n$, who can run the following algorithm `KeyGen` (illustrated in Algorithm 2) to generate their own pair of public and secret keys.

Algorithm 2 The key generation of DD-PKE cryptosystem Π_1

Input: The public system parameters $pp = (N, g, \mathcal{H}, \mathcal{G})$,
Output: (pk_i, sk_i), where $i \in [1, n]$
1: Data provider $P_i \in \mathcal{DP}$ chooses a random element $sk_i \in \{0, 1, \cdots, 2^{\kappa-1} - 1\}$ with bit length $\kappa - 1$;
2: Data provider $P_i \in \mathcal{DP}$ computes $pk_i = g^{sk_i} \bmod N$;
3: The public-secret key pair of data provider $P_i \in \mathcal{DP}$ is (pk_i, sk_i).

5.2. Data uploading

Assume that any data provider $P_i \in \mathcal{DP}$ has a sensitive data set $D_i = \{(\mathbf{x}_j^i, y_j^i) \subset \mathbf{X} \times \mathbf{Y} : j \in [1, p_i], i \in [1, n]\}$, which is of size p_i with data vector $\mathbf{x}_j^i \in \mathbb{R}^d$, and the corresponding binary label is $y_j^i \in \mathbf{Y} := \{0, 1\}$, where $\mathbf{x}_j^i = (x_{j1}^i, x_{j2}^i, \ldots, x_{jd}^i)$. To preserve the privacy of sensitive data set D_i, each data provider $P_i \in \mathcal{DP}$ needs to encrypt it by running the algorithm `Enc` (outlined in Algorithm 3):

From the above Algorithm 3, a vector $\mathbf{x}_j^i = (x_{j1}^i, x_{j2}^i, \ldots, x_{jd}^i)$ encrypted under pk_i can be presented by

$$
\begin{aligned}
&\texttt{Enc}(pp, pk_i, \mathbf{x}_j^i) = [\mathbf{x}_j^i]_i = \{[x_{jv}^i]_i\}_{v=1}^d \\
&= ([x_{j1}^i]_i, [x_{j2}^i]_i, \ldots, [x_{jd}^i]_i) \\
&= ((A_{j1}^i, B_{j1}^i), (A_{j2}^i, B_{j2}^i), \ldots, (A_{jd}^i, B_{jd}^i)) \\
&= \{(A_{jv}^i, B_{jv}^i)\}_{v=1}^d.
\end{aligned}
$$

Here, we use the notation $(\mathbf{A}_j^i, \mathbf{B}_j^i)$ to denote a encryption vector $[\mathbf{x}_j^i]_i = \{(A_{jv}^i, B_{jv}^i)\}_{v=1}^d$. Therefore, the encryption of sensitive data set D_i is computed by $\texttt{Enc}(pp, pk_i, D_i) = [D_i]_i = ([\mathbf{x}_j^i]_i, [y_j^i]_i) = ((\mathbf{A}_j^i, \mathbf{B}_j^i), [y_j^i]_i)$. To improve the efficiency of the data processing and

Algorithm 3 The encryption of DD-PKE cryptosystem Π_1

Input: The public system parameters $pp = (N, g, \mathcal{H}, \mathcal{G})$, the public key pk_i and the message $\mathbf{x}_j^i = (x_{j1}^i, x_{j2}^i, \cdots, x_{jd}^i)$, where $i \in [1, n]$, $j \in [1, p_i]$ and $v \in [1, d]$

Output: $[\mathbf{x}_j^i]_i$

1: Data provider $P_i \in \mathcal{DP}$ selects a random element $r_{jv}^i \in \mathbb{Z}_{2^{\kappa-1}}$ with bit length $\kappa - 1$, where $v \in [1, d]$;
2: Data provider $P_i \in \mathcal{DP}$ computes $h_{jv}^i = \mathcal{H}(x_{jv}^i \parallel r_{jv}^i)$ and $\tau_{jv}^i = \mathcal{G}(pk_i^{h_{jv}^i} \bmod N)$ for each component x_{jv}^i of $\mathbf{x}_j^i$;
3: Data provider $P_i \in \mathcal{DP}$ computes $A_{jv}^i = g^{h_{jv}^i} \bmod N$ and $B_{jv}^i = Enc(\tau_{jv}^i, x_{jv}^i \parallel r_{jv}^i) = \tau_{jv}^i \oplus (x_{jv}^i \parallel r_{jv}^i)$;
4: The ciphertext of x_{jv}^i under the public key pk_i is $[x_{jv}^i]_i = (A_{jv}^i, B_{jv}^i)$;
5: The ciphertext vector of $\mathbf{x}_j^i$ under the public key pk_i is $[\mathbf{x}_j^i]_i = \{[x_{jv}^i]_i\}_{v=1}^d = ([x_{j1}^i]_i, [x_{j2}^i]_i, \cdots, [x_{jd}^i]_i)$.

to keep the privacy of data processing, any data provider $P_i \in \mathcal{DP} (i \in [1, n])$ should first certify the sensitivity of a query function f_i and privacy level ϵ_i for his sensitive data set D_i. Thus, each data provider $P_i \in \mathcal{DP}$ uploads Δf_i, ϵ_i and pk_i in addition to ciphertext $[D_i]_i$ to the cloud server $\mathcal{C}$, where $i \in [1, n]$.

Recall that a data set D_i is tuple of d attributes and that each attribute value is taken from a real space $\mathbb{R}$. However, the DD-PKE cryptosystem Π_1 in this paper has a plaintext space $\mathbb{P} = \{0, 1\}^s$, and for simplicity, we continue to use D_i and x_{jv}^i to represent the binary bit of sensitive data set D_i and record x_{jv}^i, respectively.

