PDLM: Privacy-Preserving Deep Learning Model on Cloud with Multiple Keys

Xindi Ma, Jianfeng Ma, Hui Li, Qi Jiang, and Sheng Gao

Abstract—Deep learning has aroused a lot of attention and has been used successfully in many domains, such as accurate image recognition and medical diagnosis. Generally, the training of models requires large, representative datasets, which may be collected from a large number of users and contain sensitive information (e.g., users' photos and medical information). The collected data would be stored and computed by service providers (SPs) or delegated to an untrusted cloud. The users can neither control how it will be used, nor realize what will be learned from it, which make the privacy issues prominent and severe. To solve the privacy issues, one of the most popular approaches is to encrypt users' data with their public keys. However, this technique inevitably leads to another challenge that how to train the model based on multi-key encrypted data. In this paper, we propose a novel privacy-preserving deep learning model, namely PDLM, to apply deep learning over the encrypted data under multiple keys. In PDLM, lots of users contribute their encrypted data to SP to learn a specific model. We adopt an effective privacy-preserving calculation toolkit to achieve the training process based on stochastic gradient descent (SGD) in a privacy-preserving manner. We also prove that our PDLM can achieve users' privacy preservation and analyze the efficiency of PDLM in theory. Finally, we conduct an experiment to evaluate PDLM over two real-world datasets and empirical results demonstrate that our PDLM can effectively and efficiently train the model in a privacy-preserving way.

Index Terms—Privacy preservation, deep learning, cryptography, multiple keys

1 Introduction

Recent advances in deep learning has lead to impressive successes in a wide range of applications, such as image recognition, medical diagnosis, and language translation. These advances are partly enabled by the training model with the availability of large and representative datasets [1], which are usually used to discover the hidden valuable information. Based on the accurate training models, service providers (SPs) can provide many new services and applications, including accurate speech recognition [2] and image recognition that outperforms humans [3].

Although the training datasets play an important role in deep learning, the reveal of them would present some serious privacy issues. First, SPs who collect the data keep it forever, and the users can neither control how it will be used, nor realize what will be learned form it. Then, the collected data, such as texts, voices, and images, may also contain some other captured sensitive information: the voices of other people speaking, surrounding noises [4], faces, and computer screens, etc. What's worse, with the volume expansion of the collected data, SP will take more costs to store and compute them. Thus, SP usually migrate them to cloud platforms, which are untrusted though, to reduce the overhead of computational resources, but that will make the privacy issues more prominent and urgent. Because the collected data contains some privacy information, such as images and locations, the cloud can easily track users directly or release their images to other advertisers [5]. As a result, users would be afraid that their sensitive information might be leaked and refuse to contribute it.

However, as we all know that the increase and diversity of training data will make the deep learning models better [6]. Because the data from a small number of users may be very homogeneous, SP may train an overfitted model and get inaccurate results when use it on other inputs. In this case, privacy restrictions would be a huge barrier for deep learning. One way to achieve the security and privacy of the training datasets is to encrypt them with different keys (e.g., each user owns an unique key) and SP uses these multi-key encrypted datasets to train the model. However, achieving secure training over the encrypted data under multiple keys without leaking the privacy of individuals remains a hard problem.

In this paper, to address the aforementioned challenge, we propose a **P**rivacy-preserving **D**eep **L**earning model on cloud with **M**ultiple keys (PDLM). In PDLM, we utilize a public-key cryptosystem with distributed two trapdoors [7] to protect the privacy of the training data and achieve the learning process in a privacy-preserving manner. After that, any sensitive information about the training data will not be revealed and the trained model, including intermediate results during training process, cannot be obtained by other parties than SP. What's more, while achieving the security goals, we also need to obtain an accurate deep learning model with higher efficiency. The main contributions of this paper are as follows:

- *X. Ma, J. Ma, H. Li, and Q. Jiang are with the School of Cyber Engineering, Xidian University, Xi'an, Shaanxi 710071, China. E-mail: {xdma1989, jiangqixdu}@gmail.com, jfma@mail.xidian.edu.cn, hli@xidian.edu.cn.*
- *S. Gao is with the School of Information, Central University of Finance and Economics, Beijing 102202, China. E-mail: sgao@cufe.edu.cn.*

Manuscript received 5 Mar. 2018; revised 15 Aug. 2018; accepted 27 Aug. 2018. Date of publication 5 Sept. 2018; date of current version 5 Aug. 2021. (Corresponding author: Xindi Ma.)
Digital Object Identifier no. 10.1109/TSC.2018.2868750

- We design a novel mechanism, namely PDLM, which allows SP to migrate most computing to the cloud to train a deep learning model without leaking any privacy.
- To reduce the overhead, SP will send the training datasets which are encrypted with users' multiple keys to the untrusted cloud. Then, our PDLM trains the model based on stochastic gradient descent (SGD) in cloud and performs the feed-forward and back-propagation procedure based on an efficient privacy-preserving calculation toolkit. In this way, the storage and computational overhead at SP is minimized while the training data is not leaked to SP and the untrusted cloud.
- While receiving the multi-key encrypted datasets, the cloud server transforms these training datasets into encryptions under the product of all involved public keys. After that, with these transformed ciphertexts (under the same key), we can run the traditional arithmetic operations to learn the model parameters in the privacy-preserving manner.
- We conduct the analysis of PDLM in both theory and practice. Both theoretical analysis and experimental results over a real-world dataset show that PDLM can train the deep learning model efficiently and effectively.

The rest of this paper is organized as follows. Section 2 gives some related work. In Section 3, we present the system overview and problem formulation, followed by the details of PDLM in Section 4. Section 5 presents the theoretical analysis of privacy and efficiency. In Section 6, we empirically test the accuracy of the trained model and the computational costs of PDLM. Finally, we conclude this paper in Section 7.

2 Related Work

In recent years, privacy preservation has been gained significant interest. A number of approaches have been proposed to address location privacy [8], [9], [10], [11], identity privacy [12], [13], and social privacy in social network [14], [15], [16]. Simultaneously, there are many existing works which have been proposed for the privacy preservation in data mining and knowledge discovery. Based on the privacy-preserving mechanism, we divide them into two categories: cryptographic mechanism and data perturbation mechanism. In the former one, users usually encrypt their data to prevent the disclosure of privacy, which was proposed by Lindell and Pinkas [17]. Then, in the latter mechanism, the private data usually is preserved by adding sanitized noise, which was proposed by Agrawal and Srikant [18].

2.1 Cryptographic Mechanism

Based on the cryptographic mechanism, secure multi-party computation (SMC) which can protect the intermediate results when multiple parties perform the learning has been widely used in machine learning, such as learning linear regression functions [19], decision trees [17], Naive Bayes classifiers [20], and so on. To protect the privacy in reinforcement learning (RL), Miyajima et al. [21] proposed learning methods with SMC for Q-learning which is one of the typical methods for RL. Mohassel and Zhang [22] also presented new and efficient protocols based on secure two-party computation to protect the privacy in machine learning for linear regression, logistic regression and neural network training. Their protocols supported secure arithmetic operations on shared decimal numbers and non-linear functions such as sigmoid and softmax. Based on additively homomorphic encryption, Wang et al. [23] proposed a novel privacy-preserving scheme for canonical correlation analysis (CCA) and encrypted the private data by randomly splitting numerical, formalize CCA problem and then reduce it to a symmetric eigenvalue problem. To overcome the high computational cost of homomorphic encryption in high-dimensional classifiers, Yonetani et al. [24] proposed their privacy-preserving mechanism based on doubly homomorphic encryption which supported multi-party secure scalar product. However, earlier homomorphic encryption schemes only support single operation—either addition or multiplication. Zhang et al. [25] and Yuan et al. [26] proposed two mechanisms based on BGV fully homomorphic encryption and doubly homomorphic encryption to encrypt the private data respectively and employed cloud servers to perform the high-order back-propagation algorithm on the encrypted data efficiently for deep computation model training. Considering the mechanism in [6] that local data information may be leaked to an honest-but-curious server, Phong et al. [27] fixed that by building an enhanced system based on additively homomorphic encryption. Hesamifard et al. [28] also provided a theoretical foundation for implementing deep neural network algorithms in encrypted domain and developed techniques to adopt neural networks within practical limitations of current homomorphic encryption schemes. Although the above works protect the private data well, they did not consider the condition which the private data is encrypted by multiple keys from different data owners. As a directly related work, Li et al. [29] proposed two schemes based on multi-key fully homomorphic encryption (MK-FHE), which were able to preserve the privacy of sensitive data, intermediate results as well as the training model. But these fully homomorphic encryption (FHE) schemes usually have a low efficiency and we adopt a more efficient privacy-preserving calculation toolkit with multiple keys to achieve the same goal.

2.2 Data Perturbation Mechanism

As the mainstream of data perturbation mechanism, differential privacy [30] has also been widely used in privacy-preserving deep learning. Zhang and Zhu [31] focused on a class of regularized empirical risk minimization machine learning problems, and developed two methods to provide differential privacy to distributed learning algorithms over a network. Phan et al. [32] also proposed a novel mechanism to preserve differential privacy in deep neural networks, which privacy budget consumption was totally independent of the number of training steps and the noise could be injected into features based on the contribution of each to the output. To address the privacy preservation in distributed multi-task learning (MTL), Xie et al. [33] proposed an asynchronous proximal gradient algorithm to solve a general class of MTL formulations, which was robust against

TABLE 1
Definitions and Notations in PDLM

Symbol	Definition
pk_i	data owner $i's$ public key
sk_i	data owner $i's$ weak private key
pk_Σ	the union public key of data owners
PD_{SK_i}	the partial decryption algorithm with partial strong private key $SK_i, i \in \{1,2\}$
$\lambda, \lambda_1, \lambda_2$	strong private key and partial strong private key
$[X^{(k)}]_{pk_h}, [X^{(k)}]_{pk_\Sigma}$	the inputs of kth layer which are encrypted with pk_h or pk_Σ
$[W^{(k)}]_{pk_\Sigma}, [b^{(k)}]_{pk_\Sigma}$	the encrypted parameters in kth layer
$[\overline{Y}]_{pk_\Sigma}, [Y]_{pk_\Sigma}$	the encrypted model outputs and true values
τ	the error threshold in back-propagation deep learning

network delays and provided a guaranteed differential privacy through carefully designed perturbation. Shokri and Shmatikov [6] proposed a system which let data owners train independently on their own datasets and selectively shared small subsets of their models' key parameters which are also injected into noises. Abadi et al. [1] and Hamm et al. [34] considered that the training data were crowd-sourced and proposed some new techniques for learning problems with differential privacy guaranteed. Focusing on the privacy-preserving feature selection in data mining, Li et al. [35] proposed a local learning-based feature weighting framework which use objective perturbation and output perturbation strategies to produce feature selection algorithms with privacy preservation. However, the above data perturbation mechanisms are inherently flawed. All of them injected noise into training data or model parameters and that would result in a less accurate model.

