Cryptanalysis and Improvement of DeepPAR: Privacy-Preserving and Asynchronous Deep Learning for Industrial IoT

Yange Chen, Suyu He, Baocang Wang*, Pu Duan, Benyu Zhang, Zhiyong Hong, Yuan Ping*

***Abstract*—Industrial Internet of Things (IIoT) is gradually changing the mode of traditional industries with the rapid development of big data. Besides, thanks to the development of deep learning, it can be used to extract useful knowledge from the large amount of data in the IIoT to help improve production and service quality. However, the lack of large-scale datasets will lead to low performance and overfitting of learning models. Therefore, federated deep learning with distributed datasets has been proposed. Nevertheless, the research has shown that federated learning can also leak the private data of participants. In IIoT, once the privacy of participants in some special application scenarios is leaked, it will directly affect national security and people's lives, such as smart power grid and smart medical care. At present, several privacy-preserving federated learning schemes have been proposed to preserve data privacy of participants, but security issues prevent them from being fully applied. In this paper, we analyze the security of the DeepPAR scheme proposed by Zhang *et al.* and point out that the scheme is insecure in the re-encryption key generation process, which will cause the leakage of the secret key of participants or the proxy server. In addition, the scheme is not resistant to collusion attacks between the parameter server and participants. Based on this, we propose an improved scheme. The security proof shows that the improved scheme solves the security problem of the original scheme and is resistant to collusion attacks. Finally, the security and accuracy of the scheme is illustrated by performance analysis.**

***Index Terms*—Privacy-preserving, asynchronous deep learning, homomorphic encryption, proxy re-encryption.**

I. Introduction

INDUSTRIAL Internet of Things (IIoT) [1], [2], [3] refers to the continuous integration of various types of sensors with sensing and monitoring functions into various parts of the industrial production process via the Internet. IIoT can dramatically increase manufacturing efficiency, improve product quality, reduce product costs and resource consumption, and ultimately achieve the goal of upgrading traditional industries to a new stage of intelligence. In order to better improve the quality of production and services, enterprises combine industrial production with the advanced technology available. In modern industrial production, industrial equipment obtains large amounts of data from end-users through the use of wireless sensors. Then, these data are sent to cloud servers for data mining to discover useful information. However, handling and utilizing the ever-increasing amount of data effectively is a huge challenge for the IIoT. Fortunately, the emergence of deep learning [4] can effectively extract valuable information from massive amounts of data.

With the development of deep learning techniques, more and more IIoT applications are using deep learning for data analysis. Deep learning can be used to provide intelligent decisions for industrial production and services by training a large amount of data to obtain a model that solves the relevant problem. As we well known, there are two main problems with the application of deep learning: on the one hand, it is difficult to train and maintain models for users with limited resources; on the other hand, the lack of training dataset can reduce the accuracy of the model. Therefore, researchers have proposed federated deep learning [5] as an effective way to solve the above problems. In federated deep learning, the parameter server coordinates participants with parameters to work together on the training of the model. As a branch of joint deep learning, each processing unit of the parameter server in asynchronous deep learning can update the model parameters independently [6]. Therefore, using asynchronous deep learning for model training can reduce training time.

Recently, Phong *et al.* [7] designed a deep learning scheme to protect the privacy of participants' data using the Paillier and LWE homomorphic encryption algorithms. However, there are still security issues with their scheme, where participants use the same public and private keys and malicious participants can easily access the privacy of other participants. Zhang *et al.* [8] designed a new privacy-preserving asynchronous deep learning scheme based on proxy re-encryption that addressed the security issues in Phong *et al.*'s scheme. However, we found that the scheme of Zhang *et al.* could leak the secret

This work is supported by the National Natural Science Foundation of China under Grant No. U19B2021, the Key Research and Development Program of Shaanxi under Grant No. 2020ZDLGY08-04, the Key Technologies R & D Program of He'nan Province under Grant No. 212102210084, and the Innovation Scientists and Technicians Troop Construction Projects of Henan Province. (Corresponding author: B. Wang; Y. Ping.)

Y. Chen is with the School of Telecommunications Engineering, Xidian University, Xi'an, 710071, China; and the School of Information Engineering, Xuchang University, Xuchang, 461000, China (e-mail: ygchen428@163.com)

S. He is with the Shanghai Jiyin Network Technology Co., Ltd., Shanghai, 200000, China (e-mail: hesuyu0911@163.com)

B. Wang* is with the State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, 710071, China; Cryptographic Research Center, Xidian University, Xi'an, 710071, China and School of Information Engineering, Xuchang University, Xuchang, 461000, China (e-mail: bcwang@xidian.edu.cn)

P. Duan and B. Zhang are with the Secure Collaborative Intelligence Laboratory, Ant Group, Hangzhou, 310000, China (e-mail: p.duan@antgroup.com; benyu.z@antgroup.com)

Z. Hong is with the Facility of Intelligence Manufacture, Wuyi University, Jiangmen, 529020, China; Yue-Gang-Ao Industrial Big Data Collaborative Innovation Center, Wuyi University, Jiangmen, 529020, China (e-mail: hzy_wyu@163.com)

Y. Ping* is with the School of Information Engineering, Xuchang University, Xuchang, 461000, China (e-mail: pingyuan@bupt.edu.cn)

key information of the participants and the proxy server, and could not resist collusion attacks between the parameter server and participants.

In this paper, we will demonstrate that the re-encryption key generation process of Zhang *et al.*'s DeepPAR scheme suffers from security problems. An attacker can easily obtain the secret keys of the participants and the proxy server, and then acquire the private data of the participants. In addition, the scheme cannot resist collusion attacks by the parameter server and the participants. Therefore, we have improved their scheme and successfully solved the security problems of the original scheme. Specifically, our main contributions are summarized below.

1) Firstly, we show that the scheme of Zhang *et al.* suffers from security problems due to the disclosure of secret key information and is not resistant to collusion attacks between the parameter server and the participants.
2) Secondly, we redesign the re-encryption key generation method of Zhang *et al.*'s scheme so that the secret keys of the participants and the proxy server are not leaked during the generation of the re-encryption key. It also ensures that the secret key of the proxy server cannot be obtained even in the case of collusion between the parameter server and a participant, perfectly circumventing the privacy leakage problem that exists in the original scheme.
3) Finally, our security analysis and experimental evaluation demonstrate that our scheme is more secure than the DeepPAR scheme of Zhang *et al.* without loss of model training efficiency and accuracy.