Remark 1. If $P_i \in \mathcal{DP}$ $(i \in [1, n])$ wants to decrypt his/her ciphertext $[\mathbf{x}_j^i]_i = (\mathbf{A}_j^i, \mathbf{B}_j^i) = \{(A_{jv}^i, B_{jv}^i)\}_{v=1}^d = \{[x_{jv}^i]_i\}_{v=1}^d$, he/she can use the Algorithm 4, the user decryption algorithm uDec to decrypt it. The special symbol "$\perp$" denotes the fact that the ciphertext was rejected.

Algorithm 4 The user decryption of DD-PKE cryptosystem Π_1

Input: The public system parameters $pp = (N, g, \mathcal{H}, \mathcal{G})$, public key pk_i, and user private key sk_i and the ciphertext vector $[\mathbf{x}_j^i]_i = \{[x_{jv}^i]_i\}_{v=1}^d = ([x_{j1}^i]_i, [x_{j2}^i]_i, \cdots, [x_{jd}^i]_i)$, where $[x_{jv}^i]_i = (A_{jv}^i, B_{jv}^i)$, $i \in [1, n]$, $j \in [1, p_i]$ and $v \in [1, d]$

Output: $\mathbf{x}_j^i$

1: Data provider $P_i \in \mathcal{DP}$ computes $\tau_{jv}^i = \mathcal{G}({A_{jv}^i}^{sk_i} \bmod N)$ for each component $[x_{jv}^i]_i = (A_{jv}^i, B_{jv}^i)$ of $[\mathbf{x}_j^i]_i$;
2: Data provider $P_i \in \mathcal{DP}$ computes $x_{jv}^i \parallel r_{jv}^i = Dec(\tau_{jv}^i, B_{jv}^i) = B_{jv}^i \oplus \tau_{jv}^i$ and $h_{jv}^i = \mathcal{H}(x_{jv}^i \parallel r_{jv}^i)$;
3: Data provider $P_i \in \mathcal{DP}$ checks

$$\texttt{uDec}(pp, sk_i, [x_{jv}^i]_i) = \begin{cases} x_{jv}^i, & \text{if } A_{jv}^i = g^{h_{jv}^i}, \\ \perp, & \text{otherwise.} \end{cases}$$

5.3. Noise addition

After the data uploading phase, $\mathcal{C}$ collects data sets encrypted with *different public keys*, i.e., $[D_1]_1, [D_2]_2, \ldots, [D_n]_n$. Because $\mathcal{DA}$ keeps the master secret key *msk*, it can decrypt any valid ciphertext. Therefore, to securely publish data, $\mathcal{C}$ must perform some transformation of the uploaded data set before publishing to $\mathcal{DA}$.

For each uploaded data set $[D_i]_i$ of data provider $P_i \in \mathcal{DP}$, $\mathcal{C}$ generates $\boldsymbol{\eta}_j^i = (\eta_{j1}^i, \eta_{j2}^i, \ldots, \eta_{jd}^i)$, a d-dimensional noise vector sampled from a Laplace distribution with parameter $\Delta f_i/\epsilon_i$, where $i \in [1, n]$, $j \in [1, p_i]$. Then, it encrypts this noise vector $\boldsymbol{\eta}_j^i$ as $[\boldsymbol{\eta}_j^i]_i = ([\eta_{j1}^i]_i, [\eta_{j2}^i]_i, \ldots, [\eta_{jd}^i]_i) = \{[\eta_{jv}^i]_i\}_{v=1}^d = \{(C_{jv}^i, D_{jv}^i)\}_{v=1}^d = (\mathbf{C}_j^i, \mathbf{D}_j^i)$. According to the additively homomorphic property of the DD-PKE scheme Π_1, for each P_i's uploaded data set $[D_i]_i$, the computation of $[\mathbf{x}_j^i]_i \otimes [\boldsymbol{\eta}_j^i]_i$ over the encrypted domain can be computed as $[\mathbf{x}_j'^i]_i = [\mathbf{x}_j^i + \boldsymbol{\eta}_j^i]_i = (\mathbf{A}_j^i + \mathbf{D}_j^i, \mathbf{C}_j^i + \mathbf{D}_j^i) = (\mathbf{A}_j'^i, \mathbf{B}_j'^i)$. Therefore, $\mathcal{C}$ publishes ciphertext data set $[\hat{D}_i]_i$ to $\mathcal{DA}$, where $[\hat{D}_i]_i = ((\mathbf{A}_j'^i, \mathbf{B}_j'^i), [y_j'^i]_i)$ is the ciphertext data set of $[D_i]_i$ $(i \in [1, n])$ with added noise.

5.4. Learning-based ϵ-differential privacy

In this phase, $\mathcal{DA}$ downloads only noise-added ciphertexts of the data sets $[\hat{D}_1]_1, [\hat{D}_2]_2, \ldots, [\hat{D}_n]_n$. Because the master decryption algorithm mDec depends on the factoring information of N and $\mathcal{DA}$ keeps the master secret key *msk*, it can decrypt a valid ciphertext with a corresponding data provider's public key. Hence, to decrypt each noise-added encryption of $[\hat{D}_i]_i = ([\mathbf{x}_j'^i]_i, [y_j^i]_i) = ((\mathbf{A}_j'^i, \mathbf{B}_j'^i), [y_j^i]_i)$, $\mathcal{DA}$ runs the Algorithm 5, the master decryption algorithm mDec to obtain the message:

Algorithm 5 The master decryption of DD-PKE cryptosystem Π_1

Input: The public system parameters $pp = (N, g, \mathcal{H}, \mathcal{G})$, public key pk_i, and master key *msk* and the ciphertext vector $[\mathbf{x}_j'^i] = \{[x_{jv}'^i]_i\}_{v=1}^d = ([x_{j1}'^i]_i, [x_{j2}'^i]_i, \cdots, [x_{jd}'^i]_i)$, where $[x_{jv}'^i]_i = (A_{jv}'^i, B_{jv}'^i)$, $i \in [1, n]$, $j \in [1, p_i]$ and $v \in [1, d]$

Output: $\mathbf{x}_j'^i$

1: $\mathcal{DA}$ computes the secret key sk_i of $P_i \in \mathcal{DP}$, i.e., $sk_i = \frac{\mathcal{L}(pk_i^{p-1} \bmod p^2)}{\mathcal{L}(g_p)} \bmod p$;
2: $\mathcal{DA}$ checks whether sk_i is smaller than $2^{\kappa-1}$;
3: $\mathcal{DA}$ computes $\tau_{jv}'^i = \mathcal{G}({A_{jv}'^i}^{sk_i} \bmod N)$, $x_{jv}'^i \parallel r_{jv}'^i = Dec(\tau_{jv}'^i, B_{jv}'^i) = B_{jv}'^i \oplus \tau_{jv}'^i$ and $h_{jv}'^i = \mathcal{H}(x_{jv}'^i \parallel r_{jv}'^i)$;
4: $\mathcal{DA}$ checks

$$\texttt{mDec}(pp, pk_i, msk, [x_{jv}'^i]_i) = \begin{cases} x_{jv}'^i, & \text{if } A_{jv}'^i = g^{h_{jv}'^i}, \\ \perp, & \text{otherwise.} \end{cases}$$

Recall that $g_p = g^{p-1} \bmod p^2$ with order p; we have $(g_p)^p = g^{p(p-1)} \bmod p^2 = 1 \bmod p^2$. We can see that g is a primitive root mod p^2; then, there exists $b \in \mathbb{Z}_p^*$ such that $g^{p-1} = 1 + pb \bmod p^2$, i.e., $g^{p-1} \in \mathcal{X}$, $\mathcal{L}(g^{p-1}) = \frac{(1+pb)-1}{p} = b \bmod p$. Hence, $g_p = 1 + bp \bmod p^2$. For any element $u \in \mathbb{Z}_p$, compute $g_p^u \bmod p^2 = (1 + ubp) \bmod p^2$, which is not equal to 1. If $ub > p$, then we can find two integers $a', b' \in \mathbb{Z}$ such that $ub = a'p + b'$ with $b' < p$. Therefore, we have $b'b^{-1} = u \bmod p$ and $g_p^u \bmod p^2 = (1 + ubp) \bmod p^2 = 1 + b'p$. According to Corollary 1, the exponent $u \in \mathbb{Z}_p$ can be computed by

$$\begin{aligned} \frac{\mathcal{L}(g_p^u \bmod p^2)}{\mathcal{L}(g_p)} \bmod p =& \frac{\mathcal{L}(1 + b'p)}{\mathcal{L}(1 + bp)} \bmod p \\ =& \frac{b'}{b} \bmod p = u. \end{aligned}$$

Here, $\mathcal{DA}$ checks if the size of private key sk_i is less than $2^{\kappa-1}$. If the private key sk_i is smaller than $2^{\kappa-1}$ $(sk_i < 2^{\kappa-1} < p)$, then the master decryption performs correctly. Otherwise, the master decryption outputs a special symbol "$\perp$".

Based on this fact, $\mathcal{DA}$ computes the secret key $sk_1, sk_2, \ldots, sk_n$ of data providers $P_1, P_2, \ldots, P_n$, respectively. Therefore, $\mathcal{DA}$ knows the factorization of N, i.e., $msk = (p, q)$, can use the master

Please cite this article in press as: P. Li, et al., Privacy-preserving machine learning with multiple data providers, Future Generation Computer Systems (2018), https://doi.org/10.1016/j.future.2018.04.076.

Table 1
Performance of the cryptosystem Π_1.

κ	$N = (p, q)$	g	*Plaintext*	*Ciphertext*	*Time* (s)
4	$N = (11, 13)$	1759	7 bit	19 bit	16.18
5	$N = (29, 23)$	15 671	8 bit	23 bit	0.79
6	$N = (43, 37)$	50 829	8 bit	26 bit	1.51
8	$N = (241, 193)$	10 636 571	6 bit	30 bit	112.83

decryption algorithm mDec to decrypt each component $[x'^{i}_{jv}]_i$ of ciphertext $[\mathbf{x}'^{i}_j]_i$ and obtain noise-added data sets $\hat{D}_i$; it has the form $\mathbf{x}'^{i}_j = \mathbf{x}^i_j + \boldsymbol{\eta}^i_j$, where $i \in [1, n]$ and $j \in [1, p_i]$. Now, $\mathcal{DA}$ possesses the noise-added data sets $\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_n$. Assume that $\mathcal{DA}$ wants to compute $f(\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_n)$ for an n-input function f; then, it can perform any process on these data sets, as described in Section 6.