3 System Overview and Problem Formulation

As discussed above, more diverse data can train a better learning model. However, directly collecting data from users and training the deep learning model over cloud may invoke the unexpected privacy issues, which is a key hinder for the data owners to contribute training data. To this end, we design PDLM based on an efficient privacy-preserving calculation toolkit with multiple keys [7] to help service provider train a global deep learning model over semi-honest cloud environment. In this section, we first present some preliminaries that serve as the basis of PDLM, and then present the system model, threat model, and security model for PDLM. For your convenience, the notations used in the sequel are listed in Table 1.

3.1 Preliminaries

3.1.1 Deep Learning

Deep learning can be seen as a multi-layer neural network. The input layer is composed of the raw features extracted from the input data. Then, there are also several hidden layers, which extract more abstract features based on the inputs. The outputs of last layer correspond to abstract answers produced by the model. The neurons are connected via weights that determine the contribution of each input signal. In a typical multi-layer network, each neuron receives the output of the neuron in the previous layer plus a bias signal from a special neuron. In a deep learning structure of neural network, there can be multiply layers each with thousands of neurons.

Each neuron node (except the bias node) is associated with an activation function f. Examples of f in deep learning are $f(x) = max\{0, x\}$ (rectified linear), $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ (hyperbolic tangent), and $f(x) = (1 + e^{-x})^{-1}$ (sigmoid). The outputs at layer $l+1$, denoted as $a^{(l+1)}$, are computed as $a^{(l+1)} = f(W^{(l)}a^{(l)} + b^{(l)})$, in which $(W^{(l)}, b^{(l)})$ are the parameters, including weights and bias, connecting layers l and $l+1$ and $a^{(l)}$ are the outputs at layer l. So the features that are extracted at each layer are determined by the activation function f, weight variables W, and bias terms b. One of the primary goals in deep learning is to automatically learn the values of parameters (weights and bias) from training data to maximize the objective of the neural network (e.g., classification accuracy).

However, learning the parameters is always a nonlinear optimization problem and some variants of gradient descent [36] are usually used to solve this problem. Typically, if learning on a large dataset, stochastic gradient descent is a good choice to solve the optimization problem, which computes the gradient over a mini-batch dataset (i.e., a subset of the whole dataset) [37]. During the training process, the parameters are usually learned through feed-forward and back-propagation procedures. Given the input data, feed forward can compute the outputs of the network and then obtain the error between the outputs and the true values. After that, this error will be propagated back through the network using back propagation and the contribution of each neuron to the error will also be computed. Finally, we can compute the gradients of each parameter from neurons' activation values and their contributions to error. Algorithms 1 and 2 describe the processes of feed forward and back propagation during the learning, in which we assume that the network owns 1 hidden layer, input layer owns n_1 nodes, hidden layer owns n_2 nodes, output layer owns n_3 nodes, and L samples are randomly extracted for SGD each time.

Algorithm 1. Process of Feed Forward During Learning

Input: L samples randomly extracted from input data $\{X^{(1)}, Y\}$, initialized parameters $\{W^{(1)}, W^{(2)}\}$, activation function $f(x)$.
Output: the error between outputs and true values.

for sample $h = 1, \ldots, L$ **do**
  for $k = 1, 2$ **do**
    for $j = 1, \ldots, n_{k+1}$ **do**
      $v_{hj}^{(k+1)} = \sum_{i=1}^{n_k} x_{hi}^{(k)} \cdot w_{ij}^{(k)} + b_j^{(k)}$;
      $x_{hj}^{(k+1)} = f(v_{hj}^{(k+1)})$;
    end for
  end for
  $\overline{Y} = X^{(3)}$;
end for
$error = \frac{1}{2L} \sum_{h=1}^{L} \sum_{j=1}^{n_3} (\overline{y}_{hj} - y_{hj})^2$;

3.1.2 Distributed Two Trapdoors Public-Key Cryptosystem

In order to realize PDLM, the public-key cryptosystem with distributed two trapdoors (DT-PKC) proposed by Liu

et al. [7] could be a suitable solution for multi-key management. Although there are many other mechanisms [38], [39] that support multi-key outsourcing computation, they either do not support complex computations or can not decrypt the trained model for SP. Based on Paillier [40] and Bresson et al.'s cryptosystem [41], the DT-PKC works as follows:

KeyGen. Given a security parameter k and two large prime numbers p, q, where $|p| = |q| = k$. Because of the property of strong primes [7], [41], two strong primes p', q', s.t., $p' = \frac{p-1}{2}$ and $q' = \frac{q-1}{2}$, can be obtained. Then, compute $\lambda = lcm(p-1, q-1) = 2p'q'$ and $N = pq$. After that, we define a function $L(x) = \frac{x-1}{N}$ and select a generator g of order $2p'q'$. Randomly choose $\theta_i \in [1, N/4]$ and calculate $h_i = g^{\theta_i} \bmod N^2$ for user i. So user i's public key is $pk_i = \{N, g, h_i\}$ and weak private key is $sk_i = \theta_i$. The strong key for system is $SK = \lambda$.

Algorithm 2. Process of Back Propagation During Learning

Input: L samples randomly extracted from input data $\{X^{(1)}, Y\}$, learning rate η, initialized parameters $\{W^{(1)}, W^{(2)}, b^{(1)}, b^{(2)}\}$, activation function $f(x)$, maximum iteration ϕ, error threshold τ, and computed error $error$ in feed forward.

Output: the values of parameters $\{W^{(1)}, W^{(2)}, b^{(1)}, b^{(2)}\}$.

if $error > \tau$ **then**
  for sample $h = 1, \ldots, L$ **do**
    for $i = 1, \ldots, n_3$ **do**
      $\delta_{hi}^{(3)} = (\overline{y}_{hi} - y_{hi}) \cdot f'(v_{hi}^{(3)})$;
      $\Delta b_i^{(2)} = \Delta b_i^{(2)} + \delta_{hi}^{(3)}$;
    end for
    for $i = 1, \ldots, n_2$ **do**
      $\delta_{hi}^{(2)} = f'(v_{hi}^{(2)}) \cdot \sum_{j=1}^{n_3} (\delta_{hj}^{(3)} \cdot w_{ij}^{(2)})$;
      $\Delta b_i^{(1)} = \Delta b_i^{(1)} + \delta_{hi}^{(2)}$;
    end for
  end for
  for $k = 1, 2$ **do**
    for $j = 1, \ldots, n_{k+1}$ **do**
      for $i = 1, \ldots, n_k$ **do**
        for $h = 1, \ldots, L$ **do**
          $\Delta w_{ij}^{(k)} = \Delta w_{ij}^{(k)} + x_{hi}^{(k)} \cdot \delta_{hj}^{(k+1)}$;
        end for
        $w_{ij}^{(k)} = w_{ij}^{(k)} - \eta \cdot \frac{1}{L} \cdot \Delta w_{ij}^{(k)}$;
      end for
      $b_j^{(k)} = b_j^{(k)} - \eta \cdot \frac{1}{L} \cdot \Delta b_j^{(k)}$;
    end for
  end for
else
  break;
end if

Encryption (Enc). Select a random number $r \in [1, N/4]$, the ciphertext under pk_i for plaintext $m \in \mathbb{Z}_N$ can be generated as $[m]_{pk_i} = \{T_{i,1}, T_{i,2}\}$, where $T_{i,1} = g^{r\theta_i}(1 + mN) \bmod N^2$ and $T_{i,2} = g^r \bmod N^2$.

Decryption with Weak Private Key (WDec). Given ciphertext $[m]_{pk_i}$, we can decrypt it with weak private key as following:

$$m = L\left(\frac{T_{i,1}}{T_{i,2}^{\theta_i}} \bmod N^2\right).$$

Decryption with Strong Private Key (SDec). Given any ciphertext $[m]_{pk_i}$, we can decrypt it with strong private key as following:

$$m = L(T_{i,1}^{\lambda} \bmod N^2)\lambda^{-1} \bmod N = L(1 + mN\lambda)\lambda^{-1} \bmod N.$$

To achieve the DT-PKC, we can randomly split the strong private key into two parts which are denoted as $SK_j = \lambda_j (j = 1, 2)$, s.t., $\lambda_1 + \lambda_2 \equiv 0 \bmod \lambda$ and $\lambda_1 + \lambda_2 \equiv 1 \bmod N^2$ hold at the same time.

Partial Decryption with SK_1 (PSDec1). While receiving $[m]_{pk_i} = \{T_{i,1}, T_{i,2}\}$, we can partially decrypt it using algorithm $PD_{SK_1}(\cdot)$ with $SK_1 = \lambda_1$ as following:

$$ST_{i_1} = (T_{i,1})^{\lambda_1} = g^{r\theta_i\lambda_1}(1 + mN\lambda_1) \bmod N^2.$$

Partial Decryption with SK_2 (PSDec2). While receiving ST_{i_1} and $[m]_{pk_i}$, we can obtain plaintext m using algorithm $PD_{SK_2}(\cdot, \cdot)$ as following:

$$\begin{aligned} ST_{i_2} &= (T_{i,1})^{\lambda_2} = g^{r\theta_i\lambda_2}(1 + mN\lambda_2) \bmod N^2 \\ L(ST_{i_1} \cdot ST_{i_2}) &= L(g^{r\theta_i\lambda_1}(1 + mN\lambda_1) \cdot g^{r\theta_i\lambda_2}(1 + mN\lambda_2) \bmod N^2) \\ &= m(\lambda_1 + \lambda_2) \bmod N^2 \\ &= m, \end{aligned}$$

where $\lambda_1 + \lambda_2 \equiv 0 \bmod \lambda$, $g^{\lambda} \equiv 1 \bmod N^2$, and $\lambda_1 + \lambda_2 \equiv 1 \bmod N^2$.