Organization. The rest of the paper is organized as follows. We discuss the related work on privacy-preserving deep learning in Section II and introduce the background knowledge required for this paper in Section III. We then give the system model, threat model, and security requirements in Section IV. In Section V, we give a description and security analysis of Zhang *et al.*'s scheme. We present our improved scheme and the corresponding security analysis in Section VI. In Section VII, we provide the communication overhead, computational cost and the experimental performance of the scheme. Finally, we draw conclusions in Section VIII.

II. Related Work

In the last decade, the research of privacy-preserving deep learning has become a hot topic for scholars. Many privacy-preserving deep learning schemes have been born [9], [10], [11]. McMahan *et al.* [12] presented a practical federated learning model of deep networks based on iterative model averaging from decentralized data, which can jointly train the model without sending user data to an aggregation server and realize the purpose of federated training and prediction. This is a common federated averaging algorithm provided by Google. To better describe the technologies, we divide these works into two categories: one is the protection of data through data perturbation, and the other is the protection of data through cryptographic tools.

Differential privacy [13], [14], [15] has been widely used as a mainstream data perturbation mechanism for privacy-preserving deep learning. In 2015, Shokri *et al.* [6] proposed a distributed deep learning scheme with participant privacy in mind. Unlike traditional deep learning training where participants upload data to a parameter server, in their scheme, participants train independently with their own private datasets in local. After training, participants simply upload the gradients obtained from training to the parameter server. Their solution protects the privacy of the participants to a certain extent. Phan *et al.* [16] proposed a privacy-preserving mechanism that can dynamically add noise based on the contribution of features to the output in a deep learning network, which can reduce the overhead of participants. Zhang *et al.* [17] designed two privacy-preserving distributed learning schemes using differential privacy. Gong *et al.* [18] proposed a privacy-enhanced multi-party deep learning framework, which dynamically allocated privacy budgets at different stages of training to further improve security without compromising the accuracy of model training. Miran *et al.* [19] combined homomorphic encryption and differential privacy to design a privacy-preserving deep learning scheme, and their scheme can be effectively applied to deep learning. Abida *et al.* [20] a deep learning framework with differential privacy by adding Laplacian noise to gradients. However, Xiang *et al.* [21] pointed out that adding noise to the gradient ensured privacy, but it greatly reduced the accuracy of the model. In addition, they presented a differentially-private deep learning scheme from an optimization perspective.

The common cryptographic tools in deep learning are secure multiparty computation (SMC) [22–24] and homomorphic encryption [25].

SMC performs secure computing jointly between some untrustworthy participants and is often used to multi-party deep learning, but SMC has high communication complexity owing to high interaction. Bonawitz *et al.* [26] proposed a secure data aggregation method using secret sharing techniques. However, the communication overhead for users in the scheme is too high. Xu *et al.* [27] proposed an efficient privacy-preserving federated learning approach (HybridAlpha), which employed an SMC protocol based on functional encryption. In 2017, Mohassel *et al.* [28] proposed a secure two-party computation, which can support the computation of non-linear functions and can be applied to common deep learning models. Bansal *et al.* [29] constructed a two-party protocol using secret sharing and secure scalars that can protect privacy during deep learning training process. However, when the number of parties is large enough, the scheme no longer works.

Homomorphic encryption [30], [31] allows arithmetic operations to be performed under ciphertext. The decrypted result calculated on ciphertext is consistent with the result calculated on plaintext, which can effectively protect data privacy in deep learning. In 2018, Phong *et al.* [7] pointed out privacy leakage issues in the scheme of Shokri *et al.* [6], which demonstrated that even uploading gradient information without uploading local data also leaded to participant privacy leakage. Therefore, they used homomorphic encryption to encrypt the participants' gradients and then uploaded them to the parameter server,

which used homomorphic properties to update the model. Due to semantical security of homomorphic encryption schemes, the adversary cannot directly access the gradient information. However, all participants in the scheme use the same homomorphic encryption key, resulting in private information of other participants insecure. To overcome this drawback, Zhang *et al.* [8] designed DeepPAR, a privacy-preserving deep learning scheme based on proxy re-encryption. all participants in the scheme had different public and secret keys, and participants encrypted the gradients using their own public keys and uploaded them to the parameter server. The parameter server used proxy re-encryption to convert the ciphertext into another ciphertext by the proxy public key for aggregation. This effectively prevented the privacy of the participants from being compromised. However, we found that the scheme was not resistant to collusion attacks and that the generation process of the proxy re-encryption key was insecure, leading to the disclosure of secret key information of the proxy and the participants. Tang *et al.* [32] proposed a distributed deep learning scheme with security guarantees and high accuracy. In this scheme, a key transformation server was used to re-encrypt the encrypted gradients. At the same time, the data service provider performed homomorphic aggregation of the re-encrypted gradients and computed updates to the ciphertext. Their scheme solved the problem of collusion between the cloud server and any learning participants in distributed deep learning schemes, but its communication overhead is too high. Ma *et al.* [33] first introduced the concept of verifiability in privacy-preserving deep learning, but there is a risk of server and participants complicity.

III. Preliminaries

The main notations in this paper are shown in Table I.

TABLE I: Notations

Notation	Description
$\mathbb{Z}_p^*$	Multiplicative cyclic group of invertible integers modulo p
$\mathbb{Z}_{p'}^*$	Multiplicative cyclic group of invertible integers modulo p'
$\mathbb{G}$	Cyclic group
p, p'	Large primes
h, g	Random generators
θ	The secret key of ElGamal
m	Plaintext
c_1, c_2	Ciphertext of ElGamal
a, b	The secret key and public key of proxy re-encryption
w	Weight vector
η	Gradient
α	Learning rate
G	Gradient vector
x_i	The secret key of participants
rk_i	The re-encryption key
$sk(x_o)$	The secret key of the proxy server
pk	The public key of the proxy server
n	The number of participants
M	The number of gradients

A. ElGamal Cryptography

Homomorphic encryption is a special type of encryption that allows us to perform homomorphic arithmetic operations on ciphertext without decrypting the ciphertext. The result of the homomorphic operation is the same as the result of the plaintext operation when the secret key is used to decrypt the ciphertext. A common homomorphic encryption scheme is the ElGamal encryption scheme [34]. ElGamal cryptosystem is composed of three algorithms: **KeyGen**, **Enc** and **Dec**.