6. Simulation results

In this section, we show how we use our scheme to preserve data privacy according to the DD-PKE cryptosystem Π_1 and ϵ-DP. On the one hand, the performance of DD-PKE cryptosystem Π_1 is conducted on a PC with an Intel(R) Core(TM) i7-6500U CPU with 2.59 GHz and 8 GB of RAM. To perform the cryptosystem Π_1, all programs are built in MAGMA. We firstly choose a security parameter to test the operations in cryptosystem Π_1. Secondly, we randomly choose two primes p and q from the interval $(2^{\kappa-1} + 1, 2^{\kappa} - 1)$ and let $N = p^2 q$. Thirdly, we generate $g \in \mathbb{Z}_N^*$ such that $\#(g^{p-1} \bmod N) = p$. As illustrated in Table 1, when we choose four security parameters $\kappa = 4, 5, 6, 8$, the integer $N = (p, q)$ and element g is $(11, 13), (29, 23), (43, 37), (241, 193)$ and 1759, 15 671, 50 829,10 636 571, respectively. Accordingly, the size of plaintext space and ciphertext space are $s = 7, 8, 8, 6$ bit and $log_N + s = 19, 23, 26, 30$ bit. In the last column of Table 1, the performance time is given. If the security parameter $\kappa = 8$ and plaintext bit $s = 6$, the performance time in fourth line (112.83) is much higher than the first three lines.

On the other hand, all simulations of ϵ-DP are conducted on a PC with an AMD A4-3300M APU with Radeon(TM) HD Graphics 1.90 GHz and 6 GB of RAM. We treat the **Abalone**, **Wine**, **Cpu**, **Glass** and **Krkopt** data sets as our test data sets which can be downloaded from the UCI Machine Learning Repository, and use it to train the machine learning algorithms. To perform the ϵ-DP over the above five data sets, all programs are built in Java. To simulate the K-nearest neighbor (K-NN) classifier, Support vector machine (SVM), Random forest and Naive Bayes, we withdraw $\frac{1}{10}$ of the data sets' records to compose the test data set. Additionally, we choose the privacy level $\epsilon = 0.1$ to apply the Laplace mechanism to our five data sets (see Fig. 2).

7. Security analysis

In this section, we first present the security analysis of the basic cryptographic encryption primitive and ϵ-DP before analyzing the security of our PMLM scheme.

7.1. Analysis of encryption primitive

In this section, we give some a secure analysis of the DD-PKE cryptosystem Π_1.

Definition 8 (*IND-CCA2*)**.** Let $\Pi_1 = (\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{Enc}, \mathtt{uDec}, \mathtt{mDec})$ be a DD-PKE cryptosystem and let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a PPT adversary. For $1^\kappa \in \mathbb{N}$, let

$$\begin{aligned}\mathtt{Adv}^{ind-cca2}_{\Pi_1,\mathcal{A}}(1^\kappa) = 2Pr[(pk, sk) \leftarrow \mathtt{KeyGen}(1^\kappa);\\ (m_0, m_1, \mathtt{state}) \leftarrow \mathcal{A}_1^{\mathcal{O}_1,\mathcal{G},\mathcal{H}}(pk);\\ b \leftarrow \{0, 1\}; y \leftarrow \mathtt{Enc}(pk, m_b):\\ \mathcal{A}_2^{\mathcal{O}_2,\mathcal{G},\mathcal{H}}(pk, m_0, m_1, \mathtt{state}, y) = b] - 1\end{aligned}$$

where $\mathcal{O}_1(\cdot)$ and $\mathcal{O}_2(\cdot)$ are decryption oracles, and $'\mathtt{state}'$ is secret information, possibly including the public key pk. The adversary $\mathcal{A}_1$ outputs messages m_0 and m_1 with the same length $|m_0| = |m_1|$.

We say the DD-PKE cryptosystem Π_1 is IND-CCA2 secure if $\mathtt{Adv}^{ind-cca2}_{\Pi_1,\mathcal{A}}(1^\kappa)$ is negligible.

The formal definition of the one-time encryption (OTE) is as follows:

Definition 9 (*OTE*)**.** Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a PPT adversary. Define

$$\begin{aligned}\mathtt{Adv}^{OTE}_{SE,\mathcal{A}} = 2Pr[\kappa \leftarrow \{0, 1\}^l;\\ (m_0, m_1, \mathtt{state}) \leftarrow \mathcal{A}_1(\cdot);\\ b \leftarrow \{0, 1\}; y \leftarrow Enc(\kappa, m_b):\\ b \leftarrow \mathcal{A}_2(m_0, m_1, \mathtt{state}, y)] - 1.\end{aligned}$$

where the outputs m_0 and m_1 of adversary $\mathcal{A}_1$ have the same length. We say that a symmetric encryption $SE = (Enc, Dec)$ is OTE secure if $\mathtt{Adv}^{OTE}_{SE,\mathcal{A}}$ is negligible.

According to Definition 4, we obtain the following theorem:

Theorem 2. *p-DL problem over $\mathbb{Z}_N^*$ is intractable if and only if the factoring $N = qp^2$ is intractable.*

The construction of cryptosystem Π_1 uses two hash functions which are modeled as random oracles in the security analysis. In general, DL problem is hard to solve than DH problem. Hence, we can believe cryptosystem Π_1 is an enhanced ElGamal type encryption scheme which is based its security on the computational p-DH problem. Therefore, we can obtain the following theorem:

Theorem 3 (*One-Way*)**.** *In the random oracle model, the DD-PKE cryptosystem $\Pi_1 = (\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{Enc}, \mathtt{uDec}, \mathtt{mDec})$ is one-way if the p-DH problem is intractable.*