Note that for two given ciphertexts $[m_1]_{pk_i}$ and $[m_2]_{pk_i}$ under the same public key pk_i, they have the following properties:

1) the product of two ciphertexts will decrypt to the sum of their corresponding plaintexts: $[m_1]_{pk_i} \cdot [m_2]_{pk_i} = \{T_{i,1}^1 \cdot T_{i,1}^2, T_{i,2}^1 \cdot T_{i,2}^2\} = \{g^{\theta_i(r_1+r_2)}(1 + (m_1 + m_2)N) \bmod N^2, g^{r_1+r_2} \bmod N^2\} = [m_1 + m_2]_{pk_i}$.
2) given constant number $a \in \mathbb{Z}_N$ and ciphertext $[m_1]_{pk_i}$, it has $([m_1]_{pk_i})^{N-a} = \{g^{r(N-a)\theta_i}(1 + (N-a)m_1N) \bmod N^2, g^{r(N-a)} \bmod N^2\} = \{g^{r(N-a)\theta_i}(1 - a * m_1N) \bmod N^2, g^{r(N-a)} \bmod N^2\} = [-a * m_1]_{pk_i}$.

3.2 System Model

In a deep learning system, each data owner contributes his/her sensitive local training data. To guarantee the privacy of such training data during the learning process, all the privacy information (i.e., local training data, intermediate results, model's outputs, etc.) should be kept and computed in ciphertext. In this manner, we can derive the basic entities for our privacy-preserving deep learning system as follows (see in Fig. 1).

1) *Key Generation Center (KGC).* KGC is a indispensable entity which distribute and manage all the public and private keys in system. It is trusted by all entities.
2) *Data Owners (DOs).* DOs request service from service provider and contribute local data for service provider to train the deep learning model. Before uploading the training data, DOs will encrypt the data with their corresponding public keys. We assume that all DOs' datasets have similar content and distribution.

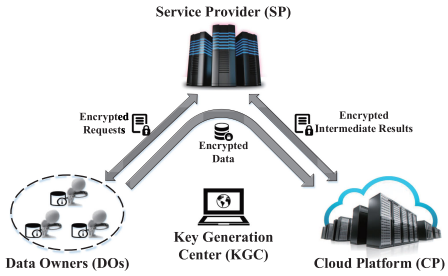


Fig. 1. System framework of PDLM.

3) *Service Provider (SP).* As a service provider, SP is responsible for collecting data from DOs and training model to provide new service. However, SP usually possesses limited storage and computational resources, and thus it sends the encrypted training data to cloud for storage and computation. Specially, SP owns the partial strong private key $SK_2 = \lambda_2$ and DOs' union public key pk_Σ.[1]
4) *Cloud Platform (CP).* CP stores and manages all the training data in the learning model. In addition, CP is able to partially decrypt ciphertexts sent by SP and performs certain calculations on ciphertexts. So CP owns another partial strong private key $SK_1 = \lambda_1$.

3.3 Threat Model

In PDLM, we consider the KGC to be a trusted entity, which generates the public and private keys for the system. On the other hand, SP and CP are *curious-but-honest (non-colluding)* adversaries, which strictly follow the protocol. But they are also interested to gather or learn the privacy information during the training process. Based on that, we introduce an active adversary $\mathcal{A}^*$ in our model. The goal of $\mathcal{A}^*$ is to decrypt the challenge DOs' original ciphertext and the challenge SP's encrypted model parameters with the following capabilities:

1) $\mathcal{A}^*$ could *eavesdrop* all communications to obtain the encrypted data and launch an *active attack* to intercept, tamper, and forge the transmitted messages.
2) $\mathcal{A}^*$ could *compromise* CP to guess the plaintext value of all ciphertexts outsourced from DOs, and all ciphertexts sent from SP by executing the training process.
3) $\mathcal{A}^*$ could *compromise* SP to guess the plaintext value of all ciphertexts sent from DOs and CP by executing the training process.
4) $\mathcal{A}^*$ could *compromise* one or more DOs, with the exception of challenge DOs, to obtain access to their decryption abilities, and guess all ciphertexts belonging to the challenge DOs.

However, the adversary $\mathcal{A}^*$ is not allowed to compromise: (1) both CP and SP concurrently, (2) the challenge DOs. We remark that such restrictions are typical in cryptographic protocols [42], [43].

3.4 Security Model

The security model adopted in this work is similar to that in [44], [45], [46], which is defined in secure two-party protocols for non-colluding semi-honest adversaries and widely used to prove the security of multi-party protocols. Consider two parties: SP and CP, we construct four simulators (S^1_{SP}, S^1_{CP}, S^2_{SP}, S^2_{CP}) for two phases against two types of attackers ($\mathcal{A}_{SP}$, $\mathcal{A}_{CP}$) that corrupt SP and CP, respectively. These attackers are deemed as non-colluding and semi-honest. Due to the length limitation, please refer to [7], [44], [45], [46] for the general security model definitions.

1. The union public key is constructed as $pk_\Sigma = (N, g, h_\Sigma = g^{\sum_{j=1,\ldots,L} \theta_j})$ which associates with DO $j (j = 1, \ldots, L)$.

4 Construction of PDLM

In this section, we present the details of PDLM based on DT-PKC. When the system is initialized, each DO registers as a data owner and KGC will negotiate a one-to-one key with him. After generating the public and private keys, KGC uses the negotiatory key to encrypt DO's private key and sends the encrypted one to him. Then, the secure socket layer (SSL) or transport layer security (TLS) protocols are made use to secure all communications between DOs and SP, CP and SP. The SSL/TLS protocols aim primarily to guarantee the data integrity and authenticity between two communicating entities. Before training the model, SP collects the encrypted training datasets from DOs and then uploads them to CP. After that, SP will train the model with the help of CP.

In PDLM, we select the usually used sigmoid function $f(x) = (1 + e^{-x})^{-1}$ as the activation function (other activation functions can also be used after the similar conversion). Because the DT-PKC does not support the exponentiation operation required by the sigmoid function, we use Taylor series to approximate the results based on Taylor theorem [47] instead of using fully homomorphic encryption or other complex computing protocols, which is defined as follow:

$$\begin{aligned} f(x) = \frac{1}{1+e^x} &= f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 \\ &\quad + \frac{f^{(3)}(0)}{3!}x^3 + \cdots + \frac{f^{(k)}(0)}{k!}x^k + h_k(x)x^k \\ &= \frac{1}{2} + \frac{x}{4} - \frac{x^3}{48} + o(x^4) \\ &\approx 0.5 + 0.25 \cdot x - 0.02 \cdot x^3. \end{aligned}$$

According to the property of Taylor series, we can decide the number of terms in the extending according to our accuracy requirement. To simplify the description of privacy-preserving calculation, we first give the procedure to calculate $[x^2]_{pk_\Sigma}$ and $[x^3]_{pk_\Sigma}$ based on DT-PKC while CP owns ciphertext $[x]_{pk_i}$. Note that the private data and parameters which are needed to encrypt are integers; therefore, we restrict the data and parameters to be in the range of $[0, \Omega]$, where $|\Omega| < |N|/8$. Although some of the data are floats, we can still calculate the results by the method in [7]. It should be noted that CP and SP are curious-but-honest and they are non-colluding adversaries. Based on the double-trapdoor cryptosystem, they can transform the multi-key encrypted datasets into the same-key encrypted ciphertexts and then perform complex arithmetic operations to train the model parameters.

4.1 Secure Multiplication Protocol (SMP)

Since CP can only partially decrypt the ciphertext $[x]_{pk_i}$ which is encrypted by DO i, it must ask SP for help to

calculate $[x^2]_{pk_\Sigma}$ and $[x^3]_{pk_\Sigma}$ cooperatively. Given ciphertext $[x]_{pk_i}$, we design secure multiplication protocol to calculate $[x^2]_{pk_\Sigma}$ and $[x^3]_{pk_\Sigma}$, which is described as follows:

Step-I(@CP). CP selects a random number $r_x \in \mathbb{Z}_N$, calculates:

$$\begin{aligned} pk_{\Sigma_{j\neq i}} &= \{N, g, h_{\Sigma_{j\neq i}}\} = \{N, g, g^{\Sigma_{j=1,\ldots,L, j\neq i}\theta_j}\} \\ [x]_{pk_\Sigma} &= \{T_{i,1} * h_{\Sigma_{j\neq i}}, T_{i,2}\} \\ &= \{g^{r\Sigma_{j=1,\ldots,L}\theta_j}(1+mN) \bmod N^2, g^r \bmod N^2\} \\ x' &= [x]_{pk_i} \cdot [r_x]_{pk_i} = [x + r_x]_{pk_i}. \end{aligned}$$

Then, CP calculates $x'_1 = PD_{SK_1}(x')$, and sends $\{x', x'_1\}$ to SP.

Step-II(@SP). While receiving the partially decrypted information, SP calculates as follows:

$$\begin{aligned} m_1 &= (PD_{SK_2}(x', x'_1))^2 = (x + r_x)^2 \\ m_2 &= (PD_{SK_2}(x', x'_1))^3 = (x + r_x)^3. \end{aligned}$$

After that, SP encrypts $\{m_1, m_2\}$ with DOs' union public key pk_Σ and sends the ciphertexts $\{t_1 = [m_1]_{pk_\Sigma}, t_2 = [m_2]_{pk_\Sigma}\}$ to CP.

