Future Generation Computer Systems 74 (2017) 76–85

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Multi-key privacy-preserving deep learning in cloud computing

Ping Li[a], Jin Li[a,*], Zhengan Huang[a], Tong Li[b], Chong-Zhi Gao[a], Siu-Ming Yiu[c], Kai Chen[d]

[a] *School of Computational Science & Education Software, Guangzhou University, 510006, Guangzhou, PR China*

[b] *College of Computer & Control Engineering, Nankai University, 300071, Tianjin, PR China*

[c] *Department of Computer Science, The University of Hong Kong, Hong Kong, PR China*

[d] *Institute of Information Engineering, Chinese Academy of Sciences, Beijing, PR China*

HIGHLIGHTS

- In the basic scheme, we use M-FHE as our privacy-preserving technique. Only the decrypt operation needs the interaction among data owners.
- In the advanced scheme, we propose a hybrid structure scheme by combining the double decryption mechanism and FHE.
- In the advanced scheme, only the encrypt and decrypt algorithms are performed by data providers.
- We prove that these two multi-key privacy-preserving deep learning schemes over encrypted data are secure.

ARTICLE INFO

Article history:
Received 21 December 2016
Received in revised form
28 January 2017
Accepted 6 February 2017
Available online 22 March 2017

Keywords:
Cryptography
Machine learning
Fully homomorphic encryption
Cloud computing

ABSTRACT

Deep learning has attracted a lot of attention and has been applied successfully in many areas such as bioinformatics, imaging processing, game playing and computer security etc. On the other hand, deep learning usually requires a lot of training data which may not be provided by a sole owner. As the volume of data gets huge, it is common for users to store their data in a third-party cloud. Due to the confidentiality of the data, data are usually stored in encrypted form. To apply deep learning to these datasets owned by multiple data owners on cloud, we need to tackle two challenges: (i) the data are encrypted with different keys, all operations including intermediate results must be secure; and (ii) the computational cost and the communication cost of the data owner(s) should be kept minimal. In our work, we propose two schemes to solve the above problems. We first present a *basic scheme* based on multi-key fully homomorphic encryption (MK-FHE), then we propose an *advanced scheme* based on a hybrid structure by combining the double decryption mechanism and fully homomorphic encryption (FHE). We also prove that these two multi-key privacy-preserving deep learning schemes over encrypted data are secure.

1. Introduction

Cloud computing provides fundamental support to address the challenges with shared computing resources including computing, storing, networking and analytical software. The application of these resources has fostered impressive Big Data advancements [1, 2] and Internet of Things (IoT). Because of the wide range of applications, people have paid more attention to cloud security for data management, data storing, and data processing in recent decades. Recently, deep learning has made impressive success in a wide range of applications, such as bioinformatics, image processing, game playing, natural language processing, computer security etc. On the other hand, in order to get an accurate result without over-fitting, deep learning requires a lot of training records to determine tens of thousands of parameters. In many cases, the massive amount of training data cannot be provided by one single user, but collected from different users. The trend is to have these huge datasets stored in an untrusted third-party cloud system in encrypted form due to the confidentiality and sensitivity of the datasets.

To also leverage the computing ability provided by the cloud platform, more and more applications choose to conduct the deep learning on the cloud platform. In fact, if the amount of training records is huge, it is difficult, to download all records by a single user for processing. This poses a new challenge to the community, i.e., how to perform deep learning over outsourced encrypted data owned by multiple users in cloud. In this paper, we consider to develop a multi-key privacy-preserving deep neural network

[*] Corresponding author.
E-mail address: jinli71@gmail.com (J. Li).

http://dx.doi.org/10.1016/j.future.2017.02.006
0167-739X/© 2017 Elsevier B.V. All rights reserved.

in cloud over encrypted data. The key challenges include (1) Data are located in different places and encrypted with different keys. To protect data privacy, all computation (e.g. inner product and the approximation of nonlinear sigmoid function used in deep learning), intermediate results generated during the deep learning process and the learning results must be secure. (2) To improve the efficiency of the deep learning process, computation should be done by the cloud server so as to decrease the computation/communication cost of the data owner(s). Existing solutions such as secure multi-party computation (SMC) [3], encryption schemes, garbled circuit, and detective controls were designed for other scenarios and cannot be applied directly to tackle these two challenges.

Our Contributions. To solve the above challenges, this paper designs two schemes to support multi-key learning system. Both schemes allow multiple data owners with different datasets to collaboratively learn a neural network model securely in cloud computing. To protect the confidentiality, data owners encrypt their sensitive data with different public keys before uploading to cloud server.

We first propose a *basic scheme* which is based on multi-key fully homomorphic encryption (MK-FHE) [4–6]. In this scheme, multiple data owners send their data (encrypted with *different public keys* chosen by data owners independently of each other) to an untrusted cloud server. Cloud server computes the output of deep learning on this joint data and issues it back to all participating data owners. Finally, all of the data owners jointly perform a secure SMC protocol to decrypt and extract results from this encrypted deep learning results.

To avoid the interaction among multiple data owners, we further propose an *advanced scheme* which is based on a hybrid structure by combining the double decryption mechanism (BCP scheme [7]) and fully homomorphic encryption (FHE) [8]. If we only use BCP scheme to support the secure computation, in the training phase, both the computation of inner product of the inputs and weights, and the computation of the activation function require additional communication with the cloud server. To solve this challenge, we introduce FHE scheme directly by transforming BCP ciphertext into FHE ciphertext, such that the computations over FHE ciphertext can be realized without interaction. In this scheme, a cloud server $\mathcal{C}$ and an authorized center (a trusted third party) $\mathcal{AU}$ are queried, which is assumed to be non-colluding and *honest-but-curious*. The cloud server $\mathcal{C}$ keeps the encrypted datasets under *different public keys* uploaded by multiple data owners. The authorized center $\mathcal{AU}$, on the other hand, only holds the master key of the master decryption of BCP scheme and the private key of FHE. In this paper, all participants are assumed to be *honest-but-curious*.

In summary, our contributions can be summarized as follows:

- We address a multi-key privacy-preserving deep learning in cloud computing by proposing two schemes, which allow multiple data owners to conduct collaboratively privacy-preserving deep learning.
- Our multi-key privacy-preserving deep learning schemes are able to preserve the privacy of sensitive data, intermediate results as well as the training model.
- We provide a security analysis to guarantee the privacy-preserving of our proposed two schemes.
- We give an application of our *advanced scheme* in face recognition. Note that our solutions are generic and can be applied to perform many other machine mining with the same setting over the same setting.

Organization. The rest of this paper is organized as follows. In Section 2, we briefly discuss the related work. Some notations, including deep learning, stochastic gradient descent, BCP scheme, FHE, and MK-FHE will be described in Section 3. We give the system model definition and describe the details of our privacy-preserving deep learning system in Section 4 and Section 5, respectively. Section 6 shows the complexity and security analysis for the proposed system. And we give an application in our system in Section 7. Finally, we conclude the paper in Section 8.

2. Related work

2.1. Deep learning

In cloud computing, deep learning has shown its success in many cases such as image recognition [9,10], speech recognition [11], and biomedical data analysis [12]. Deep learning is able to transform the original data into a higher level and more abstract expression. It means that high-dimensional original data can be converted to low-dimensional data by training a multiple neural network with a small central layer to reconstruct high-dimensional input data. Through these transformations, complicated functions can be learned by composing many simple functions. Hinton et al. [13] showed that multiple hidden layers of artificial neural network have excellent characteristics of learning ability. The characteristic obtained by learning are more intrinsic characterization of the data that facilitates an improved visualization or classification of the data. They also showed that the difficulty to optimize the weights in nonlinear auto-encoders can be overcome by layer-by-layer "pretraining" procedure. Usually, deep learning architectures are constructed as multi-layer neural networks. There are several different neural architectures, such as the feed-forward neural network, Recurrent Neural Network (RNN), and Deep Belief Network (DBN).

2.2. Privacy-preserving machine learning

With the advance of cloud computing, some related works have addressed some security problems in cloud, such as the security for the cloud framework [14,15], location privacy in mobile cloud [16, 17], security in cloud storage [18–20], data mining [21–24] and machine learning [25–27]. Existing privacy-preserving techniques are based mainly on data perturbation method and cryptographic methods (such as secure multi-party computation and secure function evaluation).