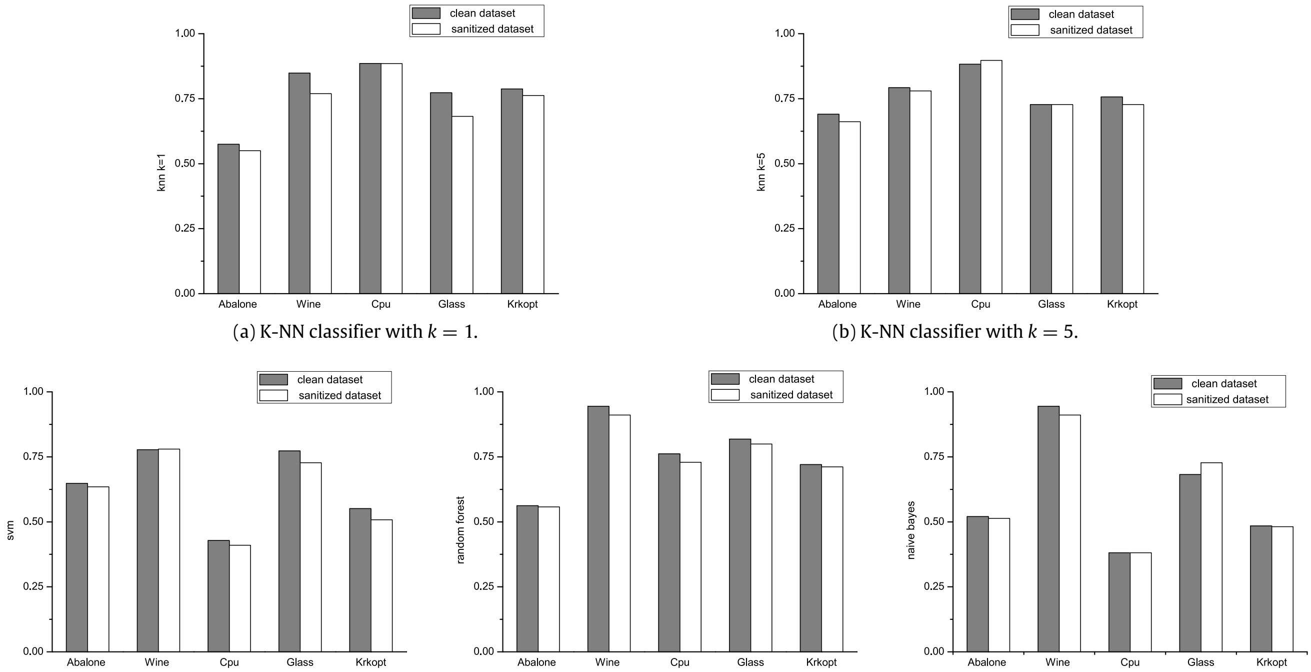
Proof. Assume that the p-DH problem is not intractable. A PT machine $\mathcal{A}$ can solve the p-DH problem. By using the machine $\mathcal{A}$, we can make a PT machine $\mathcal{A}'$. This machine will run $\mathcal{A}$ as a subroutine and break the one-wayness of the DD-PKE cryptosystem Π_1. Let the public key and challenge ciphertext be $(N, g, g^{sk} \bmod N)$ and $(A = g^h \bmod N, B = \mathcal{G}(g^{skh} \bmod N) \oplus (m \parallel r))$ respectively, where h is a string by asking the random oracle $\mathcal{H}$ on a query $m \parallel r$. Because $\mathcal{A}$ can compute $g^{skh} \bmod N$ from $(g^{sk} \bmod N, A = g^h \bmod N)$ and obtain the value τ by querying the random oracle $\mathcal{G}$ with a query $(g^{skh} \bmod N)$. Then, $\mathcal{A}'$ can compute $B \oplus \tau$ and extract the first s-bit of $B \oplus \tau$ as the corresponding plaintext m. Therefore, the DD-PKE cryptosystem Π_1 is not one-way. □

Due to the work of [33] and [34], we have the following result:

Theorem 4. *In the random oracle model, the DD-PKE cryptosystem Π_1 is IND-CCA2 secure if the computational p-DH problem is intractable and the SE is OTE secure for a message $m \in \mathbb{P}$ and randomness $r \in \mathbb{Z}_{2^{\kappa-1}}$.*

Proof. The details of the proof is given in [33]. Here, we give a simple proof as follows.

Assume that the p-DH problem is not intractable and that the symmetric encryption scheme $SE = (Enc, Dec)$ is not OTE secure.



(a) K-NN classifier with $k = 1$.

(b) K-NN classifier with $k = 5$.

(c) SVM classifier.

(d) Random forest.

(e) Naive Bayes.

Fig. 2. Simulation results.

There exists a polynomial time machine $\mathcal{A}$ which solves the p-DH problem and it must break the symmetric encryption scheme SE with non-negligible probability. We construct two polynomial time machines $\mathcal{A}'$ and $\mathcal{A}''$ with the help of $\mathcal{A}$ (both of them run $\mathcal{A}$ as a subroutine), which can break the IND-CCA2 security of DD-PKE cryptosystem Π_1. Let the public key and challenge ciphertext be $(N, g, pk = g^{sk} \bmod N)$ and $(m_0 \parallel r_0, m_1 \parallel r_1, c^* = (A^* = g^h \bmod N, B^* = (\mathcal{G}(g^{skh} \bmod N) \oplus (m_b \parallel r_b))))$, respectively, where $b \in \{0, 1\}$ and h is a random oracle value of $\mathcal{O}_{\mathcal{H}}$. Then $\mathcal{A}$ computes $g^{skh} \bmod N$ from the pair $(pk = g^{sk} \bmod N, A^* = g^h \bmod N)$ since he can break the p-DH problem. Hence, $\mathcal{A}'$ obtains the corresponding value $\tau^* = \mathcal{O}_{\mathcal{G}}(g^{skh} \bmod N)$ by querying the random oracle $\mathcal{O}_{\mathcal{G}}$.