Step-III(@CP). Once $\{t_1, t_2\}$ are received, CP computes $s_1 = ([r_x^2]_{pk_\Sigma})^{N-1}, s_2 = ([x]_{pk_\Sigma})^{N-2r_x}, s_3 = (t_1)^{N-r_x}, s_4 = ([x]_{pk_\Sigma})^{N-r_x^2}$, and calculates the following to obtain the encrypted x^2 and x^3:

$$\begin{aligned} s_1 &= ([r_x^2]_{pk_\Sigma})^{N-1} = [-r_x^2]_{pk_\Sigma} \\ s_2 &= ([x]_{pk_\Sigma})^{N-2r_x} = [-2r_x x]_{pk_\Sigma} \\ s_3 &= (t_1)^{N-r_x} = [-r_x * m_1]_{pk_\Sigma} \\ s_4 &= ([x]_{pk_\Sigma})^{N-r_x^2} = [-r_x^2 * x]_{pk_\Sigma} \\ t_3 &= t_1 \cdot s_1 \cdot s_2 = [x^2]_{pk_\Sigma} \\ t_4 &= t_2 \cdot s_3 \cdot s_4 \cdot (t_3)^{N-2r_x} = [x^3]_{pk_\Sigma}. \end{aligned}$$

While giving different ciphertexts $\{[x_1]_{pk_i}, [x_2]_{pk_i}\}$ which are encrypted with the same public key, we can also obtain the encrypted product $[x_1 \cdot x_2]_{pk_\Sigma}$ using the above secure multiplication protocol.

4.2 Privacy-Preserving Feed Forward

In this phase, CP computes the output of the network and obtains the error between the outputs and the true values in a privacy-preserving manner. Since the training data existed in CP is collected from different DOs, CP has to perform the calculations of Algorithm 1 on multi-key encrypted ciphertexts. Specially, CP owns encrypted initialized parameters $\{[W^{(k)}]_{pk_\Sigma}, [b^{(k)}]_{pk_\Sigma}, k \in \{1, 2\}\}$ which are encrypted with DOs' union public key and multi-key encrypted training data $\{[X^{(1)}]_{pk_h}, [Y]_{pk_h}, h \in [1, L]\}$ which we assume that CP randomly extracts L samples each time and each sample is encrypted with a different public key.

Step-I(@CP). For $[x_{hi}^{(k)}]_{pk_h} \in [X^{(k)}]_{pk_h}, [w_{ij}^{(k)}]_{pk_\Sigma} \in [W^{(k)}]_{pk_\Sigma}, h \in [1, L], i \in [1, n_k], j \in [1, n_{k+1}], k = 1$, CP generates three random matrixes $R_x \in \mathbb{Z}_N^{L*n_k}, R_w \in \mathbb{Z}_N^{n_k*n_{k+1}}, R' \in \mathbb{Z}_N^{L*n_k*n_{k+1}}$ and calculates as follows, $r_{hi}^x \in R_x, r_{ij}^w \in R_w, r'_{hij} \in R'$:

$$\begin{aligned} x_{hi}^{\prime(k)} &= [x_{hi}^{(k)}]_{pk_h} \cdot [r_{hi}^x]_{pk_h} = [x_{hi}^{(k)} + r_{hi}^x]_{pk_h} \\ w_{ij}^{\prime(k)} &= [w_{ij}^{(k)}]_{pk_\Sigma} \cdot [r_{ij}^w]_{pk_\Sigma} = [w_{ij}^{(k)} + r_{ij}^w]_{pk_\Sigma} \\ t_{hij} &= [r'_{hij}]_{pk_h} \cdot ([x_{hi}^{(k)}]_{pk_h})^{N-r_{ij}^w} = [r'_{hij} - r_{ij}^w \cdot x_{hi}^{(k)}]_{pk_h}. \end{aligned}$$

Then, CP calculates $x_{hi}^{\prime\prime(k)} = PD_{SK_1}(x_{hi}^{\prime(k)}), w_{ij}^{\prime\prime(k)} = PD_{SK_1}(w_{ij}^{\prime(k)})$, $t'_{hij} = PD_{SK_1}(t_{hij})$ and represents the matrixes as $\{X^{\prime\prime(k)}, W^{\prime\prime(k)}, T'\}$. Finally, CP sends $\{X^{\prime\prime(k)}, X^{\prime(k)}, W^{\prime\prime(k)}, W^{\prime(k)}, T', T\}$ to SP.

Step-II(@SP). Using another partial strong private key, SP calculates as follows, $x_{hi}^{\prime\prime(k)} \in X^{\prime\prime(k)}, x_{hi}^{\prime(k)} \in X^{\prime(k)}, w_{ij}^{\prime\prime(k)} \in W^{\prime\prime(k)}$, $w_{ij}^{\prime(k)} \in W^{\prime(k)}, t'_{hij} \in T', t_{hij} \in T, h \in [1, L], i \in [1, n_k], j \in [1, n_{k+1}]$, $k = 1$:

$$\begin{aligned} mx_{hij}^{(k)} &= PD_{SK_2}(x_{hi}^{\prime\prime(k)}, x_{hi}^{\prime(k)}) \cdot PD_{SK_2}(w_{ij}^{\prime\prime(k)}, w_{ij}^{\prime(k)}) \\ &= (x_{hi}^{(k)} + r_{hi}^x)(w_{ij}^{(k)} + r_{ij}^w) \\ t''_{hij} &= [PD_{SK_2}(t'_{hij}, t_{hij})]_{pk_\Sigma} = [r'_{hij} - r_{ij}^w \cdot x_{hi}^{(k)}]_{pk_\Sigma}. \end{aligned}$$

SP then encrypts each $mx_{hij}^{(k)}$ with union public key of DOs and represents all the ciphertexts as matrix $[MX^{(k)}]_{pk_\Sigma}$. After that, SP sends matrixes $\{[MX^{(k)}]_{pk_\Sigma}, T''\}$ to CP.

Step-III(@CP). While receiving the ciphertexts from SP, CP calculates $s_1 = ([w_{ij}^{(k)}]_{pk_\Sigma})^{N-r_{hj}^x}, s_2 = ([r'_{hij}]_{pk_\Sigma})^{N-1}, s_3 = ([r_{hi}^x \cdot r_{ij}^w]_{pk_\Sigma})^{N-1}$ and obtains the results as follows, $[mx_{hij}^{(k)}]_{pk_\Sigma} \in [MX^{(k)}]_{pk_\Sigma}, t''_{hij} \in T'', r_{hi}^x \in R_x, r_{ij}^w \in R_w, r'_{hij} \in R'$, $[w_{ij}^{(k)}]_{pk_\Sigma} \in [W^{(k)}]_{pk_\Sigma}$:

$$\begin{aligned} cx_{hij}^{(k)} &= \left[mx_{hij}^{(k)}\right]_{pk_\Sigma} \cdot t''_{hij} \cdot s_1 \cdot s_2 \cdot s_3 = \left[x_{hi}^{(k)} \cdot w_{ij}^{(k)}\right]_{pk_\Sigma} \\ cv_{hj}^{(k+1)} &= \left(\prod_{i=1}^{n_k} cx_{hij}^{(k)}\right) \cdot \left[b_j^{(k)}\right]_{pk_\Sigma} = \left[v_{hj}^{(k+1)}\right]_{pk_\Sigma} \\ \left[(v_{hj}^{(k+1)})^3\right]_{pk_\Sigma} &= SMP\left(cv_{hj}^{(k+1)}\right) \\ cx_{hj}^{(k+1)} &= [0.5]_{pk_\Sigma} \cdot \left[\left(v_{hj}^{(k+1)}\right)\right]_{pk_\Sigma}^{0.25} \cdot \left[\left(v_{hj}^{(k+1)}\right)^3\right]_{pk_\Sigma}^{N-0.02} \\ &= \left[0.5 + 0.25\left(v_{hj}^{(k+1)}\right) - 0.02\left(v_{hj}^{(k+1)}\right)^3\right]_{pk_\Sigma} \\ &\approx \left[x_{hj}^{(k+1)}\right]_{pk_\Sigma}. \end{aligned}$$

Specially, SMP(·) is the secure multiplication protocol and we can obtain the encrypted cube of information by that. The matrix of calculated results can be represented as $CX^{(k+1)}$. When $k = 2$, the processing procedure is similar with above calculations, so we do not repeat the description here again and we can obtain the encrypted output of the network as $[\overline{Y}]_{pk_\Sigma} = CX^{(3)}$. Then, CP selects two random matrixes $R_a, R_b \in \mathbb{Z}_N^{L*n_3}$ and calculates the error in the following, $[\overline{y}_{hj}]_{pk_\Sigma} \in [\overline{Y}]_{pk_\Sigma}, [y_{hj}]_{pk_h} \in [Y]_{pk_h}, r_{hj}^a \in R_a, r_{hj}^b \in R_b, h \in [1, L], j \in [1, n_3]$:

$$\begin{aligned} \overline{cy}_{hj} &= [\overline{y}_{hj}]_{pk_\Sigma} \cdot [r_{hj}^a]_{pk_\Sigma} = [\overline{y}_{hj} + r_{hj}^a]_{pk_\Sigma} \\ cy_{hj} &= [y_{hj}]_{pk_h} \cdot [r_{hj}^b]_{pk_h} = [y_{hj} + r_{hj}^b]_{pk_h}. \end{aligned}$$

CP also calculates $\overline{cy}'_{hj} = PD_{SK_1}(\overline{cy}_{hj}), cy'_{hj} = PD_{SK_1}(cy_{hj})$, and represents the matrixes as $\{\overline{CY}', \overline{CY}, CY', CY\}$ which are also sent to SP.

Step-IV(@SP). Once receiving the information from CP, SP calculates as follow:

$$\begin{aligned} re_{hj} &= [PD_{SK_2}(\overline{cy}'_{hj}, \overline{cy}_{hj}) - PD_{SK_2}(cy'_{hj}, cy_{hj})]_{pk_\Sigma} \\ &= [\overline{y}_{hj} - y_{hj} + r^a_{hj} - r^b_{hj}]_{pk_\Sigma}. \end{aligned}$$

Then, SP represents the matrix as RE and sends that to CP for remaining calculations.