- **KeyGen:** Given large primes p and p' satisfying $p = 2p' + 1$, choose a random generator $h \in \mathbb{Z}_p^*$ such that $g = h^2 \bmod p$. Then choose random number $\theta \in \mathbb{Z}_{p'}^*$ and compute $\gamma = g^\theta$. Let the public key be $PK = (p, g, \gamma)$ and the secret key be $SK = \theta$.
- **Enc:** Given a random number $r \in \mathbb{Z}_{p'}^*$, the cryptographer encrypts the plaintext m that he wants to encrypt, computing the ciphertext $c_1 = g^m \gamma^r$, $c_2 = g^r$. Finally the ciphertext $C = (c_1, c_2)$ of the plaintext m is obtained.
- **Dec:** Given a ciphertext $C = (c_1, c_2)$ and a secret key $SK = \theta$, decrypt to obtain the plaintext m:

$$g^m = c_1 / c_2^\theta. \tag{1}$$

If the message m is relatively small, the message m can be acquired efficiently by using Pollard's Rho algorithm.

Homomorphic nature: ElGamal encryption has an additive homomorphic property, which allows addition to be performed on a set of plaintexts without decrypting their ciphertexts. Given two ciphertexts C and $\hat{C}$,

$$\begin{aligned} C &= (c_1, c_2) = (g^{m_1} \cdot \gamma^{r_1}, g^{r_1}), \\ \hat{C} &= (\hat{c}_1, \hat{c}_2) = (g^{m_2} \cdot \gamma^{r_2}, g^{r_2}), \end{aligned} \tag{2}$$

additive homomorphic operations are done on the ciphertext of m_1 and m_2 as follows.

$$\begin{aligned} C \otimes \hat{C} &= (c_1 \otimes \hat{c}_1, c_2 \otimes \hat{c}_2) \\ &= \left(g^{m_1+m_2} \cdot \gamma^{r_1+r_2}, g^{r_1+r_2}\right). \end{aligned} \tag{3}$$

B. Proxy Re-encryption

In this subsection, we briefly describe the proxy re-encryption scheme based on ElGamal encryption [35]. Given a multiplicative cyclic group $\mathbb{G}$. The proxy re-encryption scheme consists of five algorithms (**KeyGen**, **RKGen**, **Enc**, **Dec**, **ReEnc**).

- **KeyGen** $\left(1^\lambda\right) \rightarrow (sk_a, pk_a)$: Given the security parameter λ, the user generates the secret key $sk_a = a \xleftarrow{R} \mathbb{Z}_p^*$ and the public key $pk_a = g^a \in \mathbb{G}$.
- **RKGen** $(sk_a, sk_b) \rightarrow rk_{a \rightarrow b}$: Given two secret keys sk_a and $sk_b = b$, generate a re-encryption key $rk_{a \rightarrow b} = sk_a^{-1} \cdot sk_b$ by the re-encryption algorithm.
- **Enc** $(pk_a, m) \rightarrow C_a$: Given the plaintext m and the public key pk_a, the user chooses a random integer r and then computes the ciphertext $C_a = \left(g^m \cdot g^r, g^{sk_a r}\right)$ using the ElGamal encryption algorithm.

- **Dec** $(sk_a, C_a) \rightarrow m$: After obtaining the ciphertext C_a, the user decrypts to obtain the original plaintext m using the secret key sk_a according to the ElGamal decryption algorithm.
- **ReEnc** $(C_a, rk_{a\rightarrow b}) \rightarrow C_b$: After getting the ciphertext C_a, the user uses the re-encryption key $rk_{a\rightarrow b}$ to transform the ciphertext into another ciphertext $C_b = \left(g^m \cdot g^r, \left(g^{sk_a r}\right)^{rk_{a\rightarrow b}}\right) = \left(g^m \cdot g^r, g^{sk_b r}\right)$.

C. Asynchronous Deep Learning

Deep learning is a special type of machine learning algorithm, a branch of machine learning. The aim of deep learning is to extract complex features from high-dimensional data and use these features to build a model that relates input to output. In deep learning model, each neuron node (except the bias node) is associated with an activation function. Examples of activation functions are hyperbolic tangent $f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$, sigmoid $f(z) = (1 + e^{-z})^{-1}$, and rectifier $f(z) = max(0, z)$.

Stochastic gradient descent (SGD). The stochastic gradient descent algorithm is an optimization algorithm for the gradient descent algorithm [36]. Simply speaking, instead of using all available data for training, it selects small batches of data to compute the gradients in the network. In the most extreme cases, we can even choose a random sample of data for the gradient descent algorithm at each training session, which corresponds to maximum randomness.

We denote the weight vector as $\boldsymbol{w}$. There are a total of d weights in the vector, denoted as $\boldsymbol{w} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_d) \in \mathbb{R}^d$. Let E be the loss function, i.e. the difference between the true value of the objective function and the output computed by the deep learning model. Calculate the gradient η of E for each parameter in $\boldsymbol{w}$ by the following equation.

$$\eta = \left(\frac{\delta E}{\delta \boldsymbol{w}_1}, \frac{\delta E}{\delta \boldsymbol{w}_2}, \ldots, \frac{\delta E}{\delta \boldsymbol{w}_d}\right), \tag{4}$$

where δ is the partial derivative. For asynchronous stochastic gradient descent (ASGD), each participant is trained using their own training dataset to obtain the gradient. Since ASGD is based on the idea of parallel computing, participants divide the gradient vector η into N parts that can be uploaded to different processing units of the parameter server. Thus, the update rule can be expressed as $\boldsymbol{w}^{(i)} := \boldsymbol{w}^{(i)} - \alpha \cdot \eta^{(i)} (1 \leq i \leq N)$, where α is the learning rate.

IV. System Model and Security Requirements

A. System model

As shown in Fig. 1, three parties are involved in the scenario used for asynchronous deep learning, including a parameter server, a proxy server and participants. The descriptions of these parties are given below.

1) **Parameter server**: The parameter server has powerful storage and computational capabilities. The parameter server is responsible for collecting the encrypted gradient vectors and converting the ciphertext gradient vectors into another set of ciphertext with the same public key of the proxy server, and then updating the global weight parameters using the encrypted gradients. It is equipped with N processors capable of independently and simultaneously updating the corresponding part of the global weight parameters.
2) **Proxy server**: The main task of the proxy server is to participate in generating the re-encryption key and decrypting the blinded ciphertext weight parameters and sending them to the participants.
3) **Participants**: n participants collect the data from sensors or IoT devices and form their own private dataset and a replica of the global model. They then work together to train a model with higher accuracy. Each participant trains the model utilizing their own dataset in local to obtain a new group of gradients. Then, they encrypt the gradients using their own public keys and send them to the parameter server. In addition, the participants are involved in the generation of the proxy re-encryption key.