In the data perturbation method, differential privacy [28–30] has been widely applied to protect privacy of statistical database. Generally speaking, differential privacy guarantees that the removing or adding one record (usually known as noise) does not (substantially) affect the outcome of any usefully analysis. Therefore no risk is incurred by joining the database, providing a mathematically rigorous means of coping with the fact that distributional information may be disclosive. For instance, Abadi et al. [31] proposed a new algorithms, which are based on differential privacy version of stochastic gradient descent (SGD) process. In their work, scaled noise is added to the computed gradient to prevent information leakage. They also implemented the model on several neural networks and analyzed the privacy leakage. In [32], the authors considered another approach, in which they proposed a distributed selective SGD by collecting computed gradients from different parties. Then the users update the parameters of deep learning model selectively according to the collected gradients. The selective SGD can ensure data privacy because the computation process is held locally and only gradients will be reported to the central server. However, it is more difficult for selective SGD to achieve the *global*/*local* optimal compared to use conventional SGD with the entire dataset. In other words, the data are not fully utilized in selective SGD.

For the cryptographic methods, the proposed schemes based on privacy-preserving machine learning have been presented recently. In these existing schemes, there are two different settings: (1) training without the aid of cloud/third-party, and (2) training with the aid of cloud/third-party. In the first setting, the authors in [33,34] proposed a privacy-preserving two-party distributed algorithm for back-propagation training with vertically partitioned data and arbitrarily partitioned data, respectively. They used ElGamal scheme to support the secure computation operations. In the second setting, the authors used the SMC technique to train the horizontal partitioned data for multi-party case [35]. Graepel et al. [27] demonstrated that some basic machine learning algorithms, such as simple linear classifiers, can be performed efficiently over a small scale encrypted datasets. However, the efficiency will be degraded rapidly when the input size grows large. Yuan et al. [36] adopted a doubly homomorphic encryption scheme (BGN) [37] and proposed a system in which the training phase can be securely delegated to a cloud system for the multi-party scenario. In [38], the authors used BGV scheme [39] to support the secure computation operations and realized a high-order back-propagation algorithm efficiently for deep computation model training on the cloud.

Recently, researchers also proposed a CryptoML [40] framework for secure delegation of iterative machine learning to untrusted cloud servers. This secure delegation protocol is based on Shamir's secret sharing model. However, none of the existing crypto-based schemes are able to deal with data encrypted with *different public keys*. In our work, we propose two solutions to tackle this challenge. We show how to achieve privacy-preserving deep learning for training encrypted datasets under *different public keys*. In our two schemes, the data owners encrypt the date before uploading it to the cloud server. Most of the computation is performed by the cloud server and only the data owners are able to obtain the final results of the training model.

3. Preliminaries

3.1. Deep learning

Deep learning can be viewed as a multi-layer neural network. The input data or variables that we are able to observe is presented at the *input layer*. There are also several *hidden layers*, which extract increasingly abstract features from the input layer. They are called "hidden" because the parameters for these layers are not given in the data. During the learning process, the model must determine which features are useful for explaining the relationships in the input data. Precisely, we take a vector of real-valued as input, calculate a linear combination of these vector and corresponding weight, and the output of the neuron by applying a nonlinear activation function to the total input value is defined as

$$x_k = f(x_{k-1}W_k + b) \tag{1}$$

where f is an activation function and W_k is a real-valued constant matrix in hidden layer k, or *weight matrix*, which determines the contribution of each input to the output. Parameter b is called *bias*, which guarantees that the input sum is greater than 0. Algorithm 1 describes the learning process of a multi-layer back-propagation neural network and Fig. 1 shows a neural network with two hidden layers and two output nodes.

Algorithm 1 Multi-layer Back-propagation network learning

```
Input: input sample $x$, target vector $t$, learning rate $\eta$, sigmoid
    function $f(x)$, and network depth $l$;
Output: the weight matrices of the model: $W^{(i)}, i \in \{1, 2, \cdots, l\}$
Feed Forward Stage:
$h^{(0)} = x$;
for $k$ = 1 to $l$ do
    $v^{(k)} = W^{(k)} h^{(k-1)}$;
    $h^{(k)} = f(v^{(k)})$;
end for
$y = h^{(l)}$
$E = E(t, y)$//compute the cost function
Back-propagation Stage:
$e \leftarrow \nabla_y E = \nabla_y E(t, y)$//compute the gradients of $E$ with
    parameter $W$
for $k = l$ to 1 do
    $e \leftarrow \nabla_{v^{(k)}} E = e f'(v^{(k)})$//convert the gradient on the layer's
        output into the gradient pre-nonlinearity activation
    $\nabla_{W^{(k)}} E = e h^{(k-1)\tau}$//compute the gradients on weights
    $e \leftarrow \nabla_h^{(k-1)} E = W^{(k)\tau} e$//modify the next lower-level hidden
        layer's activations
end for
```

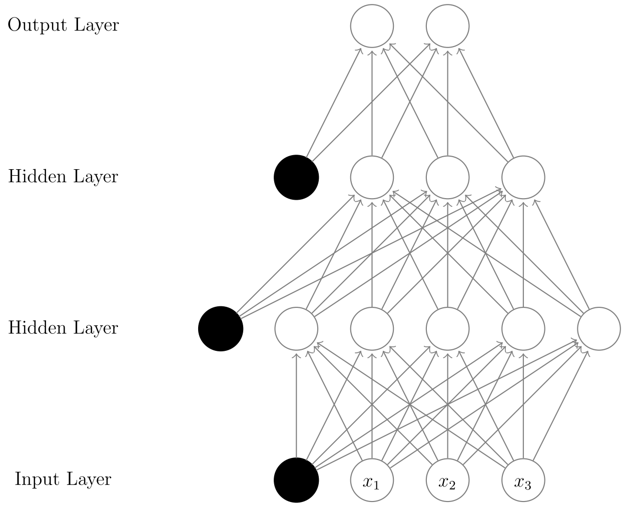


Fig. 1. Configuration of neural network with two hidden layers and two output nodes.

One of the activation functions is the *logistic sigmoid* $f(z) = \frac{1}{1+\exp(-z)}$. Usually, the logistic sigmoid function is used to produce the ϕ parameter of a Bernoulli distribution due to its range is $(0, 1)$. Another activation function is *softplus* function, $\zeta(z) = \log(1 + \exp(z))$, which can be used to produce the β or σ parameter of a normal distribution because its range is $(0, \infty)$.

Gradient-Based Optimization. Most deep learning algorithms involve optimization problem. Optimization problem aims to solve the task of either minimizing or maximizing some function $f(\mathbf{x})$. Hence, this function f is called *objective function*. If we choose to minimize it, then this function is called the *error function*, *loss function*, or *cost function*.

Assuming that there is a function $y = f(\mathbf{x})$, where $\mathbf{x}$ is vector, and y is a real number. Let $f'(\mathbf{x})$ or $\frac{\partial y}{\partial \mathbf{x}}$ denote the *derivative* of this function f, its Taylor series expansion is

$$f(\mathbf{x} + \varepsilon) \approx f(\mathbf{x}) + \varepsilon \cdot f'(\mathbf{x}). \tag{2}$$

The above Eq. (2) shows how to scale a small change ε in the input data $\mathbf{x}$, in order to make a small change in y. Therefore, we can use the derivative for minimizing a function. This technique is called *gradient descent*. If $f'(\mathbf{x}) = 0$, then the tangent orientation at point $\mathbf{x}$ is 0, it means that no information provided about what direction to move. Such points are called *critical points*, which can be classified as *local minimum* point and *global minimum* point.

Stochastic gradient descent (SGD). When we perform the gradient descent algorithm, there are two problems to be considered: the problem of the speed of the convergence to a local minimum and the problem in the error surface, there are many local minima error does not mean that the global minimum error is found. Hence, to save the computation per weight update step, a good idea for optimizing the algorithm is to use some training samples at a time, i.e., the weights are updated upon examining each individual training example. This optimization algorithm is called *stochastic* gradient descent (SGD), which can be seen as an extension of the gradient descent algorithm.

In general, the gradient in stochastic gradient descent is viewed as an expectation. This expectation can be calculated by using a subset of training sample set. In more details, assume that n is the training sample set size, we choose a subset, denoted as $X = \{X_1, \ldots, X_{n'}\}$, which is uniformly sampled from the training sample set. The size n' is called *mini-batch*, and is smaller than n. Usually, n' is fixed, and independent of n. Algorithm 2 shows the detail of the SGD by taking the average gradient on a *mini-batch* of n examples.