Meanwhile, we assume that the polynomial time machine $\mathcal{A}$ can break the OTE of SE with non-negligible probability. $\mathcal{A}$ accesses to the decryption oracle $\mathcal{O}_{Dec}$ to ask queries. Hence, the constructed $\mathcal{A}''$ runs $\mathcal{A}$ as a subroutine (an oracle) and make the answer by himself when $\mathcal{A}$ make access to $\mathcal{O}_{Dec}$ with a query. Given the $(m_0 \parallel r_0, m_1 \parallel r_1, c^* = (A^*, B^*))$, $\mathcal{A}$ makes $\mathcal{O}_{Dec}$ query of c^* and obtains m', and outputs b. Finally, $\mathcal{A}''$ outputs the answer b with the non-negligible probability, and the correct answer immediately implies whether c^* is $\mathtt{Enc}(pp, pk, m_0 \parallel r_0)$ or $\mathtt{Enc}(pp, pk, m_1 \parallel r_1)$. □

7.2. Analysis of ϵ-differential privacy

In this subsection, we show the differential privacy of the set $\hat{D}$, which consists of disjoint data sets, independent of the actual data sets, $\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_n$, and the ultimate privacy level depends on the worst of the guarantees of each analysis. This fact can be described as the following theorem:

Theorem 5 (*Parallel Composition*). *Let $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_n$ be n mechanisms, where each mechanism $\mathcal{M}_i$ ($i \in [1, n]$) provides ϵ_i-DP. Let $D_1, D_2, \ldots, D_n$ be n arbitrary disjoint data sets of the input domain D. For a new mechanism $\mathcal{M}$, the sequence of $\mathcal{M}(\mathcal{M}_1(D_1), \mathcal{M}_2(D_2), \ldots, \mathcal{M}_n(D_n))$ provides $(\max_{1\le i\le n}\epsilon_i)$-DP.*

Proof. For any sequence r of outcomes $\hat{D}_i \in Rang(\mathcal{M}_i)$, let $\hat{D}_i$ be mechanism $\mathcal{M}_i$ applied to data set D_i, i.e., $\hat{D}_i = \mathcal{M}_i^r(D_i)$. The probability of output $\hat{D}_i$ from the sequence of $\mathcal{M}_i^r(D_i)$ is

$$Pr[\mathcal{M}(A) = r] = \prod_i Pr[\mathcal{M}_i^r(D_i) = \hat{D}_i].$$

We know that if x is smaller than one, then $e^{\epsilon} \approx 1+x$. According to Definition 6, we have that

$$Pr[\mathcal{M}(A) = r] \le Pr[\mathcal{M}(B) = r] \times e^{\epsilon \times |A \boxplus B|},$$

since the triangle inequality $\||A|-|B|\| \ge |r-|B|\| - \||A|-r|$, and $\||A|-|B|\| \le |A \boxplus B|$, where $\boxplus$ denotes the symmetric difference between data sets A and B and $|A \boxplus B|$ denotes the shifted count from data set A to data set B. For each $\mathcal{M}_i^r(D_i)$, by using the definition of ϵ-DP,

$$\begin{aligned} &\prod_i Pr[\mathcal{M}_i^r(A_i) = \hat{D}_i] \\ &\le \prod_i Pr[\mathcal{M}_i^r(B_i) = \hat{D}_i] \times \prod_i e^{\epsilon \times |A_i \boxplus B_i|} \\ &\le \prod_i Pr[\mathcal{M}_i^r(B_i) = \hat{D}_i] \times e^{\epsilon \times |A \boxplus B|}. \quad \square \end{aligned}$$

7.3. Analysis of PMLM scheme

On the one hand, our PMLM scheme is based on DD-PKE cryptosystem Π_1, which uses two hash functions $\mathcal{G} : \mathbb{Z}_N \to \{0, 1\}^s \times \{0, 1\}^{\kappa-1}$ and $\mathcal{H} : \{0, 1\}^* \to \mathbb{Z}_{2^{\kappa-1}}$ and symmetric encryption function SE in the encryption function. The security proof is given in Theorem 4. The previous double decryption cryptosystem, denoted by BCP [35], is semantically secure in the standard model. In addition, we can also show that our PMLM scheme can against the adversary described in Section 4.3. Here, we give the analysis as follows:

- If $\mathcal{A}$ has corrupted $\mathcal{DA}$ or $\mathcal{C}$ to obtain the outsourced data, $\mathcal{A}$ cannot obtain the corresponding plaintext by using the master key *msk* because of the IND-CCA2 security of our PMLM scheme.
- Assume that $\mathcal{A}$ has corrupted some data providers of $\mathcal{DP}$ and obtains the public/private keys of these corrupted data providers. Due to the non-interactive and independent key generation algorithm KeyGen of the data provider in $\mathcal{DP}$, these multiple private/public keys are uncorrelated. Therefore, $\mathcal{A}$ still cannot decrypt the ciphertext.

On the other hand, our PMLM scheme is also support ϵ-DP. Noting that some query functions are non-linear functions, such as exponential operation, derivative operation and so on. These operations cannot be directly performed over the encrypted domain. Although this non-linear function can be computed by interpolation, fitting and approximation of a polynomial, the cost of computation over the encrypted domain is higher than the computation over the plaintext domain. Based on this fact, our PMLM scheme improves the efficiency and accuracy of data processing. This is because PMLM scheme transforms the data encrypted under *different public keys* into noise-added plaintext data. $\mathcal{DA}$ can perform machine learning model on this noise-added plaintext domain with ϵ-DP and without information leakage.