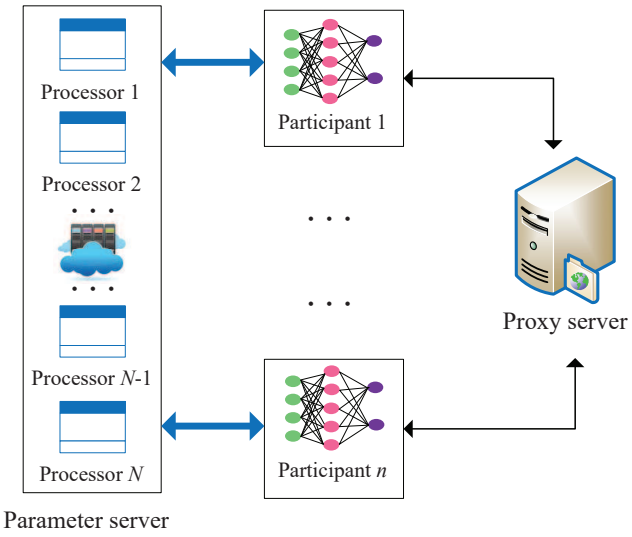


Fig. 1: System model

B. Threat model

In Zhang *et al.*'s [8] scheme, they assume that the parameter server and proxy server are honest-but-curious. However, the scheme cannot against collusion attacks between the parameter server and participants. Moreover, in the real application, the parameter server, proxy server, and participants are likely to be compromised by external adversaries or become a malicious adversary $\mathcal{A}$. Therefore, for the problem of the scheme, we give a formal definition of the threat model for the malicious adversaries through the real/ideal world model. In the real/ideal world model, if any adversary in the real world can be simulated by a simulator adversary in the ideal world, the protocol for the specific attack is secure. The framework is as follows:

The real model: The protocol has three parties-participant P_i, parameter server $\mathcal{S}$ and proxy server $\mathcal{O}$. All parties can access the public parameters such as the public keys and protocol $\Pi : (X, Y) \rightarrow (X \cap Y, X \cap Y)$. Participant P_i and

the proxy server $\mathcal{O}$ have the inputs X, Y respectively, and the parameter server $\mathcal{S}$ has an input $\{\circ, \perp\}$, where $\circ$ represents the null or rejecting to participant in the protocol, and $\perp$ expresses the interrupt. The adversary $\mathcal{A}$ can interact with any parties in the system model. After performing the protocol, the joint execution of P_i, $\mathcal{S}$, $\mathcal{O}$, $\mathcal{A}$ in the real world is marked as $\text{REAL}_{\Pi,\mathcal{A}}(X, Y)$.

The ideal model: In the ideal world, a trusted third party T is introduced to perform the output of the ideal functionality f and parties $\bar{P}_i$, $\bar{\mathcal{S}}$, $\bar{\mathcal{O}}$. Besides, parties $\bar{P}_i$, $\bar{\mathcal{O}}$ have the inputs $\bar{X}$, $\bar{Y}$ respectively, and the party $\bar{\mathcal{S}}$ has an input $\{\circ, \perp\}$. The appointment is followed:

(1) If $\bar{P}_i$, $\bar{\mathcal{S}}$, $\bar{\mathcal{O}}$ are honest, $\bar{P}_i$ and $\bar{\mathcal{O}}$ send their inputs to T, and $\bar{\mathcal{S}}$ sends $\circ$ to T. The simulator adversary SIM replaces the parties to send the inputs and obtains T's response.

(2) If $\bar{P}_i$ ($\bar{\mathcal{O}}$) sends $\bar{X}(\bar{Y})$ or $\perp$ to T respectively, then $\bar{\mathcal{S}}$ sends $d = \{\circ, \perp\} \cup X_{\bar{P}_i}(Y_{\bar{\mathcal{O}}})$, where $X_{\bar{P}_i}$ and $Y_{\bar{\mathcal{O}}}$ are any sets. In the process, if a party is malicious, $\bar{X}$ and $\bar{Y}$ may be different from X and Y.

(3) If $d = \{\circ, \perp\}$ and T has received $\bar{X}(\bar{Y}) \neq \perp$ or $\perp$, then T sends $\bar{X} \cap \bar{Y}$ or $\perp$ to $\bar{P}_i$ ($\bar{\mathcal{O}}$); If $d \neq \circ$, the party T sends d to $\bar{P}_i$ ($\bar{\mathcal{O}}$).

In the ideal world, the joint execution of $\bar{P}_i$, $\bar{\mathcal{S}}$, $\bar{\mathcal{O}}$, SIM is marked as $\text{IDEAL}_{f,SIM}(X, Y)$.

Simulatability: In the real/ideal model, the protocol Π is called as the security computation functionality f in the malicious model. If the real world adversary $\mathcal{A}$ exists an ideal world adversary SIM, causing the outputs in the real world and the ideal world are computationally indistinguishable:

$$\text{IDEAL}_{f,SIM}(X, Y) \overset{c}{\approx} \text{REAL}_{\Pi,\mathcal{A}}(X, Y).$$

C. Security Requirements

Our solution ensures system security even under a malicious adversary model. In real life, cloud servers may actively deviate from the protocol and try to compromise the privacy of participants, and may deceive participants with false results in order to save on computational costs or storage resources. In our scenario, even if a participant colludes with the server, there is no way to access the privacy information of other participants. We assume that parameter server and proxy server do not collude, which is feasible in the real world because major cloud service providers will guarantee honest and reliable services for their own reputation and credibility. We present the specific security requirements for our scheme.

1) **Input privacy:** Since the gradients can reveal the privacy of the participants, when the participants have trained with local data to obtain the gradients, the participants encrypt the gradients and send them to the parameter server. Due to the protection of the encryption scheme, the adversary cannot obtain the plaintext value of the ciphertext gradients. Furthormore, the parameter server cannot access the private information of other participants even if it colludes with some participants.
2) **Model privacy:** The model trained by the parameter server and the participants is their private property and the proxy server is only responsible for decrypting the blinded weight parameters and therefore cannot access the model parameters. Besides, other participants who are not authorized to do so cannot access the model parameters. Therefore our solution needs to satisfy the privacy of the model and ensure that the model parameters are not accessible to illegal users.