Algorithm 2 Stochastic gradient descent (SGD)

```
Input: Learning rate $\eta$
Output: Initial weight parameter $w$
  while stopping criterion not met do
    Sample a mini-batch of $n'$ examples from the training sample
    set $\{X_1, \ldots, X_{n'}\}$ with the corresponding target outputs $t_i$.
    for $i = 1$ to $n'$ do
      Calculate gradient estimate: $y \leftarrow \frac{1}{n'} \nabla_w \sum_i E(f(X_i; w); t_i)$;
    end for
    Apply update: $w \leftarrow w_k - \eta \cdot y$
  end while
```

Here, *learning rate* η is a small positive scalar, which is used to moderate the step size in the gradient descent search. E is the *cost function* or *error function*, which is essentially the difference between the target output of the network and output of the objective function. Notice that $\nabla_w E(\cdot)$ is a vector, and its components are the partial derivatives of E with respect to each of the w_j, where w_j is the component of weight vector w.

3.2. Double decryption mechanism

In a public-key encryption schemes with a double decryption mechanism, there exists two independent decryption algorithms. These two decryption algorithms are called user decryption algorithm (which has the general private key as input) and master entity decryption algorithm (which has the master private key as input), respectively. Taking the master private key as input, the master decryption procedure can decrypt any given ciphertext successfully. We formally define the BCP scheme as follows.

Definition 3.1 (*BCP Scheme* [7]). There are five algorithms in the BCP scheme, including setup algorithm *Setup*, key-generation algorithm *KeyGen*, encryption algorithm *Enc*, user decryption algorithm *Dec* and master decryption algorithm *mDec*, which is defined as $\mathcal{E} = \{Setup, KeyGen, Enc, Dec, mDec\}$.

- $(pp, mk) \leftarrow Setup(1^\kappa)$: given a security parameter κ, let p, q, p', q' be distinct odd primes and $p = 2p' + 1$ and $q = 2q' + 1$. Set the bit-length as $|N| = |pq| = \kappa$. For a multiplication group $\mathbb{Z}^*_{N^2}$, the algorithm *Setup* chooses a random element $g \in \mathbb{Z}^*_{N^2}$ of order $pp'qq'$ such that $g^{p'q'} \equiv 1 + kN \pmod{N^2}$ for $k \in \{1, 2, \cdot, N - 1\}$. After this step, the algorithm's outputs are

  public parameter $pp = (N, k, g)$ and

  master private key $mk = (p', q')$.
- $(pk, sk) \leftarrow KeyGen(pp)$: choose a random element $a \in \mathbb{Z}_{N^2}$ and compute the user's public key $h = g^a \bmod N^2$, where the user's private key is $sk = a$. Finally the algorithm outputs (pk, sk).
- $(A, B) \leftarrow Enc_{(pp,pk)}(m)$: given a message $m \in \mathbb{Z}_N$ and pick a random element $r \in \mathbb{Z}_{N^2}$, and outputs the ciphertext (A, B), where $A = g^r \bmod N^2$ and $B = h^r(1 + mN) \bmod N^2$.
- $m \leftarrow Dec_{(pp,sk)}(A, B)$: given a ciphertext (A, B) and private key $sk = a$, it returns the message m as

  $$m = \frac{\frac{B}{A^a} - 1 \bmod N^2}{N}$$

  or the special message "reject" if it is invalid ciphertext.
- $m \leftarrow mDec_{(pp,pk,mk)}(A, B)$: given a ciphertext (A, B), pubic key $pk = h$ and master private key mk, the user's private key $(sk = a)$ can be computed as

  $$a \bmod N = \frac{h^{p'q'} - 1 \bmod N^2}{N} \cdot k^{-1} \bmod N.$$

  In order to remove the random element $r \in \mathbb{Z}_{N^2}$, it is necessary to compute

  $$r \bmod N = \frac{A^{p'q'} - 1 \bmod N^2}{N} \cdot k^{-1} \bmod N$$

  then compute $\tau = ar \bmod N$. Finally, it returns the message m as

  $$m = \frac{\left(\frac{B}{g^\tau}\right)^{p'q'} - 1 \bmod N^2}{N} \cdot \zeta^{-1} \bmod N$$

  or the special message "reject" if it is invalid ciphertext, where k^{-1} and ζ^{-1} denote the inverse of $k \bmod N^2$ and $p'q' \bmod N^2$, respectively.

To simplify the notation, we use $Enc_{pk}(m)$ instead of $Enc_{pp,pk}(m)$ and use $Add(\cdot)$ to denote the addition-gate, i.e., $(\overline{A}, \overline{B}) := (A_1 \cdot A_2 \bmod N^2, B_1 \cdot B_2 \bmod N^2) = Add(C_1, C_2)$, where $\{C_i\}_{i=1}^2 = \{(A_i, B_i)\}_{i=1}^2$ is a ciphertext set.

3.3. Fully homomorphic encryption

Fully homomorphic encryption (FHE) allows one to compute the encrypted results of addition and multiplication of two plaintexts using the corresponding ciphertexts directly without decryption. Generally speaking, a FHE system $\mathcal{E}^{\mathsf{F}}$ consists of four algorithms: key generation algorithm $\mathsf{F.KeyGen}$, encryption algorithm $\mathsf{F.Enc}$, decryption algorithm $\mathsf{F.Dec}$, and evaluation algorithm $\mathsf{F.Eval}$. For this evaluation algorithm $\mathsf{F.Eval}$, given a circuit C, a public key pk_{F}, and any ciphertexts c_i is generated by $\mathsf{F.Enc}_{pk_{\mathsf{F}}}(m_i)$, outputs a refreshed ciphertext c^* such that $\mathsf{F.Dec}_{sk_{\mathsf{F}}}(c^*) = C(m_1, \ldots, m_n)$.

Suppose that circuit $D_{Add_{\mathsf{F}}}$ can handle addition gate, denoted by Add_{F}. If c_1 and c_2 are two ciphertexts that encrypt m_1 and m_2, respectively, under pk_{F}, then we can compute

$$c \leftarrow \mathsf{F.Eval}(pk_{\mathsf{F}}, D_{Add_{\mathsf{F}}}, c_1, c_2)$$

which is a ciphertext under pk_{F} of $m_1 + m_2$. Similarly, circuit $D_{Multi_{\mathsf{F}}}$ can handle multiplication gate, denoted as $Multi_{\mathsf{F}}$. For ciphertexts c_1 and c_2 as defined before,

$$c \leftarrow \mathsf{F.Eval}(pk_{\mathsf{F}}, D_{Multi_{\mathsf{F}}}, c_1, c_2)$$

is a ciphertext under pk_{F} of $m_1 \times m_2$. Assume $[m] = \mathsf{F.Enc}_{pk_{\mathsf{F}}}(m)$, then $[x] +_{\mathsf{F}} [y] = [x + y] = Add_{\mathsf{F}}([x], [y])$ and $[x] \times_{\mathsf{F}} [y] = [xy] = Multi_{\mathsf{F}}([x], [y])$ denote the addition and multiplication computation of FHE, respectively.

We formally define the multi-key fully homomorphic encryption [4,5] as follows.

Definition 3.2 (*Multi-Key Fully Homomorphic Encryption, MK-FHE*)**.** For arbitrary circuit class C, a family of encryption schemes $\{\mathcal{E}^n = (\mathsf{MF.KeyGen}, \mathsf{MF.Enc}, \mathsf{MF.Dec}, \mathsf{MF.Eval})\}_{n>0}$ is said to be a multi-key fully homomorphic encryption, if for every $n > 0$, $\mathcal{E}^n$ satisfies the following properties:

- $(pk_{\mathsf{MF}}, sk_{\mathsf{MF}}, ek_{\mathsf{MF}}) \leftarrow \mathsf{MF.KeyGen}(1^{\kappa})$: given a security parameter κ, outputs a public key pk_{MF}, a private key sk_{MF} and a public evaluation key ek_{MF}.
- $c \leftarrow \mathsf{MF.Enc}(pk_{\mathsf{MF}}, x)$: for a message x and public key pk_{MF}, this algorithm outputs a ciphertext c.
- $x' \leftarrow \mathsf{MF.Dec}(sk_{\mathsf{MF}1}, sk_{\mathsf{MF}2}, \dots, sk_{\mathsf{MF}n}, c)$: given a ciphertext c and n private keys $sk_{\mathsf{MF}1}, \dots, sk_{\mathsf{MF}n}$, this algorithm outputs a message x'.
- $c^* \leftarrow \mathsf{MF.Eval}((c_1, pk_{\mathsf{MF}1}, ek_{\mathsf{MF}1}), \dots, (c_m, pk_{\mathsf{MF}m}, ek_{\mathsf{MF}m}), C)$: taken any boolean circuit $C \in \mathsf{C}$, any valid m key pairs $(pk_{\mathsf{MF}1}, ek_{\mathsf{MF}1}), \dots, (pk_{\mathsf{MF}m}, ek_{\mathsf{MF}m})$, and any ciphertexts $c_1, \dots, c_m$ as input, this algorithm outputs a refreshed ciphertext c^*.