8. Conclusion and future work

In this paper, we proposed PMLM, a scheme for privacy-preserving machine learning under multiple keys, which allows multiple data providers to outsource encrypted data sets to a cloud server for data storing and processing. In our work, the cloud server can add different statistical noises to the outsourced data sets according to the different queries of the data analyst, which is different from existing works (i.e., data providers add statistical noise by themselves). Our work is mainly based on DD-PKE cryptosystem Π_1 and ϵ-DP, which can be proven to achieve the goal of outsourced computation on multi-party's data sets without privacy leakage in the random oracle model.

Many important works have shown that differential privacy is an effective and useful tool for data privacy calculations. As a further research work, we hope that our PMLM scheme will be useful in both the application domain and theory domain of privacy-preserving machine learning.

Acknowledgments

This work was supported by Natural Science Foundation of Guangdong Province for Distinguished Young Scholars, China (2014A030306020), Guangzhou scholars project for universities of Guangzhou, China (No. 1201561613), Science and Technology Planning Project of Guangdong Province, China (2015B010129015), National Natural Science Foundation of China (No. 61472091, No. 61702126) and National Natural Science Foundation for Outstanding Youth Foundation, China (No. 61722203).

References

[1] Z. Liu, Z. Wu, T. Li, J. Li, C. Shen, Gmm and cnn hybrid method for short utterance speaker recognition, IEEE Trans. Ind. Inf. (2018). http://dx.doi.org/10.1109/TII.2018.2799928.
[2] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[3] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an, H. Ye, Android malware detection, IEEE Trans. Ind. Inf. (2018). http://dx.doi.org/10.1109/TII.2017.2789219.
[4] N.D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E.E. Papalexakis, C. Faloutsos, Tensor decomposition for signal processing and machine learning, IEEE Trans. Signal Process. 65 (13) (2017) 3551–3582.
[5] T. Chen, S. Zhong, Privacy-preserving backpropagation neural network learning, IEEE Trans. Neural Netw. 20 (10) (2009) 1554–1564.
[6] 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.
[7] S. Samet, A. Miri, Privacy-preserving back-propagation and extreme learning machine algorithms, Data Knowl. Eng. 79 (2012) 40–61.
[8] T. Graepel, K. Lauter, M. Naehrig, ML confidential: Machine learning on encrypted data, in: International Conference on Information Security and Cryptology, Springer, Berlin, Heidelberg, 2012, pp. 1–21.
[9] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, J. Wernsing, Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy, in: International Conference on Machine Learning, 2016, pp. 201–210.
[10] J.W. Bos, K. Lauter, J. Loftus, M. Naehrig, Improved security for a ring-based fully homomorphic encryption scheme, in: IMA International Conference on Cryptography and Coding, Springer, Berlin, Heidelberg, 2013, pp. 45–64.
[11] E. Hesamifard, H. Takabi, M. Ghasemi, CryptoDL: Deep neural networks over encrypted data, 2017. ArXiv preprint ArXiv:1711.05189.
[12] C. Gao, Q. Cheng, P. He, W. Susilo, J. Li, Privacy-preserving naive bayes classifiers secure against the substitution-then-comparison attack, Inform. Sci. (2018). http://dx.doi.org/10.1016/j.ins.2018.02.058.
[13] P. Li, J. Li, Z. Huang, T. Li, C. Gao, S. Yiu, K. Chen, Multi-key privacy-preserving deep learning in cloud computing, Future Gener. Comput. Syst. 74 (2017) 76–85.
[14] P. Li, J. Li, Z. Huang, C. Gao, W. Chen, K. Chen, Privacy-preserving outsourced classification in cloud computing, Cluster Comput. (2017) 1–10.
[15] 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.
[16] X. Chen, X. Huang, J. Li, J. Ma, W. Lou, D.S. Wong, New algorithms for secure outsourcing of large-scale systems of linear equations, IEEE Trans. Inf. Forensics Secur. 10 (1) (2015) 69–78.
[17] Q. Lin, J. Li, Z. Huang, W. Chen, J. Shen, A short linearly homomorphic proxy signature scheme, IEEE Access (2018). http://dx.doi.org/10.1109/ACCESS.2018.2809684.
[18] Q. Lin, H. Yan, Z. Huang, W. Chen, J. Shen, Y. Tang, An ID-based linearly homomorphic signature scheme and its application in blockchain, IEEE Access (2018). http://dx.doi.org/10.1109/ACCESS.2018.2809426.
[19] J. Li, X. Huang, J. Li, X. Chen, Y. Xiang, Securely outsourcing attribute-based encryption with checkability, IEEE Trans. Parallel Distrib. Syst. 25 (8) (2014) 2201–2210.
[20] 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.
[21] C. Dwork, F. McSherry, K. Nissim, A. Smith, Calibrating noise to sensitivity in private data analysis, in: Theory of Cryptography Conference, Springer, Berlin, Heidelberg, 2006, pp. 265–284.
[22] C. Dwork, M. Naor, T. Pitassi, G.N. Rothblum, Differential privacy under continual observation, in: Proceedings of the Forty-Second ACM Symposium on Theory of Computing, ACM, 2010, pp. 715–724.
[23] C. Dwork, Differential privacy, in: International Colloquium on Automata, Languages, and Programming, 2006, pp. 1–12.
[24] I. Mironov, O. Pandey, O. Reingold, S. Vadhan, Computational differential privacy, in: Advances in Cryptology-CRYPTO 2009, Springer, Berlin, Heidelberg, 2009, pp. 126–142.
[25] Z. Huang, S. Mitra, G. Dullerud, Differentially private iterative synchronous consensus, in: Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society, ACM, 2012, pp. 81–90.
[26] A. Friedman, A. Schuster, Data mining with differential privacy, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2010, pp. 493–502.
[27] M. Abadi, A. Chu, I. Goodfellow, H.B. McMahan, I. Mironov, K. Talwar, L. Zhang, Deep learning with differential privacy, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2016, pp. 308–318.
[28] C. Dimitrakakis, B. Nelson, Z. Zhang, A. Mitrokotsa, B. Rubinstein, Differential privacy for Bayesian inference through posterior sampling, J. Mach. Learn. Res. 18 (11) (2017) 1–39.
[29] T. Li, J. Li, Z. Liu, P. Li, C. Jia, Differentially private naive bayes learning over multiple data sources, Inform. Sci. (2018). http://dx.doi.org/10.1016/j.ins.2018.02.056.
[30] T. Okamoto, S. Uchiyama, A new public-key cryptosystem as secure as factoring, in: International Conference on the Theory and Applications of Cryptographic Techniques, Springer, Berlin, Heidelberg, 1998, pp. 308–318.
[31] T.-Y. Youn, Y.-H. Park, C.H. Kim, J. Lim, An efficient public key cryptosystem with a privacy enhanced double decryption mechanism, in: International Workshop on Selected Areas in Cryptography, Springer, Berlin, Heidelberg, 2005, pp. 144–158.
[32] M. Bellare, P. Rogaway, Random oracles are practical: A paradigm for designing efficient protocols, in: Proceedings of the 1st ACM Conference on Computer and Communications Security, ACM, 1993, pp. 62–73.