Step-V(@CP). After receiving matrix RE, CP will calculate $s_4 = [r^b_{hj} - r^a_{hj}]_{pk_\Sigma}$ and obtain the error as follows, $h \in [1, L], j \in [1, n_3]$:

$$\begin{aligned} ce_{hj} &= [e_{hj}]_{pk_\Sigma} = re_{hj} \cdot s_4 = [\overline{y}_{hj} - y_{hj}]_{pk_\Sigma} \\ [e^2_{hj}]_{pk_\Sigma} &= SMP(ce_{hj}) \\ [error]_{pk_\Sigma} &= \left(\prod_{h=1}^{L} \prod_{j=1}^{n_3} [e^2_{hj}]_{pk_\Sigma} \right)^{1/2L}. \end{aligned}$$

After obtaining the encrypted error, CP will compare it with the threshold τ. And if it is greater than τ, CP will perform the privacy-preserving back propagation in following phase.

4.3 Privacy-Preserving Back Propagation

In this phase, CP first securely compares the encrypted error $[error]_{pk_\Sigma}$ with error threshold τ and then performs the back propagation of Algorithm 2 in the privacy-preserving manner.

Step-I(@CP). Given $[error]_{pk_\Sigma}$ and error threshold τ, CP encrypts τ with union public key and calculates as follows:

$$\begin{aligned} c\tau' &= ([\tau]_{pk_\Sigma})^2 \cdot [1]_{pk_\Sigma} = [2\tau + 1]_{pk_\Sigma} \\ ce' &= ([error]_{pk_\Sigma})^2 = [2 \cdot error]_{pk_\Sigma}. \end{aligned}$$

Then, CP flips a coin ξ randomly. If $\xi = 1$, CP calculates $[\beta]_{pk_\Sigma} = c\tau' \cdot (ce')^{N-1}$, else $[\beta]_{pk_\Sigma} = ce' \cdot (c\tau')^{N-1}$. CP also selects another random number r, s.t., $|r| < |N|/4$, and calculates $[\beta']_{pk_\Sigma} = ([\beta]_{pk_\Sigma})^r$. After that, CP partially decrypts $[\beta']_{pk_\Sigma}$ as $\beta'' = PD_{SK_1}([\beta']_{pk_\Sigma})$ and sends the result to SP.

Step-II(@SP). Using partial strong private key SK_2, SP decrypts β'' and obtains β'. If $|\beta'| > |N|/2$, SP denotes $u' = 1$, else $u' = 0$. Then, SP sends u' to CP.

Step-III(@CP). While receiving u' from SP, CP computes as follows:

- If $\xi = 1$, $u'' = u'$, else $u'' = 1 - u'$.
- If $u'' = 1$, $error > \tau$, else $error \leq \tau$.

If $error > \tau$, CP will continue to calculate in the following steps.

Step-IV(@CP). Since $f(x) = (1 + e^{-x})^{-1}$ and $x^{(k+1)}_{hi} = f(v^{(k+1)}_{hi}), i \in [1, n_{k+1}], h \in [1, L], k \in \{1, 2\}$, we can calculate $f'(v^{(k+1)}_{hi})$ as $f'(v^{(k+1)}_{hi}) = x^{(k+1)}_{hi}(1 - x^{(k+1)}_{hi})$. While giving $cx^{(k+1)}_{hi} \in CX^{(k+1)}$, CP calculates:

$$\begin{aligned} cx'^{(k+1)}_{hi} &= [1]_{pk_\Sigma} \cdot (cx^{(k+1)}_{hi})^{N-1} = [1 - x^{(k+1)}_{hi}]_{pk_\Sigma} \\ \phi^{(k+1)}_{hi} &= SMP(cx^{(k+1)}_{hi}, cx'^{(k+1)}_{hi}) \\ &= [x^{(k+1)}_{hi} \cdot (1 - x^{(k+1)}_{hi})]_{pk_\Sigma} = [f'(v^{(k+1)}_{hi})]_{pk_\Sigma}. \end{aligned}$$

Step-V(@CP). When $k = 2$, CP calculates the ciphertexts of $\delta^{(3)}_{hi}, h \in [1, L], i \in [1, n_3]$ through SMP and obtains the results as follow:

$$\begin{aligned} c\delta^{(3)}_{hi} &= SMP(ce_{hi}, \phi^{(3)}_{hi}) \\ &= [(\overline{y}_{hi} - y_{hi}) \cdot f'(v^{(3)}_{hi})]_{pk_\Sigma}. \end{aligned}$$

So CP can also calculate the encrypted gradients of bias for output layer as follow, $i \in [1, n_3], h \in [1, L]$:

$$\begin{aligned} [\Delta b^{(2)}_i]_{pk_\Sigma} &= [\Delta b^{(2)}_i]_{pk_\Sigma} \cdot c\delta^{(3)}_{hi} \\ &= [\Delta b^{(2)}_i + \delta^{(3)}_{hi}]_{pk_\Sigma}. \end{aligned}$$

Step-VI(@CP). When $k = 1$, CP calculates the ciphertexts of $\delta^{(2)}_{hi}, h \in [1, L], i \in [1, n_2], j \in [1, n_3]$ through SMP and obtains the results as follows:

$$\begin{aligned} \zeta_{hij} &= SMP\left(c\delta^{(3)}_{hj}, [w^{(2)}_{ij}]_{pk_\Sigma}\right) \\ &= \left[\delta^{(3)}_{hj} \cdot w^{(2)}_{ij}\right]_{pk_\Sigma} \\ c\delta^{(2)}_{hi} &= SMP\left(\prod_{j=1}^{n_3} \zeta_{hij}, \phi^{(2)}_{hi}\right) \\ &= \left[f'\left(v^{(2)}_{hi}\right) \cdot \sum_{j=1}^{n_3} \left(\delta^{(3)}_{hj} \cdot w^{(2)}_{ij}\right)\right]_{pk_\Sigma}. \end{aligned}$$

And the encrypted gradient of bias for hidden layer can also be calculated as follow, $i \in [1, n_2], h \in [1, L]$:

$$\begin{aligned} [\Delta b^{(1)}_i]_{pk_\Sigma} &= [\Delta b^{(1)}_i]_{pk_\Sigma} \cdot c\delta^{(2)}_{hi} \\ &= [\Delta b^{(1)}_i + \delta^{(2)}_{hi}]_{pk_\Sigma}. \end{aligned}$$

Step-VII(@CP). Finally, CP calculates the sum of encrypted gradients of L samples for each parameter weight $w^{(k)}_{ij}, k \in \{1, 2\}, i \in [1, n_k], j \in [1, n_{k+1}]$ and then updates the parameter variables. When $k = 1$, CP owns $[x^{(1)}_{hi}]_{pk_h}$ and $c\delta^{(2)}_{hj}$, uses the similar manner of *Step I-III* in Section 3.2 to compute the product of two ciphertexts which are encrypted with different public keys and obtains the results as $\varrho^{(1)}_{hij} = [x^{(1)}_{hi} \cdot \delta^{(2)}_{hj}]_{pk_\Sigma}$. When $k = 2$, CP uses the SMP protocol to compute the product of $cx^{(2)}_{hi}$ and $c\delta^{(3)}_{hj}$ and obtains the results as $\varrho^{(2)}_{hij} = [x^{(2)}_{hi} \cdot \delta^{(3)}_{hj}]_{pk_\Sigma}$. After that, CP calculates as follows, $k = \{1, 2\}, i \in [1, n_k], j \in [1, n_{k+1}], h \in [1, L]$:

$$\begin{aligned} [\Delta w^{(k)}_{ij}]_{pk_\Sigma} &= [\Delta w^{(k)}_{ij}]_{pk_\Sigma} \cdot \varrho^{(k)}_{hij} = [\Delta w^{(k)}_{ij} + x^{(k)}_{hi} \cdot \delta^{(k+1)}_{hj}] \\ [w^{(k)}_{ij}]_{pk_\Sigma} &= [w^{(k)}_{ij}]_{pk_\Sigma} \cdot ([\Delta w^{(k)}_{ij}]_{pk_\Sigma})^{N - \eta/L} \\ &= \left[w^{(k)}_{ij} - \frac{\eta}{L} \cdot \Delta w^{(k)}_{ij}\right]_{pk_\Sigma} [b^{(k)}_j]_{pk_\Sigma} \\ &= \left[b^{(k)}_j - \frac{\eta}{L} \cdot \Delta b^{(k)}_j\right]_{pk_\Sigma}. \end{aligned}$$

After a number of iterations in Sections 4.2 and 4.3, we can obtain the optimal deep learning model. Then, CP sends the partial decrypted parameters to SP to get the model parameters.

TABLE 2
Communication Cost Comparison with Prior Work

Privacy-preserving mechanism	Communication cost
PDLM	$O(L \cdot n_2(8n_1 + 7n_3))$
SecureML [22]	$O(L \cdot (n_1 + n_2) + n_1 \cdot n_2 + n_2 \cdot n_3)$
Zhang et al. [25]	$O(L \cdot (2Ln_1n_2 + n_1 + n_2))$

5 Theoretical Analysis

In this section, we theoretically show that PDLM fulfills the privacy and efficiency requirements.

5.1 Privacy Preservation

Based on the adopted security model shown in Section 3.4, we consider a protocol is secure if each party participating in it can be computed based on its input and output only. While giving the input and output only, we can simulate the party's view which is executed in the protocol. This implies that the party learns nothing from the execution of the protocol itself. Additionally, because the SSL/TLS protocols use an authentication policy to resist data tampering attacks, the transmitted data can be kept intact between two communicating entities.