The scheme should satisfied the following properties: the correctness of decryption and compactness of ciphertexts. That is to say, for any circuit $C \in \mathsf{C}$, the support of $\mathsf{MF.KeyGen}(1^{\kappa})$: n key pairs $\{(pk'_{\mathsf{MF}t}, sk'_{\mathsf{MF}t}, ek'_{\mathsf{MF}t})\}_{t\in[1,n]}$ and its any subset of m key pairs $\{(pk_{\mathsf{MF}i}, sk_{\mathsf{MF}i}, ek_{\mathsf{MF}i})\}_{i\in[1,m]}$, and any valid ciphertext $c_i \leftarrow \mathsf{MF.Enc}(pk_{\mathsf{MF}i}, x_i)$.

$(i \in [1, m])$, the algorithm $\mathsf{MF.Eval}$ holds the properties:

Correctness of decryption: given a tuple of private key $sk_{\mathsf{MF}}{}'_1, \dots, sk_{\mathsf{MF}}{}'_n$, a refreshed ciphertext c^*, the correct decryption is

$$\mathsf{MF.Dec}(sk_{\mathsf{MF}}{}'_1, \dots, sk_{\mathsf{MF}}{}'_n, c^*) = C(x_1, \dots, x_m),$$

where $c^* \leftarrow \mathsf{MF.Eval}((c_1, pk_{\mathsf{MF}1}, ek_{\mathsf{MF}1}), \dots, (c_m, pk_{\mathsf{MF}m}, ek_{\mathsf{MF}m}), C)$.

Compactness of ciphertexts: let

$$c^* \leftarrow \mathsf{MF.Eval}((c_1, pk_{\mathsf{MF}1}, ek_{\mathsf{MF}1}), \dots, (c_m, pk_{\mathsf{MF}m}, ek_{\mathsf{MF}m}), C)$$

be a refreshed ciphertext, the size of c^* is independent of the parameter m and the size of C, i.e., there is a polynomial f holds $|c^*| \le f(\kappa, n)$, where $|c^*|$ is denoted as the size of c^*.

If $n = 1$, the Definition 3.2 is the standard definition of FHE scheme. We use Add_{MF} to denote the secure addition gate, which is given the ciphertext c_1 and c_2 of plaintext m_1 and m_2 under public key $pk_{\mathsf{MF}1}$ and $pk_{\mathsf{MF}2}$ respectively, the server calculates the sum as $c_1 +_{\mathsf{MF}} c_2 = Add_{\mathsf{MF}}(c_1, c_2)$. Similarly, $Multi_{\mathsf{MF}}$ denotes the secure multiplication gate, the server calculates the products as $c_1 \times_{\mathsf{MF}} c_2 = Multi_{\mathsf{MF}}(c_1, c_2)$.

4. System model

4.1. Multi-key privacy-preserving deep learning system

In a deep learning system, each data owner has a sensitive dataset DB_i in which local resources are fully administrated by the data owner. In our multi-key system, we consider n such data owners, denoted by $P_1, \dots, P_n$. Each data owner $P_i (i \in [1, n])$ has own pair of public and private keys (pk_i, sk_i), local sensitive dataset DB_i has l_i attributes $\{X_1^{(i)}, X_2^{(i)}, \dots, X_{l_i}^{(i)}\}$ and $DB_1 \cap DB_2 \cdots \cap DB_n = \Phi$, where $i \in [1, n]$. These data owners want to perform collaborative deep learning with the other data owners. Due to the computational ability is limited, data owners need to outsource the data to an untrusted cloud server for collaboratively deep learning. Before running a neural network, data owners jointly negotiate and set up the target vector $t = \{t_i\}_{i=1}^n$ and weight matrix $W^{(j)}$ in advance, where $t_i = (t_1^{(i)}, t_2^{(i)}, \dots, t_{l_i}^{(i)}), j = 1, 2$.

Because these datasets are highly sensitive, to preserve confidentiality of data, each data owner P_i $(i \in [1, n])$ has to encrypt his/hers data before uploading them to a cloud server. The cloud server will train a model over these encrypted datasets which are encrypted under *different public keys*. To realize a multi-key privacy-preserving deep learning system, we consider two schemes, i.e., *basic scheme* and *advance scheme*. Both schemes ensure the inputs, intermediate results generated during the learning process and final output are secure and no information will be leaked during the whole learning process. In addition, we assume that each data owner stays online with broadband access to the cloud server.

4.2. Adversary model

In this paper, we assume that the data owners and servers are *honest-but-curious*, sometimes called *semi-honest*. It means that all the entities (i.e., all the data owners and the servers) will honestly follow the protocol, and try to gather or discover information about the intermediate results of the learning process by observing the transcripts. And we also assume that all the data owners' dataset are sensitive and required to be fully protected against the cloud server. Based on this assumption, we consider three kinds of attacks (1) online attack by compromised active data owners, who aim to learn or infer sensitive information on the dataset from other data owners and cloud server; (2) online attack by a compromised active server, which is the cloud server can pretend as participants execute the learning process for every guess; (3) outside attack whose goal is to obtain private information from the data owners and cloud server.

In our work, we try to address the privacy-preserving problem of deep learning in cloud computing. The security and privacy of the data owners' sensitive data, model's outputs, and intermediate results should be protected during the deep learning process.

Security goals. There are two aspects of the security goals for data privacy and model privacy. The detailed security goals for these two notions are as follows.

- *Privacy of Data*. The data privacy should be kept secret even if a subset of data owners and cloud server are corrupted. In more details, no information about the data will be leaked to the adversary.
- *Privacy of training model*. The training model should be kept secret and can be only known by data owners. The adversary even with all the information of cloud server, is not able to get the information of the training model.

5. Multi-key privacy-preserving deep learning system

5.1. The basic scheme

In this subsection, we give a *basic scheme* (refer to Fig. 2) to realize a scenario that multiple data owners want to collaboratively learn the parameters $W^{(1)}, W^{(2)}$ with their partitioned data without leaking the information of their sensitive datasets.

Main idea. Generally speaking, SMC cannot handle the data encrypted with *different public keys*, and it can only deal with the ciphertext under the *same public key*. In our *basic scheme*, to preserve the data privacy when multiple parties are involved in deep learning model, we use MK-FHE [4] $\mathcal{E}^n$ to encrypt the data before uploading it to a cloud. Assume there are n data owners $P_1, \dots, P_n$, who hold their respect mutually disjoint, sensitive and vertically partitioned dataset $DB_1, \dots, DB_n$, and corresponding target vectors $t_1, \dots, t_n$. Each

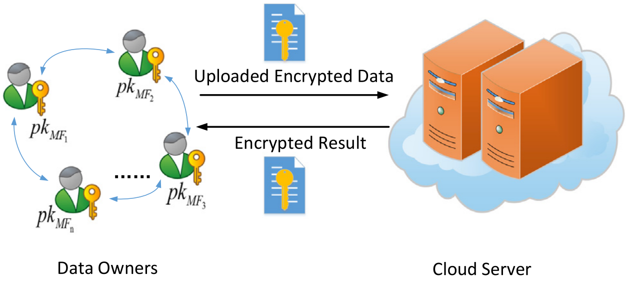


Fig. 2. The basic model.

data owner P_i ($i \in [1, n]$) encrypts his dataset with MK.Enc and uploads the ciphertexts MK.Enc(pk_{MF_i}, DB_i), MK.Enc($pk_{\mathsf{MF}_i}, W_i^{(j)}$), and MF.Enc($pk_{\mathsf{MF}_i}, t_i$) and its public key pk_{MF_i} to an untrusted cloud server $\mathcal{C}$ for secure deep learning.