V. Description and Security Analysis of the DeepPAR Scheme Proposed by Zhang *et al.*

A. Zhang et al.'s DeepPAR Scheme

In this section, we first briefly introduce the DeepPAR scheme proposed by Zhang *et al.* [8]. Then the security issues are analyzed. Their scheme is described in detail as follows.

1) **KeyGen**: Each participant P_i randomly picks a number $x_i \in \mathbb{Z}_{p'}^*$ as its secret key sk_i and computes its public key $pk_i = g^{x_i}$, denoted as (pk_i, sk_i), where $i \in [1, n]$. Similarly, the proxy server randomly chooses a number $x_o \in \mathbb{Z}_{p'}^*$ as its secret key sk and computes its public key $pk = g^{x_o}$, denoted as (pk, sk). Finally, the parameter server, proxy server and participants generate the re-encryption key rk_i following the steps below.
   a) The parameter server $\mathcal{S}$ chooses a random number $r_i \in \mathbb{Z}_{p'}^*$ and sends it to the participant $\mathcal{P}_i$.
   b) $\mathcal{P}_i$ calculates $x_i^{-1} \cdot r_i$ and sends it to the proxy server $\mathcal{O}$.
   c) $\mathcal{O}$ uses its secret key x_o to compute $x_i^{-1} \cdot r_i \cdot x_o$ and then sends it to $\mathcal{S}$.
   d) After receiving $x_i^{-1} \cdot r_i \cdot x_o$, $\mathcal{S}$ obtains the re-encryption key $rk_i = x_i^{-1} \cdot x_o$ for the ith participant by multiplying by r_i^{-1}.
2) **Compute**: Each participant P_i downloads the weight parameters stored in the processing unit PU_j of the parameter server and then decrypts them, with the specific decryption process described in the **Decrypt** phase, which yields the current weights $\boldsymbol{w}_{\text{global}}$, where $j \in I_i^{(\text{down})} \subset [1, N]$, and $I_i^{(\text{down})} = [1, N]$ means to download all encrypted parts of the global weight. Then, a small batch dataset is selected and deep learning training is performed to obtain a new gradient vector $G = \left(G^{(1)}, \ldots, G^{(N)}\right)$. Finally, the participant encrypts the gradient to obtain $E_{pk_i}\left(-\alpha \cdot G^{(d)}\right)$, where $d \in I_i^{(\text{up})} \subset [1, N]$ and sends it to the corresponding processing unit PU_j of the parameter server.
3) **Re-encrypt**: When the parameter server $\mathcal{S}$ receives the gradient $E_{pk_i}\left(-\alpha \cdot G^{(d)}\right)$ encrypted by participant $\mathcal{P}_i$ using his public key pk_i, $\mathcal{S}$ converts the ciphertext into another ciphertext $E_{pk}\left(-\alpha \cdot G^{(d)}\right)$ under the same public key pk of the $\mathcal{O}$. Let the gradient ciphertext be $(c_1, c_2) = (g^{\Delta w} \cdot g^{r_i}, g^{x_i r_i})$ and the transformation process is as follows:
$$c_2^{rk_i} = (g^{x_i r_i})^{x_i^{-1} \cdot x_o} = g^{x_o r_i}. \tag{5}$$
4) **Aggregate**: When each processing unit PU_j of $\mathcal{S}$ has completed the re-encryption of the received gradient vectors, it aggregates all the gradient vector ciphertexts to obtain $E_{pk}\left(\boldsymbol{w}_{\text{global}}^{(d)}\right) = E_{pk}\left(\boldsymbol{w}_{\text{global}}^{(d)}\right) +$

$E_{pk}\left(-\alpha \cdot G^{(d)}\right)$, where w_{global} is the global weights, and $d \in I_i^{(\text{up})} \subset [1, N]$.

5) **Decrypt**: When the $\mathcal{S}$ completes the aggregation, each participant downloads the aggregation result, but at this point the ciphertext is encrypted using the public key of the proxy server, and the participant cannot decrypt it, so it can only be sent to the proxy server for decryption. To protect the value of encrypted global weight parameters $E_{pk}\left(\boldsymbol{w}_{\text{global}}\right)$, P_i selects the blind factor $s_i \in \mathbb{Z}_p$ and calculates $E_{pk}\left(\boldsymbol{w}_{\text{global}} + \boldsymbol{s}_i\right)$, and then sends them to the proxy server for decryption. The proxy server decrypts them and sends $\boldsymbol{w}_{\text{global}} + \boldsymbol{s}_i$ to the participant. The participant receives $\boldsymbol{w}_{\text{global}} + \boldsymbol{s}_i$ and directly removes $\boldsymbol{s}_i$ to get the plaintext of the weight parameters $\boldsymbol{w}_{\text{global}}$, and then updates the model using the weight parameters.

B. Cryptanalysis of Zhang et al.'s DeepPAR Scheme

Theorem 1. *During the re-encryption key generation process, if the proxy server eavesdrops on r_i sent by the parameter server and the parameter server eavesdrops on $x_i^{-1} \cdot r_i$ sent by the participant, the participant's secret key is no longer secure.*

Proof: Suppose the proxy server eavesdrops on r_i sent by the parameter server, once the proxy server receives $x_i^{-1} \cdot r_i$ from the participant, it can remove r_i to get x_i^{-1}. Since p' is a prime number, we can know that x_i^{-1} and p' have a greatest common factor of 1, then there must exist integers u and v such that x_i^{-1} and p satisfy the following equation:

$$1 = u \cdot x_i^{-1} + v \cdot p'. \tag{6}$$

So we can use the extended Euclidean algorithm to find u, and finally u is the x_i we require to get the secret key of the participant. Similarly, when the parameter server eavesdrops on the $x_i^{-1} \cdot r_i$ sent by the participant, since the parameter server knows r_i, it can easily find x_i^{-1}, then the parameter server can also use the extended Euclidean algorithm to find x_i, and thus acquires the secret key of the participant. ■

Theorem 2. *During the re-encryption key generation, if the parameter server eavesdrops on $x_i^{-1} \cdot r_i$ that the participant sends to the proxy server and the participant eavesdrops on $x_i^{-1} \cdot r_i \cdot x_o$ that the proxy server sends to the parameter server, the proxy server's secret key is no longer secure.*