Theorem 1. *The learning process in PDLM is secure against the adversary $\mathcal{A}^*$ defined in the attack model.*

Proof. The proof is given in appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TSC.2018.2868750. □

5.2 Efficiency Analysis

In this section, we study the communication costs and storage overhead in PDLM. At the beginning of feed forward ($k = 1$), CP calculates the products of $[X^{(k)}]_{pk_h}$ and $[W^{(k)}]_{pk_h}$, which costs CP $O(2L \cdot n_1 \cdot n_2)$ and SP $O(2L \cdot n_1 \cdot n_2)$ to transmit the encrypted data. CP then computes $\{[(v_{hi}^{k+1})^2]_{pk_\Sigma}, [(v_{hi}^{k+1})^3]_{pk_\Sigma}\}$ using SMP protocol and spends $O(2L \cdot n_2)$ to transmit data to SP. To return the results, SP also costs $O(2L \cdot n_2)$. When $k = 2$, CP and SP also cost $O(2L \cdot n_2 \cdot n_3)$ and $O(2L \cdot n_2 \cdot n_3)$ to transmit the ciphertexts respectively. Finally, CP calculates the error between model outputs and the true values, which costs CP $O(6L \cdot n_3)$ and SP $O(2L \cdot n_3)$ respectively. Thus, CP totally interacts 4 times with SP and the communication cost during one round feed forward is $O(4L \cdot n_2(n_1 + n_3))$. In addition, we assume that each ciphertext tuple requires $|N|$ to be stored. Hence, in feed forward, it costs CP $((n_2 + L + 1) \cdot n_1 + n_2 + (n_2 + L) \cdot n_3) \cdot |N|$ to store all the encrypted data. In all, another $((n_2 + L + 1) \cdot n_1 + n_2 + (n_2 + L) \cdot n_3) \cdot (|N| - 1)$ are needed in CP because of the adoption of the privacy-preserving technique.

While performing the privacy-preserving back propagation, CP first compares the computed error with threshold τ, which costs $O(1)$ to transmit the ciphertexts. After that, $c\delta_{hi}^{(k+1)}$ is calculated, which costs CP $O(2L \cdot (4n_2 + 5n_3) + 2n_2 \cdot n_3)$ and SP $O(L \cdot n_2 \cdot n_3)$ to transmit the intermediate results respectively. Finally, CP updates the values of weights and bias, which costs $O(2L \cdot (n_1 \cdot n_2 + n_3))$ to transmit. Correspondingly, SP also costs $O(2L \cdot n_2 \cdot (n_1 + n_3))$ to respond the results. Thus, CP totally interacts 9 times with SP and the communication cost during one round privacy-preserving back propagation is $O(L \cdot n_2 \cdot (4n_1 + 3n_3))$. Since no additional storage overhead is incurred, we do not analyze it in this process.

Finally, we also compare the communication cost with the prior work in [22] and [25]. As shown in Table 2, our PDLM has the same magnitude of communication cost with the mechanism in [25]. But the mechanism in [22] took less that. In our PDLM, most communication cost is to achieve the computation on multi-key encrypted ciphertexts. If DOs' data is encrypted by only one public key, we can optimize our mechanism and obtain lower communication cost.

6 Performance Evaluations

In this section, we present a series of empirical results of PDLM conducted over two real-world datasets, which indicated that PDLM can efficiently and effectively fulfill the aforementioned design goals. The experiments were conducted on a machine with a 2.4 GHz eight-core processor and 128 GB RAM.

Dataset. We mainly adopt the MNIST dataset [48] which is composed of 60,000 training handwritten digits and 10,000 test ones from "0" to "9". Each of them contains 784 features representing $28 * 28$ pixels in the image. Additionally, the CIFAR-10 dataset [49] is also used to evaluate the efficiency of PDLM. CIFAR-10 is composed of $32 * 32$ color images in 10 classes, with 6,000 images per class. In total, there are 50,000 training images and 10,000 test images. Then, based on the Torch7 *nn* [50] packages, we construct and train a LeNet deep learning model to evaluate the performance of our PDLM.

6.1 Classification Accuracy

In the experiment evaluation, we mainly focus on analyzing the classification accuracy of the trained model. To measure the accuracy, we first split the training data into three parts and assume that each DO contributes 20,000 examples to train the model. Then, we evaluate the accuracy loss which is caused by using Taylor theorem to approximate the Sigmoid function (AppSigmoid) and compare that with the non-privacy-preserving mechanism (NPP). The classification accuracy is defined as CN/SN, where CN represents the number of correctly classified items and SN represents the total number of test items.

First of all, we carry out the analysis on the classification accuracy of trained model influenced by the number of data. As shown in Fig. 2, we assume that three different DOs contribute their data to train the deep learning model and the results show that the classification accuracy increases with the number of data. With the training process, the parameters of weights and bias would be amended by the back propagation. So the classification of the test will be more accurate. And as the increase of the number of data (more DOs contribute more training data), the parameters will be more fully and reasonably amended. So we can obtain a more accurate trained model. As described above, the increase and diversity of the training data will make the deep learning models better.

In the following, we also compare the classification accuracy of PDLM with the **N**on-**P**rivacy-**P**reserving mechanism (NPP, as the baseline). We analysis the accuracy loss by

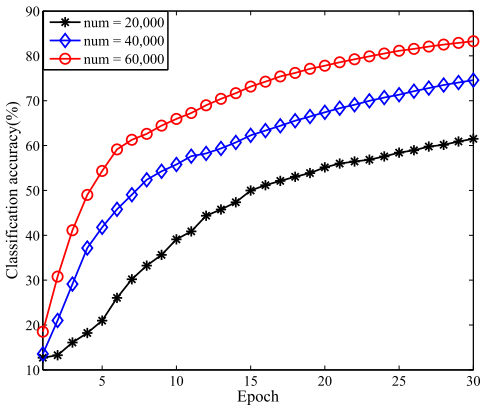


Fig. 2. Classification accuracy influenced by number of data.

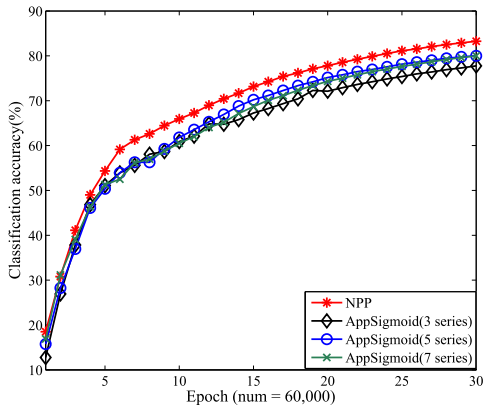


Fig. 3. Accuracy loss caused by Taylor theorem.

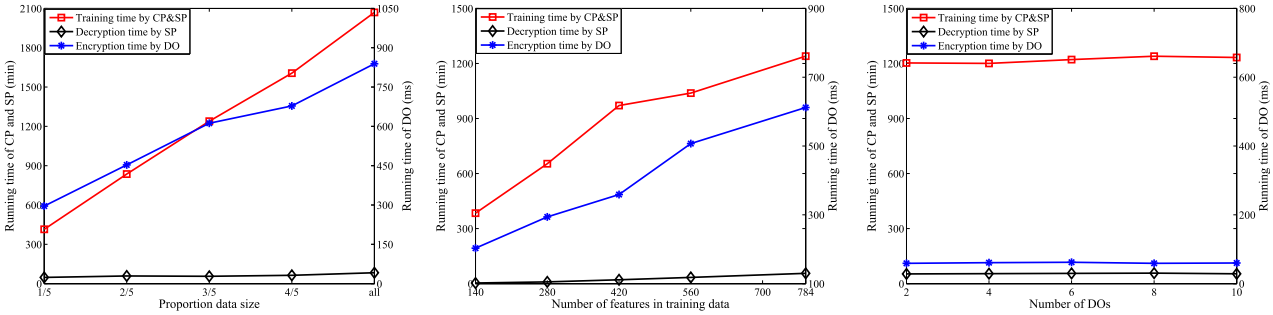
extending the number of Taylor series terms from 3 to 7. As shown in Fig. 3, as the number of Taylor series terms grows, the classification accuracy increases but not so much. In other words, adding more Taylor series terms can reduce the accuracy loss but the impact is minimal. Specifically, the classification accuracy increases about 2 percent by extending the number of Taylor series terms from 3 to 5. And from 5 to 7, it only increases 0.5 percent. Additionally, compared with NPP, the accuracy of our PDLM only decreases by 2 percent when we extend the series term to 7. So we realize that the approximation of Sigmoid function is feasible and we can use it in our privacy-preserving mechanism. As a result, due to the introduction of privacy protection, the training of deep learning model will obtain a suboptimal result, up to 5 percent loss in the aspect of classification accuracy when we extend the Taylor series term to 3.

6.2 Training Efficiency

To test the efficiency of our PDLM, we first discuss the computation costs in different stages by varying the number of training data, the number of features in training data, the number of DOs and the number of nodes in hidden layer. Then we consider the influence of Taylor series on computation cost and weight the efficiency and accuracy to determine that which series is reasonable. Varying with the factors, we evaluate the running time of three stages in PDLM, including the running time for encrypting the training data ($@DO_i$), the running time for training the model ($@CP\&SP$), and the running time for decrypting the encrypted parameters ($@SP$). It should be noted that all the reported time are counted over ten epochs and averaged over 10 runs.

In Fig. 4, we plot the running time of our PDLM by varying with the training data and DOs. As shown in Fig. 4a, we first evaluate the efficiency against the number of training data. The simulation results show that the encryption time and training time clearly increase with the number of training data. As the number of training data increase, more and more data needs to be encrypted and DO will take more time to do that. However, DO only need 839 ms to encrypt all training data samples. Additionally, we also find that the training time increase with the number of training data while the decryption time remains stable. The reason for that is CP and SP have more encrypted data to train but the number of encrypted trained parameters is invariable. So CP and SP will spend more time on training while SP does not need more time to decrypt the encrypted parameters of weights and bias.

In Fig. 4b, we evaluate the efficiency of our PDLM against the number of features in training data. The simulation results show that the training time, encryption time, and decryption time always increase with the number of features. As the features increase, more pixels of the handwritten images need to be encrypted and trained. So DO spends more time to encrypt the training data. While the input features increase, there will be more neural connections to be established and more parameters to be trained. So CP and SP spend more time training the model. Finally, since more parameters are generated between the input layer and hidden layer, SP also spends more time decrypting.