Algorithm 3 Overall scheme of multi-key privacy-preserving deep learning based on MK-FHE

```
Input: $\{DB_1, DB_2, \ldots, DB_n\}$, initial $W^{(1)}, W^{(2)}$; $iteration_{max}$, Learning rate $\eta$
Output: $W^{(1)}, W^{(2)}$
Data owner $P_i (i \in [1, n])$ does:
    Initialize the parameters randomly;
    Sample a key tuple $(pk_{\mathsf{MF}_i}, sk_{\mathsf{MF}_i}, ek_{\mathsf{MF}_i}) \leftarrow \mathsf{MF.KeyGen}(1^\kappa)$;
    Each data owner encrypts data $DB_i, W_i^{(j)}$: $c_i \leftarrow \mathsf{MF.Enc}(pk_{\mathsf{MF}_i}, DB_i)$, $d_i \leftarrow \mathsf{MK.Enc}(pk_{\mathsf{MF}_i}, W_i^{(j)})$, $g_i \leftarrow \mathsf{MK.Enc}(pk_{\mathsf{MF}_i}, t_i)$;
    Upload $(pk_{\mathsf{MF}_i}, ek_{\mathsf{MF}_i}, c_i, d_i, g_i)$ to the cloud;
Cloud server dose:
    Execute Algorithm 1 over the ciphertext domain;
    Update the ciphertext of $W^{(j)}$;
    Send the learning results $\tau^*$ to the data owners;
Data owners $P_1, \cdots, P_n$ do:
    Data owners $P_1, \cdots, P_n$ jointly run a SMC protocol to calculate $\mathsf{MF.Dec}(sk_{\mathsf{MF}_1}, \cdots, sk_{\mathsf{MF}_n}, \tau^*)$;
```

To realize a multi-key privacy-preserving deep learning system, the concrete operation is shown in Algorithm 3. It is worth noting that the activation function $f(x) = \frac{1}{1+\exp(-x)} \in (0, 1)$ in Algorithm 1 is nonlinear. To support effective computation of the sigmoid function, we use Taylor series and approximate which is defined as:

$$\frac{1}{1+e^{(-x)}} = \frac{1}{2} + \frac{x}{4} - \frac{x^3}{48} + o(x^4). \tag{3}$$

According to the property of Taylor series, the number of terms in the expansion can be decided according to the accuracy requirement. The secure computation of the activation function Eq. (3) is listed in Algorithm 4.

5.2. *The advanced scheme*

In this subsection, we describe another more practical method for multi-key privacy-preserving deep learning system without interaction among data owners. These types of entities are involved in the *advanced scheme*, including a cloud server C, an authorized center $\mathcal{AU}$ and n data owners $P_1, P_2, \ldots, P_n$ (illustrated in Fig. 3).

Main idea. To decrease the computation/communication, we propose a hybrid multi-key deep learning training system model. By using a double decryption mechanism and FHE, the training process can be performed over ciphertexts under *different public key*. Assume each data owner has uploaded the ciphertext (encrypted under its own public key of BCP scheme) to cloud server $\mathcal{C}$. $\mathcal{C}$ will perform a neural network with $(\alpha - \beta - \gamma)$ configuration over multi-key encrypted domain. However, for any valid ciphertext, authorized center $\mathcal{AU}$ (holds master key mk) can decrypt it by using the second decryption algorithm $mDec(\cdot)$ of BCP scheme. Hence, $\mathcal{C}$ needs to blind the ciphertexts before sending them to the authorized center $\mathcal{AU}$. After receiving the blinded ciphertexts from the cloud $\mathcal{C}$, $\mathcal{AU}$ decrypts it by using the $mDec(\cdot)$ and re-encrypts the blinded plaintext by using FHE $\mathsf{F.Enc}_{pk_\mathsf{F}}(\cdot)$ of $\mathcal{E}^\mathsf{F}$. Finally, $\mathcal{AU}$ sends the fully homomorphic ciphertext to the cloud $\mathcal{C}$. After that, deep learning can be performed over the re-encrypted data. We assume $\mathcal{C}$ and $\mathcal{AU}$ are *semi-honest* and do not collude with each other.

Algorithm 4 Securely outsourcing computation of activation function

```
Input: Ciphertexts $[x]$, $[a_0]$, $[a_1]$, $[a_3]$, where $a_0$, $a_1$, $a_3$ are constants.
Output: $[y]$
Compute $c_0$ with $\mathsf{F.Enc}(pk_\mathsf{F})$, i.e., $c_0 = [a_0]$;
Compute $c_1$ with secure multiplication $Multi_\mathsf{F}$, i.e., $c_1 = [a_1] \times_\mathsf{F} [x]$;
Compute $c_3$ with secure multiplication $Multi_\mathsf{F}$, i.e., $c_3 = [a_3] \times_\mathsf{F} [x] \times_\mathsf{F} [x] \times_\mathsf{F} [x]$;
Compute $[y]$ with secure addition $Add_\mathsf{F}$, i.e., $[y] = c_0 +_\mathsf{F} c_1 +_\mathsf{F} c_3$;
return $[y]$;
```

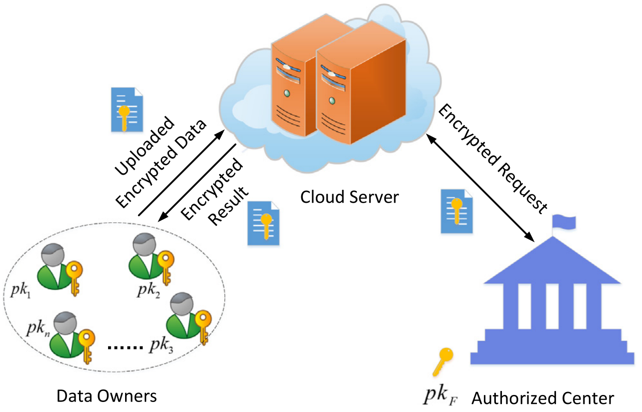


Fig. 3. The advanced model.

We explain the more details of the *advanced scheme* as follows.

Initialization. To protect the security and privacy of the data, each data owner encrypts his/her dataset before uploading to $\mathcal{C}$. We choose a hybrid scheme which is a combination of double decryption mechanism (BCP scheme) and FHE as our encryption technique. In the initialization process, $\mathcal{AU}$ sets up BCP scheme and FHE, uses *Setup* of BCP scheme to generate a public parameter $pp = (N, k, g)$ and a master key $mk = (p', q')$. Then $\mathcal{AU}$ sends pp to $\mathcal{C}$ while keeping mk. The initialized parameters $W^{(1)}, W^{(2)}$ and target vector $\{T_i\}_{i=1}^n$ should be jointly negotiated in advance by the data owners. Note that $W^{(1)} = (W_1^{(1)}, \ldots, W_n^{(1)})^\tau$ and $W_i^{(1)} = (w_{ij}^{(1)})_{I_i \times \gamma}$. $W^{(2)} = (W_1^{(2)}, \ldots, W_n^{(2)})^\tau$, $\sum_{i=1}^n I_i = \alpha$; $W_i^{(2)} = (w_{ij}^{(2)})_{J_i \times \beta}$, $\sum_{i=1}^n J_i = \gamma$; $T = \{T_i\}_{i=1}^n$ is target vector and $T_i = (t_1, \ldots, t_{l_i})$, I_i and J_i is the number of input and output of P_i ($i \in [1, n]$), respectively.

Data uploading. In this phase, data owner $P_i (i \in [1, n])$ uses the received public parameter $pp = (N, k, g)$ to generate it own pair of public and private keys $\{(pk_i, sk_i)\}_{i=1}^n$. Let us assume each data owner $P_i (i \in [1, n])$ has a set of vectors $DB_i = (X_1^{(i)}, X_2^{(i)} \cdots, X_{l_i}^{(i)})$.

Data owner P_i ($i \in [1, n]$) uses BCP scheme to encrypt its private dataset ($Enc_{pk_i}(DB_i) = \{Enc_{pk_i}(X_j^{(i)})\}_{j=1}^{l_i} = \{(A_j^{(i)}, B_j^{(i)})\}_{j=1}^{l_i}$) and weight parameter $Enc_{pk_i}(W_i^{(1)}), Enc_{pk_i}(W_i^{(2)})$.

After that, data owner P_i ($\imath \in [1, n]$) uploads the encrypted data, the target vector T_i and the public key pk_i to the cloud server $\mathcal{C}$. Here, we assume the communication channel for uploading is secure, which means nobody can obtain the uploaded data from $\mathcal{C}$. For simplicity, we use (A_i, B_i) to denote BCP ciphertext of the DB_i for the data owner P_i ($i \in [1, n]$).

Training. In this phase, we describe how the cloud server $\mathcal{C}$ handles α request from data owners in each learning round. After receiving the data encrypted with *different public keys*, cloud $\mathcal{C}$ needs to perform an α-input deep learning algorithm over encrypted domain $\{(Enc_{pk_1}(DB_1), Enc_{pk_1}(W_1^{(k)}), T_1), \dots, (Enc_{pk_n}(DB_n), Enc_{pk_n}(W_n^{(k)}), T_n)\}$, where $k = 1, 2$; $\alpha = \sum_{s=1}^{n} l_s$.