Proof: Suppose the parameter server eavesdrops on the $x_i^{-1} \cdot r_i$ sent by the participant to the proxy server, when the parameter server receives the $x_i^{-1} \cdot r_i \cdot x_o$ sent by the proxy server, he uses the eavesdropped $x_i^{-1} \cdot r_i$ to obtain the secret key x_o of the proxy server by direct calculation, so that the secret key x_o of the proxy server will be leaked, which will cause the gradients leakage in the parameter server. Similarly, when the participant eavesdrops on $x_i^{-1} \cdot r_i \cdot x_o$ sent by the proxy server to the parameter server, the participant can use $x_i^{-1} \cdot r_i$ to directly get the secret key x_o of the proxy server. Both types of eavesdropping cause the proxy server's secret key to be no longer secure. ■

Theorem 3. *DeepPAR is not resistant to collusion attacks between the parameter server and participants.*

Proof: In the DeepPAR scheme, the re-encryption key $rk_i = x_i^{-1} \cdot x_o$, which includes the secret key information of the proxy server, can be easily obtained if the parameter server conspires with any of the participants to get x_i^{-1}, so that the parameter server can get the secret key x_o of the proxy server. In the re-encryption stage, all participants upload their encrypted gradient information to the parameter server, and after the parameter server finishes re-encryption, all participants' ciphertext is obtained with the same public key pk of the proxy server, so that the parameter server can use the obtained secret key x_o of the proxy server to decrypt all users' ciphertext, which causes the users' private information to be disclosed to the parameter server, so the scheme is not resistant to collusion attacks. ■

VI. Our Proposal

In this section, we propose an improved DeepPAR-based asynchronous deep learning scheme to address the security issues of Zhang *et al.*'s scheme, the security analysis will be presented later.

A. Improved DeepPAR Scheme

In our improved scheme, we have modified the generation and transmission of re-encryption key to enhance security.

1) **KeyGen**: Each participant P_i randomly picks a number $x_i \in \mathbb{Z}_{p'}^*$ as its secret key and computes its public key g^{x_i}, denoted as (pk_i, sk_i). In like manner, the proxy server randomly chooses a number $x_o \in \mathbb{Z}_{p'}^*$ as its secret key sk and computes its public key $pk = g^{x_o}$, denoted as (pk, sk). Finally, the parameter server, proxy server and participants generate the re-encryption key rk_i following the steps below.
   a) The parameter server $\mathcal{S}$ selects a random number r_i and sends its ciphertext $[\![r_i]\!]_{pk_i}$ encrypted with the public key pk_i to the participant P_i.
   b) After receiving the ciphertext $[\![r_i]\!]_{pk_i}$, P_i decrypts it with its own secret key sk_i, calculates $x_i^{-1} \cdot r_i$, and then sends its ciphertext $[\![x_i^{-1} \cdot r_i]\!]_{pk}$ encrypted with the proxy server's public key pk to the proxy server $\mathcal{O}$.
   c) $\mathcal{O}$ chooses a random number t_i and computes g^{t_i}. Then $\mathcal{O}$ decrypts $[\![x_i^{-1} \cdot r_i]\!]_{pk}$ using its own secret key x_o. Finally $\mathcal{O}$ computes $x_i^{-1} \cdot r_i \cdot t_i^{-1} \cdot x_o$ using its own secret key x_o and t_i^{-1} and sends it to $\mathcal{S}$.
   d) $\mathcal{S}$ receives $x_i^{-1} \cdot r_i \cdot t_i^{-1} \cdot x_o$ and multiplies it by r_i^{-1}, and $\mathcal{S}$ can obtain the re-encryption key $rk_i = x_i^{-1} \cdot t_i^{-1} \cdot x_o$ for the ith participant.
2) **Compute**: Each participant P_i downloads the weight parameters stored in the processing unit PU_j of the parameter server and then decrypts them, with the specific decryption process described in the **Decrypt** phase, which yields the current weights $\boldsymbol{w}_{\text{global}}$, where $j \in I_i^{(\text{down})} \subset [1, N]$. Then, a small batch dataset is selected and trained with deep learning to obtain a new gradient vector $G = \left(G^{(1)}, \ldots, G^{(N)}\right)$, and finally the participants encrypt $-\alpha \cdot G^{(d)}$ to obtain the ciphertext

$E_{pk_i}\left(-\alpha \cdot G^{(d)}\right)$, where $d \in I_i^{(\mathrm{up})} \subset [1, N]$, and sends it to the corresponding processing unit PU_j of the parameter server.

In the following, we specify the process of encryption. For the sake of convenience of description, we take one of the gradients in the gradient vector Δw for illustration, P_i uses g^{t_i} for encryption, and the ciphertext $C = (c_1, c_2)$ is obtained according to the following equation.

$$c_1 = g^{\Delta w} \cdot g^{r_i}, \tag{7}$$

$$c_2 = (g^{t_i})^{x_i r_i}. \tag{8}$$

3) **Re-encrypt**: When $\mathcal{S}$ receives the gradient encrypted by participant P_i using his public key pk_i, $\mathcal{S}$ converts the ciphertext into another ciphertext under the same public key pk of the $\mathcal{O}$ using the re-encryption key rk_i. The conversion process is as follows.

$$c_1 = g^{\Delta w} \cdot g^{r_i}, \tag{9}$$

$$c_2^{rk_i} = (g^{t_i x_i r_i})^{x_i^{-1} \cdot t_i^{-1} \cdot x_o} = g^{x_o r_i}. \tag{10}$$

4) **Aggregate**: When each processing unit PU_j of $\mathcal{S}$ has completed re-encryption of the received gradient vectors, $\mathcal{S}$ aggregates all the gradient vector ciphertexts to obtain $E_{pk}\left(\boldsymbol{w}_{\text{global}}^{(d)}\right) = E_{pk}\left(\boldsymbol{w}_{\text{global}}^{(d)}\right) + E_{pk}\left(-\alpha \cdot G^{(d)}\right)$, where $d \in I_i^{(\mathrm{up})} \subset [1, N]$.