Please cite this article in press as: P. Li, et al., Privacy-preserving machine learning with multiple data providers, Future Generation Computer Systems (2018), https://doi.org/10.1016/j.future.2018.04.076.

[33] E. Kiltz, J. Malone-Lee, A general construction of IND-CCA2 secure public key encryption, in: IMA International Conference on Cryptography and Coding, Springer, Berlin, Heidelberg, 2003, pp. 152–166.
[34] E. Fujisaki, T. Okamoto, How to enhance the security of public-key encryption at minimum cost, in: International Workshop on Public Key Cryptography, 1999, pp. 53–68.
[35] E. Bresson, D. Catalano, D. Pointcheval, A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications, in: Advances in Cryptology - ASIACRYPT 2003, International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings, 2003, pp. 37–54.

Ping Li received the M.S. and Ph.D. degree in applied mathematics from Sun Yat-sen University in 2011 and 2016, respectively. She is currently a postdoc under supervisor Jin Li in School of Computer Science and Educational Software, Guangzhou University. Her current research interests include cryptography, privacy-preserving and cloud computing.

Tong Li received his B.S. (2011) and M.S. (2014) from Taiyuan University of Technology and Beijing University of Technology, respectively, both in Computer Science Technology. Currently, he is a Ph.D. candidate at Nankai University. His research interests include applied cryptography and data privacy protection in cloud computing.

Heng Ye received his B.S. (2011) from Beijing Jiaotong university in computer science & technology. Now he is a Ph.D. candidate at Beijing Jiaotong University since 2016. His research interests include differential privacy, attribute based encryption and internet of things.

Jin Li received the B.S. degree in mathematics from Southwest University in 2002 and the Ph.D. degree in information security from Sun Yat-sen University in 2007. Currently, he works at Guangzhou University as a professor. He has been selected as one of science and technology new star in Guangdong province. His research interests include applied cryptography and security in cloud computing. He has published more than 70 research papers in refereed international conferences and journals and has served as the program chair or program committee member in many international conferences.

Xiaofeng Chen (SM16) received the B.S. and M.S. degrees in mathematics from Northwest University, China, in 1998 and 2000, respectively, and the Ph.D. degree in cryptography from Xidian University in 2003. He is currently with Xidian University as a Professor. He has authored or co-authored over 100 research papers in refereed international conferences and journals. His current research interests include applied cryptography and cloud computing security. His work has been cited over 3000 times in Google Scholar. He is on the Editorial Board of Security and Communication Networks, Computing and Informatics, and the International Journal of Embedded Systems. He has served as the Program/General Chair or Program Committee Member in over 30 international conferences.

Yang Xiang (A'08-M'09-SM'12) received the Ph.D. degree in computer science from Deakin University, Melbourne, Australia. He is currently a Full Professor with the School of Information Technology, Deakin University. He is the Director of the Network Security and Computing Lab (NSCLab) and the Associate Head of School (Industry Engagement). He is the Chief Investigator of several projects in network and system security, funded by the Australian Research Council (ARC). He has published more than 150 research papers in many international journals and conferences. Two of his papers were selected as the featured articles in the April 2009 and the July 2013 issues of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS. He has published two books, Software Similarity and Classification (Springer, 2012) and Dynamic and Advanced Data Mining for Progressing Technological Development (IGI-Global, 2009). His research interests include network and system security, distributed systems, and networking. Prof. Xiang has served as the Program/General Chair for many international conferences. He serves as an Associate Editor of the IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, and Security and Communication Networks, and an Editor of the Journal of Network and Computer Applications. He is the Coordinator, Asia, for the IEEE Computer Society Technical Committee on Distributed Processing (TCDP).

Please cite this article in press as: P. Li, et al., Privacy-preserving machine learning with multiple data providers, Future Generation Computer Systems (2018), https://doi.org/10.1016/j.future.2018.04.076.