However, the learning algorithm cannot process the ciphertext under the *same public keys* directly. Therefore, cloud server $\mathcal{C}$ first runs Algorithm 5 with authorized center $\mathcal{AU}$ to transform the ciphertexts under *different public keys* into ciphertext under the *same public key*. Since $\mathcal{AU}$ holds the master key mk, it can decrypt any given valid ciphertext by using the master decryption of BCP scheme. Hence, $\mathcal{C}$ needs to blind ciphertexts $\{(A_i, B_i)\}_{i=1}^{n}$ with a random message $\{r_i\}_{i=1}^{n}$($r_i \in \mathbb{Z}_N^{l_i}$) before sending the ciphertexts to $\mathcal{AU}$. After receiving the blinded ciphertexts $\{(A_i', B_i')\}_{i=1}^{n}$, $\mathcal{AU}$ decrypts the blinded ciphertexts and re-encrypts the blinded plaintext z_i with FHE $\mathsf{F.Enc}_{pk_\mathsf{F}}(z_i)$ of $\mathcal{E}^\mathsf{F}$, which is denoted by Z_i, and sends this new ciphertext Z_i to $\mathcal{C}$. By removing the blinding factor r_i, $\mathcal{C}$ can get the ciphertext under the public key pk_F of $\mathsf{F.Enc}(\cdot)$ without knowing the underlying plaintext.

Algorithm 5 Transformation of BCP ciphertexts under $pk_1, pk_2, \cdots, pk_n$ to FHE ciphertexts under pk_F

```
Input: $((A_i, B_i), pk_i)$ for $i \in [1, n]$
Output: $(A_i'', B_i'')$
Cloud server $\mathcal{C}$ does:
choose randomness value $r_i \in \mathbb{Z}_N^{l_i}$, and compute the blinded ciphertexts $(A_i', B_i') \leftarrow Add((A_i, B_i), Enc_{pk_i}(r_i))$;
send these blinded ciphertexts and $pk_i$ to the authorized center $\mathcal{AU}$;
Authorized center $\mathcal{AU}$ does:
$z_i \leftarrow mDec_{pk_i, mk}(A_i', B_i')$ //$\mathcal{AU}$ holds the mk of BCP scheme;
$Z_i \leftarrow \mathsf{F.Enc}_{pk_\mathsf{F}}(z_i)$//$\mathcal{AU}$ encrypts $z_i$ with fully homomorphic encryption $\mathcal{E}^\mathsf{F}$;
send $Z_i$ to the authorized center $\mathcal{AU}$;
Cloud server $\mathcal{C}$ does:
$C_i' \leftarrow Add_\mathsf{F}((Z_i), \mathsf{F.Enc}_{pk_\mathsf{F}}(-r_i))$;
```

After performing the Algorithm 5, cloud server $\mathcal{C}$ holds the data encrypted with the *same public key* pk_F of $\mathsf{F.Enc}$. Later $\mathcal{C}$ can realize deep learning over the encrypted domain. We give some notations as follows: DB denotes the set of $\{DB_i\}_{i=1}^{n}$, $net^{(2)}$ and $net^{(3)}$ represents the input and output values of the hidden layer, $o^{(2)}$ and $o^{(3)}$ represents the activation and output layer values, respectively.

Secure feed forward: Cloud server $\mathcal{C}$ needs to compute the values of $net^{(2)}$, $net^{(3)}$, $o^{(2)}$ and $o^{(3)}$ over the encrypted domain. In the plaintext domain, we have

$$o^{(2)} = f(net^{(2)}) = f(W^{(1)} \cdot DB),$$
$$o^{(3)} = f(net^{(3)}) = f(W^{(2)} \cdot o^{(2)}),$$

then we use Algorithm 4 to compute the activation function over the encrypted domain as follows.

$$[o^{(2)}] = f([net^{(2)}]) = f([W^{(1)}] \times_\mathsf{F} [DB]),$$
$$[o^{(3)}] = f([net^{(3)}]) = f([W^{(2)}] \times_\mathsf{F} [o^{(2)}]).$$

Secure back propagation: The stochastic gradient descent (SGD) over the plaintext domain is described in Algorithm 2,

$$\delta_k^{(3)} = o_k^{(3)}(1 - o_k^{(3)})(t_k - o_k^{(3)}), \tag{4}$$

$$\delta_h^{(2)} = o_k^{(2)}(1 - o_k^{(2)}) \left(\sum_{k \in \mathcal{D}} W_{kh} \delta_k^{(3)} \right) \tag{5}$$

where $\delta_k^{(3)}$ and $\delta_h^{(2)}$ are the error terms for each network output unit k and for each hidden unit h, respectively. $\mathcal{D}$ denotes all of units whose immediate inputs including the output of unit h.

Finally, update each network weight $W^{(1)}$, $W^{(2)}$

$$W^{(2)} := W^{(2)} + \eta \delta_k^{(3)} x_{kh} \tag{6}$$

$$W^{(1)} := W^{(1)} + \eta \delta_h^{(2)} x_{hi}. \tag{7}$$

From the Eqs. (4) and (5) over plaintext domain, the ciphertext can be securely computed by Add_F and $Multi_\mathsf{F}$ as $[\delta_k^{(3)}] = [o_k^{(3)}] \times_\mathsf{F} ([1] +_\mathsf{F} [o_k^{(3)}]) \times_\mathsf{F} ([t_k] +_\mathsf{F} [o_k^{(3)}])$ and $[\delta_h^{(2)}] = [o_k^{(2)}] \times_\mathsf{F} ([1] +_\mathsf{F} [o_k^{(2)}]) \times_\mathsf{F} (\sum_{k \in \mathcal{D}} [W_{kh}] \times_\mathsf{F} [\delta_k^{(3)}])$ respectively. Since the ciphertexts $[x_{hi}]$, $[x_{kh}]$ and $[\eta]$ are known, $[\delta_k^{(3)}]$ and $[\delta_h^{(2)}]$ are securely computed, the Eqs. (6) and (7) can be performed over encrypted domain.

Algorithm 6 Transformation of the FHE ciphertext under pk_F to BCP ciphertext under different public keys

```
Input: A ciphertext $C$ under $pk_\mathsf{F}$ and $pk_1, pk_2, \cdots, pk_n$
Output: $\{(A_i, B_i)\}_{i=1}^{n}$
Cloud server $\mathcal{C}$ does:
randomly choose $r \in \mathbb{Z}_N$, and compute $D \leftarrow Add_\mathsf{F}(C, \mathsf{F.Enc}_{pk_\mathsf{F}}(r))$;
send blinded ciphertext $D$ to the authorized center $\mathcal{AU}$.;
Authorized center $\mathcal{AU}$ does:
$z \leftarrow \mathsf{F.Dec}_{sk_\mathsf{F}}(D)$;
$(Z_i, D_i) \leftarrow Enc_{pk_i}(z)$ for all $i \in [1, n]$;//Encrypting ciphertexts by using Enc of BCP scheme
send $(Z_i, D_i)$ to the cloud server $\mathcal{C}$;
Cloud server $\mathcal{C}$ does:
$(A_i, B_i) \leftarrow Add((Z_i, D_i), Enc_{pk_i}(-r))$;//Removing the blinding factor
```

Extraction. After running the deep learning model, cloud server $\mathcal{C}$ obtains a set of learning results $[\tau]$. Then $\mathcal{C}$ runs Algorithm 6 with $\mathcal{AU}$ to transform the data encrypted with pk_F into ciphertexts under the data owners' public key $pk_1, \dots, pk_n$. Finally, $\mathcal{C}$ sends these ciphertexts to the data owner $P_1, \dots, P_n$, who can decrypt this ciphertext with its private key sk_i.

6. Security analysis

Our system aims to achieve the privacy of input data, the intermediate results security and output results security of the deep learning under the *semi-honest* model, i.e., all participants in our two schemes are assumed to be *semi-honest*. Now, we describe the definition of the semantic security [41], i.e., security against polynomially indistinguishable chosen-plaintext attack (referred as IND-CPA security).