5) **Decrypt**: When the $\mathcal{S}$ completes the aggregation, each participant downloads the aggregation results $E_{pk}\left(\boldsymbol{w}_{\text{global}}\right)$, but at this point the ciphertexts are encrypted using the public key of the proxy server, and the participant cannot decrypt them, so they can only be sent to the proxy server for decryption. To protect the value of the global weight parameters $E_{pk}\left(\boldsymbol{w}_{\text{global}}\right)$, P_i selects the blind factor s_i and calculates $E_{pk}\left(\boldsymbol{w}_{\text{global}} + \boldsymbol{s}_i\right)$, then sends them to the proxy server for decryption. The proxy server decrypts them and sends $\boldsymbol{w}_{\text{global}} + \boldsymbol{s}_i$ to the participant. The participant receives $\boldsymbol{w}_{\text{global}} + \boldsymbol{s}_i$ and directly removes $\boldsymbol{s}_i$ to get the plaintext $\boldsymbol{w}_{\text{global}}$ of the weight parameters, and then updates the model using the weight parameters.

B. Security analysis

In this section, we analyze the security of the above scenario. All data in our scenario is transmitted over an insecure public channel and is at risk of being eavesdropped. It is assumed in this scenario that the parameter server and the proxy server do not conspire, but the server may conspire with some participants to access the private data of other participants. Besides, the scheme utilizes the ElGamal cryptography, which is semantically secure under the decisional Diffie Hellman (DDH) and discrete logarithm hard problem. Based on the ElGamal security, we prove the security of our improved scheme according to the real/ideal world model. The specific security analysis is as follows.

Theorem 4. *The **KeyGen** phase is secure against the malicious adversaries $\mathcal{A}_{P_i}$, $\mathcal{A}_O$, $\mathcal{A}_S$.*

Proof: In the key generation process, a challenge parameter server can safely interact with a malicious adversary $\mathcal{A}_S$. In the real world, the view of the adversary $\mathcal{A}_S$ in step a) and step d) is:

$$V_{\text{REAL}} = \{r_i, [r_i]_{pk_i}, x_i^{-1} \cdot r_i \cdot t_i^{-1} \cdot x_o, x_i^{-1} \cdot t_i^{-1} \cdot x_o\}.$$

After constructing a simulator SIM, the simulator SIM replaces the parameter server to obtain the output of the party T in the ideal world. The view of the simulator SIM is:

$$V_{\text{IDEAL}} = \{r_1', r_{11}', r_1' \cdot r_2', r_2'\},$$

where $r_1', r_{11}', r_1' \cdot r_2', r_2' \in \mathbb{Z}_p{}^*$. Owing to the semantic security of ElGamal cryptosystem and the randomness of the random numbers, it can conclude that the real world and the ideal world are indistinguishable in the process:

$$\{\text{IDEAL}_{f,SIM}(V_{\text{IDEAL}})\} \overset{c}{\approx} \left\{\text{REAL}_{\prod,\mathcal{A}_S}(V_{\text{REAL}})\right\}.$$

In step b), a challenge participant P_i communicates with a malicious adversary $\mathcal{A}_{P_i}$. The view of the adversary $\mathcal{A}_{P_i}$ in the real world is:

$$V_{\text{REAL}}{}' = \{[r_i]_{pk_i}, r_i, x_i^{-1} \cdot r_i, [x_i^{-1} \cdot r_i]_{pk}\}.$$

The view of the constructed simulator SIM in the ideal world is:

$$V_{\text{IDEAL}}{}' = \{r_{ii}, r_i', r_{xi}, r_{xi}'\},$$

where $r_{ii}, r_i', r_{xi}, r_{xi}' \in \mathbb{Z}_p{}^*$. If the server S sends $\{\perp\}$ to the trusted party T, the view of the simulator SIM remains unchanged. Owing to the semantic security of ElGamal cryptosystem and the randomness of the random numbers, it demonstrates that the ideal world is indistinguishable from the real world:

$$\left\{\text{IDEAL}_{f,SIM}(V_{\text{IDEAL}}{}')\right\} \overset{c}{\approx} \left\{\text{REAL}_{\prod,\mathcal{A}_S}(V_{\text{REAL}}{}')\right\}.$$

In step c), a challenge proxy server O interacts with a malicious adversary $\mathcal{A}_O$. The view of the adversary $\mathcal{A}_O$ in the real world is:

$$U_{\text{REAL}} = \{t_i, g^{t_i}, x_i^{-1} \cdot r_i, x_i^{-1} \cdot r_i \cdot t_i^{-1} \cdot x_o\}.$$

In the ideal world, the simulator SIM replacing the proxy server O is built to communicate with the trusted party T, and obtains the output. The view of the simulator SIM in the ideal world is:

$$U_{\text{IDEAL}} = \{r_t, r_t', r_{xt}, r_{xt} \cdot r_{tx}'\},$$

where $r_t, r_t', r_{xt}, r_{xt} \cdot r_{tx}' \in \mathbb{Z}_p{}^*$. If the participant P_i sends $\{\perp\}$ to the trusted party T, the view of the simulator SIM remains unchanged. In short, owing to discrete logarithm problem and randomness of the random numbers, the output of the ideal world cannot distinguish between the output of the real world:

$$\{\text{IDEAL}_{f,SIM}(U_{\text{IDEAL}})\} \overset{c}{\approx} \left\{\text{REAL}_{\prod,\mathcal{A}_O}(U_{\text{REAL}})\right\}.$$

In summary, because the re-encryption key generation process is secure, the secret keys of the participants and the proxy server are secure. ■

Theorem 5. *The **Compute, Re-encrypt**, and **Aggregate** phase is secure against the malicious adversaries $\mathcal{A}_{P_i}$, $\mathcal{A}_S$.*

Proof: In the **Compute** process, knowning g^{t_i} and g^{x_i}, any adversary cannot acquire $g^{t_i x_i}$, and then cannot obtain r_i from $g^{t_i x_i r_i}$ owing to DDH and discrete logarithm problem. Besides, in the real world, the adversary $\mathcal{A}_{P_i}$ can obtain the view by interacting with the participant P_i: $U_{\text{REAL}}{}' = \{C\}$. In the ideal world, the contructed simulator SIM replacing the participant P_i communicates with the trusted party T, obtaining the view $U_{\text{IDEAL}}{}' = \{r_C\}$, where $r_C \in \mathbb{Z}_p{}^*$. According to the semantic security of ElGamal cryptosystem, the output of the ideal world cannot distinguish between the output of the real world:

$$\left\{\text{IDEAL}_{f,SIM}(U_{\text{IDEAL}}{}')\right\} \overset{c}{\approx} \left\{\text{REAL}_{\prod,\mathcal{A}_{P_i}}(U_{\text{REAL}}{}')\right\}.$$