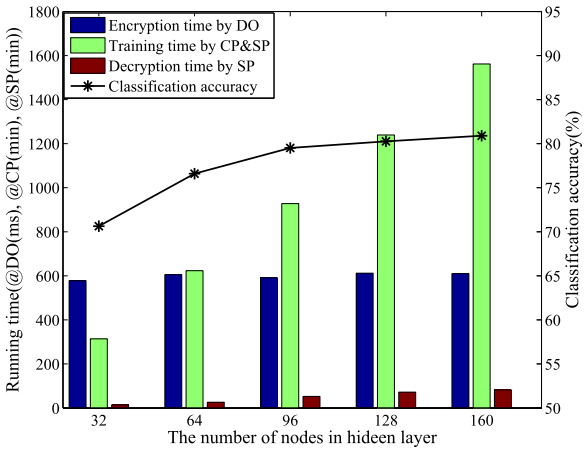


(a) Efficiency against the number of training data

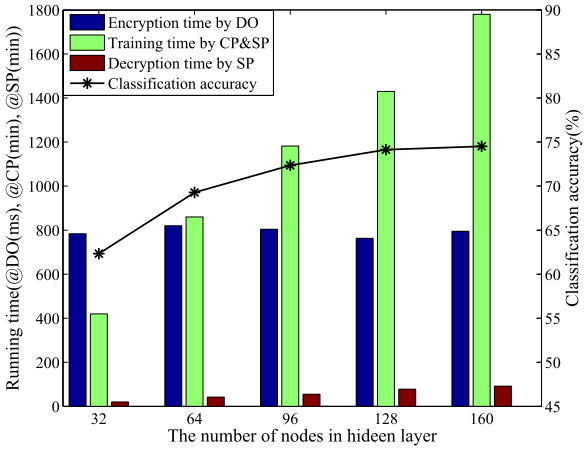
(b) Efficiency against the number of features in training data

(c) Efficiency against the number of DOs

Fig. 4. Efficiency evaluation against training data and DOs.

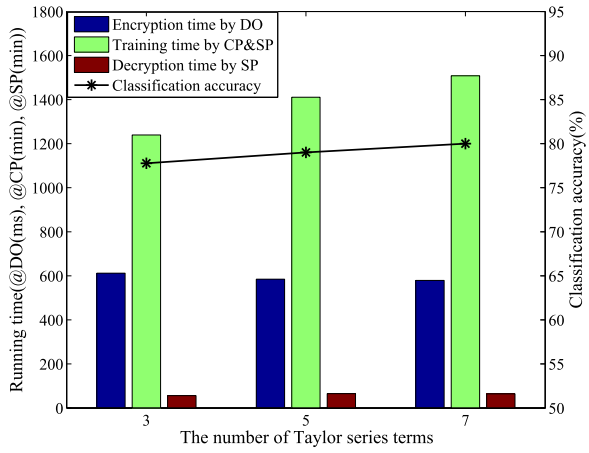


(a) Efficiency evaluation on MNIST

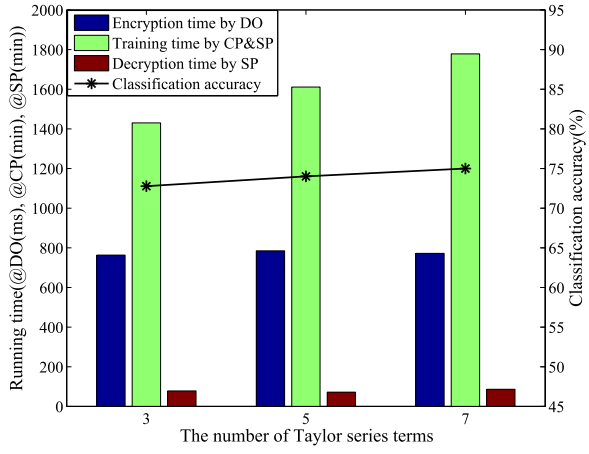


(b) Efficiency evaluation on CIFAR-10

Fig. 5. Efficiency against the number of nodes in hidden layer.



(a) Efficiency evaluation on MNIST



(b) Efficiency evaluation on CIFAR-10

Fig. 6. Efficiency against the number of Taylor series terms.

Then, we also plot the running time of our PDLM by varying with the number of DOs. We assume that the total training data is unchanged and more DOs just means more different keys are introduced. As shown in Fig. 4c, the encryption time of DOs is remained unchanged. The reason for that is the varying of DOs does not change the input data and its features, so the encryption time by DOs is stable. Additionally, the training time and decryption time by CP and SP also remain basically unchanged. Although more encryption keys are introduced, the calculation and interaction processes between CP and SP do not increase with that. So the training time by CP and SP remains unchanged. After the training process, the parameters are encrypted by DOs' union public key and SP decrypts them by the partial strong private key. So SP also does not need any more time to decrypt the encrypted parameters.

In Fig. 5, based on the MNIST and CIFAR-10 datasets, we plot the classification accuracy and running time of the stages by varying with the number of nodes in hidden layer. The simulation results show that the training time by CP and SP is clearly increase with the number of nodes in hidden layer. The reason for that is CP and SP have to perform more calculations to compute the relationships and update the parameters between nodes with the increased number of nodes in hidden layer. And in the above process, more parameters are generated. So SP also spends more time decrypting the encrypted parameters. The increase of nodes in hidden layer does not affect the input training data, so the encrypting time by DO is stable. Additionally, with the increase of nodes in hidden layer, the classification accuracy increases significantly. The reason for that is more nodes in hidden layer will extract more abstract features based on the inputs and it can fully reflect the relationships between nodes and obtain a more detailed classification. However, we can also find that the classification accuracy increase is small by increasing the number of node from 128 to 160. So we realize that designing 128 nodes in hidden layer is feasible and more nodes will reduce the performance of PDLM significantly with the negligible accuracy improvement. What's more, because more features in CIFAR-10 are input to the model, CP and SP spent more time on training the model than that for MNIST dataset.

Then, based on the MNIST and CIFAR-10 datasets, we plot the classification accuracy and running time of the stages by varying with the number of Taylor series terms. As shown in Fig. 6, the training time by CP and SP increases

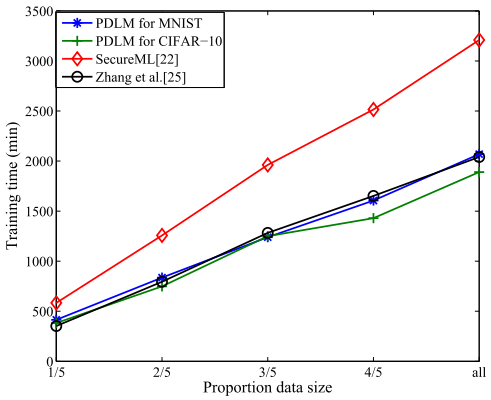


Fig. 7. Efficiency comparison with prior work.

with the number of Taylor series terms. The reason for this is that with the extending of Taylor series terms, CP and SP have to calculate more series in approximation of Sigmoid function in feed forward stage. And since the extending of series terms has no effect on the number of parameters and input training data, the decryption time by SP and encryption time by DO remain unchanged. However, the extending of Taylor series terms will reduce the classification accuracy loss, which has been indicated in Fig. 3. Hence, based on the increased computational overhead and reduced accuracy loss, we realize that extending the Taylor series term to 3 is the most reasonable choice. Since more parameters are used to connect the input layer and hidden layer in CIFAR-10, SP spents more time on decrypting the encrypted parameters than that for MNIST dataset.

Finally, we also discuss the efficiency of PDLM along with the privacy-preserving mechanisms in [22] and [25]. Generally speaking, the training time will increase with the growth of the number of training data and features [25] and we have verified that in Figs. 4a and 4b. So we only compare the training time with two privacy-preserving mechanisms by varying with the number of training data. As shown in Fig. 7, the training time of three mechanisms varies as expected, which increases with the number of training data. However, based on all datasets, our PDLM requires 2069.7 min and 2041.3 min to train the model for MNIST and CIFAR-10 respectively, but the mechanism in [22] will spent 3209.3 min to train that over the same number of epochs. While given one simulation node in cloud, the mechanism in [25] also required about 2041 min to train the model for the same MNIST dataset. But by varying with the number of cloud nodes, their training time will be greatly reduced, which is shown in Fig. 10 in [25]. Therefore, we believe that our PDLM is efficient enough and can be more efficient when we adopt a multi-node cloud platform. Since the training process is conducted over multi-key encrypted datasets, our PDLM has obvious lower efficiency than the conventional non-privacy-preserving mechanism.

7 Conclusion

The sensitive data which is collected to discover valuable information by deep learning model seriously threats users' personal privacy, especially when service providers move the collected data to untrusted clouds. In this paper, we present a novel solution, namely PDLM, to address the grand challenges in privacy-preserving deep learning model. In PDLM, we consider different DOs encrypt their data with multiple keys and upload the encrypted data to SP. Then, SP and CP will train the model based on the multi-key encrypted data with an efficient privacy-preserving calculation toolkit. Moreover, we evaluate the effectiveness and performance of PDLM and the results justify that PDLM is effective and efficient. As part of our future work, we will consider the different distributions of collected training data and design a more efficient privacy-preserving mechanism to achieve the deep learning for different DOs.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant Nos. U1405255, 61672413, 61672408, 61602537, 61602357), China Postdoctoral Science Foundation Funded Project (Grant No. 2016M592762), Shaanxi Science & Technology Coordination & Innovation Project (Grant No. 2016TZC-G-6-3), Central University of Finance and Economics program of the Youth Talent Support Plan (No. QYP1808), CCF-VenustechRP (No. 2017005), Fundamental Research Funds for the Central Universities (No. JB181505), Huawei Innovation Research Program, Natural Science Basic Research Plan in Shaanxi Province of China (No. 2018JM6073), China 111 Project (Grant No. B16037), Beijing Municipal Social Science Foundation (Grant No. 16XCC023).