Definition 6.1 (*Semantic Security(SS), IND-CPA*). A public-key encryption scheme $\mathcal{E} = (KeyGen, Enc, Dec)$ is semantically secure, if for any stateful PPT adversary $A = (A_1, A_2)$, its advantage $\mathsf{Adv}_{\mathcal{E},A}^{SS}(k) := |Pr[\mathsf{Exp}_{\mathcal{E},A}^{SS}(k) = 1] - \frac{1}{2}|$ is negligible, where the experiment $\mathsf{Exp}_{\mathcal{E},A}^{SS}(k)$ is defined as follows:

```
$\mathsf{Exp}^{SS}_{\mathcal{E},A}(k)$:
    $b \leftarrow \{0, 1\}$
    $(pk, sk) \leftarrow KeyGen(1^k)$
    $(m_0, m_1) \leftarrow A_1(pk)$
    $c \leftarrow Enc_{pk}(m_b)$
    $b' \leftarrow A_2(c)$
    If $b' = b$, return 1;
    else, return 0
```

Here, we require that the two plaintexts have the same length. If they are not, padding message can be applied.

Privacy of Data. To protect the data privacy of the data owners, in *basic scheme* and *advanced scheme*, we use MK-FHE and hybrid encryption scheme to support the secure computation, respectively. According to the definition of semantic security, we can obtain the following conclusions.

Corollary 6.2 (*MK-FHE Semantic Security*). *If the underlying encryption scheme is semantically secure, then the multi-key fully homomorphic encryption is semantically secure.*

Proof (*Sketch*)**.** Let us assume the public key encryption scheme $\mathcal{E} = \{KeyGen, Enc, Dec\}$ is semantically secure. Based on this scheme $\mathcal{E}$, the challenger constructs a evaluate algorithm *Eval*, such that the new public key encryption scheme $\mathcal{E}' = \{KeyGen, Enc, Dec, Eval\}$ keeps homomorphic of addition and multiplication operations. If the evaluation key *ek* is public, then the adversary can compute *Eval* directly according to the public key *pk*, the ciphertext *c* and the evaluation key *ek*. Therefore, the MK-FHE scheme is semantically secure. □

Recall that in our *basic scheme*, data owners do not communicate with each other until the decryption phase. Each data owner $P_i (i \in [1, n])$ generates its own key tuple $(pk_{\mathsf{MF}_i}, sk_{\mathsf{MF}_i}, ek_{\mathsf{MF}_i})$ and encrypts its input DB_i under the public key pk_{MF_i} of MK-FHE. A *semi-honest* data owner *P* may collude with some data owners, and wants to reveal a sample vector *DB* uploaded by other data owners. However, data owners do not need to coin-flip for each other's random coins. From the Definition 3.2, it guarantees our *basic scheme* is secure against corrupt data owners. Therefore, the privacy of the data owners is confidential.

In our *advanced scheme*, because the BCP scheme and FHE are semantically secure (the detailed proof [7], Theorem 11), the cloud server $\mathcal{C}$ should be probabilistic polynomially bounded, and sends the blinded ciphertexts to $\mathcal{AU}$ for computation. The computing power of $\mathcal{AU}$, on the other hand, dose not have to be bounded, since it only receives the blinded ciphertexts and only be able to see the blinded messages, which are decrypted by master decryption algorithm of BCP scheme. Hence, the cloud server $\mathcal{C}$ and authorized center $\mathcal{AU}$ cannot obtain the learning results. Therefore, the privacy of the learning results is confidential and we obtain the following lemma.

Lemma 6.3. *Without any collusion, Algorithm* 5, 6 *is privacy-preserving for the weight* $W^{(1)}, W^{(2)}$.

Privacy of Training Model. A *semi-honest* cloud server $\mathcal{C}$ can train a deep learning model privately. Because Add_{F} and $Multi_{\mathsf{F}}$ are both semantically secure, there is no information leakage for $\mathcal{C}$. Hence, for the weights $W^{(1)}, W^{(2)}$ in the training process of *In the feed work stage* and *In the back propagation stage*, cloud server $\mathcal{C}$ performs Add_{F} and $Multi_{\mathsf{F}}$ operations, therefore privacy for the whole training process is guaranteed.

7. Application

In this section, we show an application of our *advanced scheme* in face recognition.

Privacy-Preserving Face Recognition. As a typical biometric authentication technique, face recognition is increasingly applied in real life. The widespread use of this technique arouse people's many privacy concerns, especially outsourcing computing in an untrusted cloud server.

Assume there exists an image sample set, which collect $n \times m$ grayscale images, n (e.g. $n = 20$) person $P_1, \ldots, P_n$ in various poses, and each person has m (e.g. $m = 30$) images (each image pixel $p_1 \times p_2, p_1 = p_2 = constant, n, m \ll p_1, p_2$ and $p_1, p_2 \in (0, 255)$). These m images include person's expression (such as sad, dismay, happy, angry), which direction they are watching (e.g. straight, right, left, up), and whether they are wearing mask or not. The main task is to learn a target function from this image sample set securely. Suppose we choose a network with a $(\alpha - \beta - \gamma)$ configuration, i.e., one input layer with α nodes, one hidden layer with β nodes and one output layer with γ nodes. The system includes three phases, including input encoding, learned hidden representation, and output encoding. The details are described as follows.

Input encoding. Before uploading the image to cloud server $\mathcal{C}$ for collaborative deep learning, $P_1, \ldots, P_n$ should preprocess the image data to feature extraction, then execute deep learning network with these features (e.g. edges) as input. Using this feature as input can reduce the number of input and corresponding weights, and cut down the computation and keep the classification correctly. Since the deep learning network has a fixed number of input units, then encoding the image has a fixed pixel intensity values, i.e., $n \times m$, which can be seen as the abstract expression of the original $p_1 \times p_2$ image. For example, P_i $(i \in [1, n])$ has m original images $\Gamma_1, \ldots, \Gamma_m$, each image $\Gamma_i (i \in [1, m])$ with $p_1 \times p_2$ pixel. After feature extraction, the original image is replaced by a feature set of $n \times m$ pixel, while each pixel value as one network input. However, in order to efficiently compute network of hidden and output activation functions, the input feature sample range space $(0, 255)$ should be converted to range space $(0, 1)$, which is represented as $(A_1, \ldots, A_s)$, where $A_i \in (0, 1)^{n \times m}, s < n$. Finally, each P_i $(i \in [1, n])$ runs **Data Uploading** protocol (which is described in Section 5.2), and uploads the ciphertexts $Enc_{pk_i}(A_i), Enc_{pk_i}(W^{(1)}), Enc_{pk_i}(W^{(2)})$ to the cloud server $\mathcal{C}$.

Learned Hidden Representations. Right now, cloud server $\mathcal{C}$ has obtained the data encrypted with *different public keys*, so cloud server $\mathcal{C}$ and authorized center $\mathcal{AU}$ run **Training** protocol (which is described in Section 5.2) to training a neural network securely.

Out encoding. Let us assume the deep learning network has four output units, and each output unit represents one of the four face directions. These four outputs can be viewed as a four-dimension vector, i.e., (*straight*, *right*, *left*, *up*) and we use four real numbers to represent the possibility of directions. If some component value is the highest, then the corresponding direction can be seen as the deep learning network prediction. For instance, (0.3, 0.3, 0.3, 0.7) is target output vector and 0.7 is the highest value, this result indicates the person is looking at his *up*. Since all the computation is performed in encrypted domain, the output results are also encrypted. After comparing these four components of the output vector over the encrypted domain, cloud server $\mathcal{C}$ chooses the highest one. Once the network output is given, $\mathcal{C}$ runs Algorithm 6 and sends the result encrypted with pk_i to person P_i, where $i \in [1, n]$.

8. Conclusions and future work

In this paper, we focused on the privacy issues of collaborative deep learning in cloud computing, and proposed two schemes, i.e., *basic scheme* and *advanced scheme*, to protect the privacy in deep learning. The *basic scheme* is based on a Mk-FHE scheme, and the *advanced scheme* is based on a hybrid structure, which combines the double decryption mechanism with FHE scheme. Both schemes are able to tackle the problem of privacy-preserving collaborative deep learning ciphertext with *different public keys*. Compared with the *basic scheme*, the *advanced scheme* does not need the interaction among the data owners during the decryption of the learning result. Our future work will be focused on two open problems: how to implement the FHE scheme in practical machine learning, and how to reduce the cost of computation and communication.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 61472091), Natural Science Foundation of Guangdong Province for Distinguished Young Scholars (2014A030306020), Science and Technology Planning Project of Guangdong Province, China (2015B010129015) and the Innovation Team Project of Guangdong Universities (No. 2015KCXTD014).

References

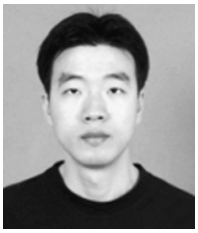
[1] V. Chang, Towards a big data system disaster recovery in a private cloud, Ad Hoc Networks 35 (2015) 65–82.
[2] Z.W. Wang, C. Cao, N.H. Yang, V. Chang, ABE with improved auxiliary input for big data security, J. Comput. System Sci. (2016).
[3] O. Goldreich, Secure multi-party computation. Manuscript. Preliminary version, 1998, pp. 86–97.
[4] A. López-Alt, E. Tromer, V. Vaikuntanathan, On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption, in: Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing, ACM, 2012, pp. 1219–1234.
[5] P. Mukherjee, D. Wichs, Two round multiparty computation via multi-key FHE, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer Berlin, Heidelberg, 2016, pp. 735–763.
[6] P. Mukherjee, P, D. Wichs, Two Round MPC from LWE via Multi-Key FHE. IACR Cryptology ePrint Archive, 2015, p. 345.
[7] 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, 2003, pp. 37–54.
[8] C. Gentry, Fully homomorphic encryption using ideal lattices, in: Symposium on the Theory of Computing, 2009.
[9] T.H. Chan, K. Jia, S. Gao, J. Liu, et al., PCANet: A simple deep learning baseline for image classification? IEEE Trans. Image Process. 24 (12) (2015) 5017–5032.
[10] A. Graves, A.R. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks. in: ICASSP, 2013.
[11] G. Hinton, L. Deng, D. Yu, et al., Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, Signal Process. Mag. 29 (6) (2012) 82–97.
[12] M. Liang, Z. Li, T. Chen, J. Zeng, Integrative data analysis of multi-platform cancer data with a multimodal deep learning approach, IEEE/ACM Trans. Comput. Biol. Bioinf. (TCBB) 12 (4) (2015) 928–937.
[13] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[14] V. Chang, M. Ramachandran, Towards achieving data security with the cloud computing adoption framework, IEEE Trans. Serv. Comput. 9 (1) (2016) 138–151.
[15] V. Chang, Y.H. Kuo, M. Ramachandran, Cloud computing adoption framework: A security framework for business clouds, Future Gener. Comput. Syst. 57 (2016) 24–41.
[16] G. Sun, Y. Xie, D. Liao, et al., User-defined privacy location-sharing system in mobile online social networks, J. Netw. Comput. Appl. (2016).
[17] G. Sun, D. Liao, H. Li, et al., L2P2: A location-label based approach for privacy preserving in LBS, Future Gener. Comput. Syst. (2016).
[18] J. Li, X.F. Chen, M.Q. Li, et al., Secure deduplication with efficient and reliable convergent key management, IEEE Trans. Parallel Distrib. Syst. 25 (6) (2014) 1615–1625.
[19] J. Li, Y.K. Li, X.F. Chen, et al., A hybrid cloud approach for secure authorized deduplication, IEEE Trans. Parallel Distrib. Syst. 26 (5) (2015) 1206–1216.
[20] J. Li, X.F. Chen, X.Y. Huang, et al., Secure distributed deduplication systems with improved reliability, IEEE Trans. Comput. 64 (12) (2015) 3569–3579.
[21] C.C. Aggarwal, S.Y. Philip, A general survey of privacy-preserving data mining models and algorithms, in: Privacy-Preserving Data Mining, Springer, US, 2008, pp. 11–52.
[22] S.Q. Ren, B.H.M. Tan, S. Sundaram, et al., Secure searching on cloud storage enhanced by homomorphic indexing, Future Gener. Comput. Syst. (2016).
[23] A. Evfimievski, T. Grandison, Privacy Preserving Data Mining, IGI Global, 2009, pp. 1–8.
[24] P. Vijayakumar, V. Chang, L.J. Deborah, et al., Computationally efficient privacy preserving anonymous mutual and batch authentication schemes for vehicular ad hoc networks, Future Gener. Comput. Syst. (2016).
[25] W. Du, Y. Han, S. Chen, Privacy-preserving multivariate statistical analysis: Linear regression and classification, in: SDM, 2004, Vol. 4, pp. 222–233.
[26] G. Jagannathan, R.N. Wright, Privacy-preserving distributed k-means clustering over arbitrarily partitioned data, in: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, ACM, 2005, pp. 593–599.
[27] T. Graepel, K. Lauter, M. Naehrig, ML confidential: Machine learning on encrypted data, in: Information Security and Cryptology, (ICISC), 2012, pp. 1–21.
[28] D. Agrawal, R. Srikant, Privacy-preserving data mining, in: Proc. ACM Conf. Manage. Data, 2000, pp. 439–450.
[29] N. Li, M. Lyu, D. Su, et al., Differential privacy: From theory to practice, Synth. Lect. Inf. Secur. Privacy Trust 8 (4) (2016) 1–138.
[30] T. Zhang, Q. Zhu, Dynamic differential privacy for ADMM-based distributed classification learning, IEEE Trans. Inf. Forensics Secur. 12 (1) (2017).
[31] M. Abadi, A. Chu, I. Goodfellow, et al., Deep learning with differential privacy, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2016, pp. 308–318.
[32] R. Shokri, V. Shmatikov, Privacy-preserving deep learning, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, ACM, 2015, pp. 1310–1321.
[33] T.T. Chen, S. Zhong, Privacy-preserving back-propagation neural network learning, IEEE Trans. Neural Netw. 20 (10) (2009) 1554–1564.
[34] 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.
[35] N. Schlitter, A protocol for privacy preserving neural network learning on horizontally partitioned data, in: PSD, 2008.
[36] J.W. Yuan, S.C. Yu, Privacy preserving back-propagation neural network learning made practical with cloud computing, IEEE Trans. Parallel Distrib. Syst. 25 (1) (2015) 212–221.
[37] D. Boneh, E.J. Goh, K. Nissim, Evaluating 2-dnf formulas on ciphertexts, in: Proceedings of the Second International Conference on Theory of Cryptography, TCC05, Berlin, Heidelberg, 2005, pp. 325–341.
[38] Q. Zhang, L.T. Yang, Z. Chen, Privacy preserving deep computation model on cloud for big data feature learning, IEEE Trans. Comput. 65 (5) (2016) 1351–1362.
[39] Z. Brakerski, C. Gentry, V. Vaikuntanathan, (Leveled) fully homomorphic encryption without bootstrapping, in: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ACM, 2012, pp. 309–325.
[40] A. Mirhoseini, A.R. Sadeghi, F. Koushanfar, CryptoML: Secure outsourcing of big data machine learning applications. 2016.
[41] S. Goldwasser, S. Micali, Probabilistic encryption, J. Comput. System Sci. 28 (2) (1984) 270–299.

Ping Li received the M.S. and Ph.D. degree in mathematics from Sun Yat-sen University in 2010 and 2016, respectively. Currently, she works at Guangzhou University as postdoctoral. And hers main research interest include cryptography, privacy-preserving and cloud computing.

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.

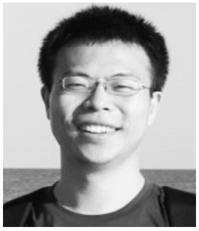
Zhengan Huang received his B.S. and M.S. degrees from Department of Mathematics, Sun Yat-sen University in 2009 and 2011, respectively, and his Ph.D. degree from Department of Computer Science and Engineering, Shanghai Jiao Tong University in 2015. He served as a security engineer in Huawei Technologies Co. Ltd. from 2015 to 2016. Currently, he is a postdoctoral researcher in Guangzhou University. His research interests include public-key cryptography and information security.

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.



Chong-Zhi Gao received his Ph.D. (2004) in applied mathematics from Sun Yat-sen University. Currently, he is a professor at the School of Computer Science of Guangzhou University. His research interests include cryptography and privacy in machine learning.

Siu-Ming Yiu received a B.S. in Computer Science from the Chinese University of Hong Kong, a M.S. in Computer and Information Science from Temple University, and a Ph.D. in Computer Science from The University of Hong Kong. Currently, he is a associate professor of the University of Hong Kong. His research interest include bioinformatics, computer security and cryptography.



Kai Chen received the B.S. degree from Nanjing University, China, in 2004, and the Ph.D. degree from University of Chinese Academy of Sciences in 2010. He is an Associate Professor in the Institute of Information Engineering, Chinese Academy of Sciences. He was also a postdoc at Pennsylvania State University, State College, PA USA. His research interests include software security, security testing on smartphones, and privacy protection in social networks.