References

[1] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 308–318.

[2] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, et al., "Deep speech: Scaling up end-to-end speech recognition," arXiv:1412.5567, 2014.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1026–1034.

[4] D. Shultz, "When your voice betrays you," *Sci.*, vol. 347, no. 6221, pp. 494–494, 2015.

[5] A. Levi, O. Mokryn, C. Diot, and N. Taft, "Finding a needle in a haystack of reviews: Cold start context-based hotel recommender system," in *Proc. 6th ACM Conf. Recommender Syst.*, 2012, pp. 115–122.

[6] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1310–1321.

[7] X. Liu, R. H. Deng, K.-K. R. Choo, and J. Weng, "An efficient privacy-preserving outsourced calculation toolkit with multiple keys," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 11, pp. 2401–2414, Nov. 2016.

[8] X. Ma, H. Li, J. Ma, Q. Jiang, S. Gao, N. Xi, and D. Lu, "APPLET: A privacy-preserving framework for location-aware recommender system," *Sci. China Inf. Sci.*, vol. 60, no. 9, 2017, Art. no. 092101.

[9] X. Ma, J. Ma, H. Li, Q. Jiang, and S. Gao, "AGENT: An adaptive geo-indistinguishable mechanism for continuous location-based service," *Peer-to-Peer Netw. Appl.*, vol. 11, no. 3, pp. 473–485, 2018.

[10] S. Gao, J. Ma, W. Shi, G. Zhan, and C. Sun, "TrPF: A trajectory privacy-preserving framework for participatory sensing," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 6, pp. 874–887, Jun. 2013.

[11] S. Gao, J. Ma, C. Sun, and X. Li, "Balancing trajectory privacy and data utility using a personalized anonymization model," *J. Netw. Comput. Appl.*, vol. 38, pp. 125–134, 2014.

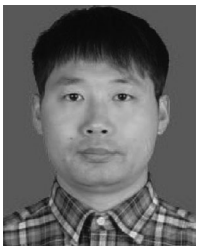
[12] Q. Jiang, S. Zeadally, J. Ma, and D. He, "Lightweight three-factor authentication and key agreement protocol for internet-integrated wireless sensor networks," *IEEE Access*, vol. 5, no. 1, pp. 3376–3392, Mar. 2017.

[13] Q. Jiang, J. Ma, C. Yang, X. Ma, J. Shen, and S. A. Chaudhry, "Efficient end-to-end authentication protocol for wearable health monitoring systems," *Comput. Elect. Eng.*, vol. 63, pp. 182–195, 2017.
[14] X. Ma, J. Ma, H. Li, Q. Jiang, and S. Gao, "ARMOR: A trust-based privacy-preserving framework for decentralized friend recommendation in online social networks," *Future Generation Comput. Syst.*, vol. 79, pp. 82–94, 2018.
[15] D. Wu, F. Zhang, H. Wang, and R. Wang, "Security-oriented opportunistic data forwarding in mobile social networks," *Future Generation Comput. Syst.*, vol. 87, pp. 803–815, 2018.
[16] D. Wu, S. Si, S. Wu, and R. Wang, "Dynamic trust relationships aware data privacy protection in mobile crowd-sensing," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2958–2970, Aug. 2018.
[17] Y. Lindell and B. Pinkas, "Privacy preserving data mining," in *Proc. 20th Annu. Int. Cryptology Conf.*, 2000, pp. 36–54.
[18] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2000, pp. 439–450.
[19] W. Du, Y. S. Han, and S. Chen, "Privacy-preserving multivariate statistical analysis: Linear regression and classification," in *Proc. 4th SIAM Int. Conf. Data Mining*, 2004, pp. 222–233.
[20] J. Vaidya, M. Kantarcioglu, and C. Clifton, "Privacy-preserving Naïve Bayes classification," *Very Large Database Endowment J.*, vol. 17, no. 4, pp. 879–898, 2008.
[21] H. Miyajima, N. Shigei, S. Makino, H. Miyajima, Y. Miyanishi, S. Kitagami, and N. Shiratori, "A proposal of privacy preserving reinforcement learning for secure multiparty computation," *Artif. Intell. Res.*, vol. 6, no. 2, pp. 57–68, 2017.
[22] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 19–38.
[23] Q. Wang, S. Hu, M. Du, J. Wang, and K. Ren, "Learning privately: Privacy-preserving canonical correlation analysis for cross-media retrieval," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.
[24] R. Yonetani, V. N. Boddeti, K. M. Kitani, and Y. Sato, "Privacy-preserving visual learning using doubly permuted homomorphic encryption," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2059–2069.
[25] Q. Zhang, L. T. Yang, and Z. Chen, "Privacy preserving deep computation model on cloud for big data feature learning," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1351–1362, May 2016.
[26] J. Yuan and S. Yu, "Privacy preserving back-propagation neural network learning made practical with cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 212–221, Jan. 2014.
[27] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 5, pp. 1333–1345, May 2018.
[28] E. Hesamifard, H. Takabi, M. Ghasemi, and R. N. Wright, "Privacy-preserving machine learning as a service," *Proc. Privacy Enhancing Technol.*, vol. 2018, no. 3, pp. 123–142, 2018.
[29] P. Li, J. Li, Z. Huang, T. Li, C. Gao, S. Yiu, and K. Chen, "Multi-key privacy-preserving deep learning in cloud computing," *Future Generation Comput. Syst.*, vol. 74, pp. 76–85, 2017.
[30] C. Dwork, "Differential privacy," in *Proc. 33rd Int. Colloq. Automata Languages Program.*, 2006, pp. 1–12.
[31] T. Zhang and Q. Zhu, "Dynamic differential privacy for ADMM-based distributed classification learning," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 1, pp. 172–187, Jan. 2017.
[32] N. Phan, X. Wu, H. Hu, and D. Dou, "Adaptive laplace mechanism: Differential privacy preservation in deep learning," in *Proc. IEEE Int. Conf. Data Mining*, 2017, pp. 385–394.
[33] L. Xie, I. M. Baytas, K. Lin, and J. Zhou, "Privacy-preserving distributed multi-task learning with asynchronous updates," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 1195–1204.
[34] J. Hamm, A. C. Champion, G. Chen, M. Belkin, and D. Xuan, "Crowd-ML: A privacy-preserving learning framework for a crowd of smart devices," in *Proc. 35th IEEE Int. Conf. Distrib. Comput. Syst.*, 2015, pp. 11–20.
[35] Y. Li, J. Yang, and W. Ji, "Local learning-based feature weighting with privacy preservation," *Neurocomput.*, vol. 174, pp. 1107–1115, 2016.
[36] M. Avriel, *Nonlinear Programming: Analysis and Methods*. Chelmsford, MA, USA: Courier Corporation, 2003.
[37] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, Art. no. 116.
[38] A. Peter, E. Tews, and S. Katzenbeisser, "Efficiently outsourcing multiparty computation under multiple keys," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 12, pp. 2046–2058, Dec. 2013.
[39] B. Wang, M. Li, S. S. M. Chow, and H. Li, "A tale of two clouds: Computing on data encrypted under multiple keys," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2014, pp. 337–345.
[40] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 1999, pp. 223–238.
[41] E. Bresson, D. Catalano, and D. Pointcheval, "A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications," in *Proc. 9th Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2003, pp. 37–54.
[42] Q. Do, B. Martini, and K.-K. R. Choo, "A forensically sound adversary model for mobile devices," *PloS One*, vol. 10, no. 9, 2015, Art. no. e0138449.
[43] Y. Yang, X. Liu, and R. Deng, "Multi-user multi-keyword rank search over encrypted data in arbitrary language," *IEEE Trans. Depend. Secure Comput.*, vol. 1, no. 1, pp. 1–19, 2017.
[44] O. Goldreich, *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[45] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in *Proc. 22nd Annu. Netw. Distrib. Syst. Secur. Symp.*, 2015, pp. 222–233.
[46] S. Kamara, P. Mohassel, and M. Raykova, "Outsourcing multiparty computation," *IACR Cryptology ePrint Archive*, vol. 2011, 2011, Art. no. 272.
[47] A. Genocchi, *Calcolo Differenziale E Principii Di Calcolo Integrale*, vol. 1. Bocca, Torino, 1884.
[48] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[49] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, vol. 1, no. 4, pp. 7–67, 2009.
[50] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A Matlab-like environment for machine learning," in *BigLearn, NIPS Workshop*, no. EPFL-CONF-192376, 2011, pp. 1–6.

Xindi Ma received the BS degree from the School of Computer Science and Technology, Xidian University, in 2013, and the PhD degree in computer science from Xidian University, in 2018. He is now a postdoctor with the School of Cyber Engineering, Xidian University. His current research interests include privacy computing, recommender system and machine learning with focus on security and privacy issues.

Jianfeng Ma received the BS degree in computer science from Shaanxi Normal University, in 1982, and the MS and PhD degrees in computer science from Xidian University, in 1992 and 1995, respectively. Currently, he is a professor with the School of Cyber Engineering, Xidian University. He has published more than 150 journal and conference papers. His research interests include information security, cryptography, and network security.

Hui Li received the BEng degree from the Harbin Institute of Technology, in 2005, and the PhD degree from Nanyang Technological University, Singapore, in 2012. He is now a professor with the School of Cyber Engineering, Xidian University, China. His research interests include data mining, knowledge management and discovery, privacy-preserving query, and analysis in big data.



Qi Jiang received the BS degree in computer science from Shaanxi Normal University, in 2005, and the PhD degree in computer science from Xidian University, in 2011. He is now an assistant professor with the School of Cyber Engineering, Xidian University. His research interests include security protocols and wireless network security, cloud seurity, etc.

Sheng Gao received the BS degree in information and computation science from the Xi'an University of Posts and Telecommunications, in 2009, and the PhD degree in computer science and technology from Xidian University, in 2014. He is an assistant professor with the School of Information, Central University of Finance and Economics. His current research interests include finance information security and privacy computing.