In the **Re-encrypt** phase, knowning g^{x_o} and ciphertext $[r_i]_{pk_i}$, any adversary cannot obtain $g^{x_o r_i}$ owing to the semantic security of ElGamal cryptosystem. In addition, the output of the **Re-encrypt** and **Aggregate** phases also cannot distinguish between the ideal world and the real world. ■

Theorem 6. *The **Decrypt** phase is secure against the malicious adversaries $\mathcal{A}_{P_i}$, $\mathcal{A}_O$.*

Proof: In the **Decrypt** phase, a challenge participant P_i can communicate with the adversary $\mathcal{A}_{P_i}$. In the real world, the view of the adversary $\mathcal{A}_{P_i}$ is:

$$Q_{\text{REAL}} = \{E_{pk}\left(\mathbf{w}_{\text{global}}\right), E_{pk}\left(\mathbf{w}_{\text{global}} + \boldsymbol{s}_i\right), \mathbf{w}_{\text{global}} + \boldsymbol{s}_i\}.$$

In the ideal world, the constructed simulator SIM replacing the participant P_i acquires the output from the trusted party T. The view of the simulator SIM is:

$$Q_{\text{IDEAL}} = \{r_{wg}, r_{wgs}, r_{ws}\},$$

where $r_{wg}, r_{wgs}, r_{ws} \in \mathbb{Z}_p{}^*$. According to the semantic security of ElGamal cryptosystem and the randomness of the random numbers, the output of the ideal world is indistinguishable from that of the real world:

$$\{\text{IDEAL}_{f,SIM}(Q_{\text{IDEAL}})\} \overset{c}{\approx} \left\{\text{REAL}_{\prod,\mathcal{A}_{P_i}}(Q_{\text{REAL}})\right\}.$$

For a challenge proxy server, the adverary $\mathcal{A}_O$ can obtain its view by communicating with the proxy server in the real world:

$$Q_{\text{REAL}}{}' = \{E_{pk}\left(\mathbf{w}_{\text{global}} + \boldsymbol{s}_i\right), \mathbf{w}_{\text{global}} + \boldsymbol{s}_i\}.$$

In the ideal world, the simulator SIM replacing the challenge proxy server interacts with the trusted party T, obtaining the view as follows:

$$Q_{\text{IDEAL}}{}' = \{r_{wgs}{}', r_{ws}{}'\},$$

where $r_{wgs}{}', r_{ws}{}' \in \mathbb{Z}_p{}^*$. Due to the same semantic security and randomness, the output cannot distinguish between the ideal world and the real world:

$$\left\{\text{IDEAL}_{f,SIM}(Q_{\text{IDEAL}}{}')\right\} \overset{c}{\approx} \left\{\text{REAL}_{\prod,\mathcal{A}_O}(Q_{\text{REAL}}{}')\right\}.$$

■

Theorem 7. *The improved scheme can resist collusion attacks between the parameter server and participants.*

Proof: The parameter server in the improved scheme has the re-encryption key $rk_i = x_i^{-1} \cdot t_i^{-1} \cdot x_o$. Assuming that the parameter server and any of the participants conspire, the parameter server can only get x_i^{-1}, and due to the presence of t_i^{-1}, the parameter server cannot get the secret key of the proxy server x_o, so the private information of the other participants will not be obtained by the parameter server. The process solves the problem of theorem 3. In summary, the improved scheme can resist the server and participant's collusion threat. ■

VII. Performance Evaluation

In this section, we evalate and experiment with the performance and accuracy of our improved scheme. First, we compare the communication overhead and compatational cost with DeepPAR [8] and Tang *et al.* [32]. Thereinto, DeepPAR [8] is the original scheme of our scheme, and Tang *et al.* [32] is a comparison scheme solving the collusion problem. For comparison, in the scheme of Tang *et al.*, we define step 1-step 4 as the **KeyGen** phase, step 5-step 6 as the **Compute** phase, step 7 as the **Re-encrypt** and **Aggregate** phase, and step 8 as the **Decrypt** phase.

A. Communication Overhead

Improved scheme: Let the length of the large prime p be k bits, then the length of the ciphertext for each ElGamal encryption algorithm is $2k$ bits. In the **KeyGen** phase, the main communication overhead is in the re-encryption key generation process, and its total communication overheads are $5nk$. In the **Compute** phase, participants compute the ciphertext gradients and upload them to the parameter server, and their communication overheads are $2nMk$, where M is the number of gradients. In the **Decrypt** phase, participants need to download the global ciphertext weights and decrypt them with the proxy server, and their communication overheads are $5nMk$. There is no communication overhead for the other phases of the scheme. Thus the communication overheads of the whole process are $7nMk + 5nk$ bits.

Compared scheme: In the **KeyGen** phase, the total communication overheads of the re-encryption key generation process in DeepPAR [8] and Tang *et al.* [32] are $3nk$ and $5nk$, respectively. In the **Compute** phase, the user computes the ciphertext gradients and uploads them to the parameter server, and their communication overheads in DeepPAR [8] and Tang *et al.* [32] are $2nMk$ and $4nMk$+$8Mk$, respectively. In the **Decrypt** phase, the participant needs to download the ciphertext weights and decrypt them with the proxy server, and their communication overheads in DeepPAR [8] and Tang *et al.* [32] are $5nMk$ and $4nMk$ severally. There is no communication overhead for the other phases of the scheme.

Since several iterations of training are required in the process of training a deep learning model, initialization process is required for the first training, and initialization is not required after the first training. Then, we compare the communication

overhead in different phases of several schemes, as shown in Table II.

TABLE II: Communication overhead comparison

Phase	Our scheme	DeepPAR [8]	Tang *et al.* [32]
KeyGen	$5nk$	$3nk$	$5nk$
Compute	$2nMk$	$2nMk$	$4nMk + 8Mk$
Decrypt	$5nMk$	$5nMk$	$4nMk$
Total	$7nMk+5nk$	$7nMk+3nk$	$8nMk+5nk+8Mk$

In addition, we tested the communication overhead of the improved and comparison schemes with different numbers of participants and parameters. The test results are shown in Fig. 2 and Fig. 3. From the test results and the table results we can see that the communication overheads of the improved scheme and the DeepPAR [8] scheme are basically the same, and that of Tang *et al.* [32] is obviously higher than the other two schemes. The results demonstrate that the difference in communication overhead between the improved and original schemes is not significant.

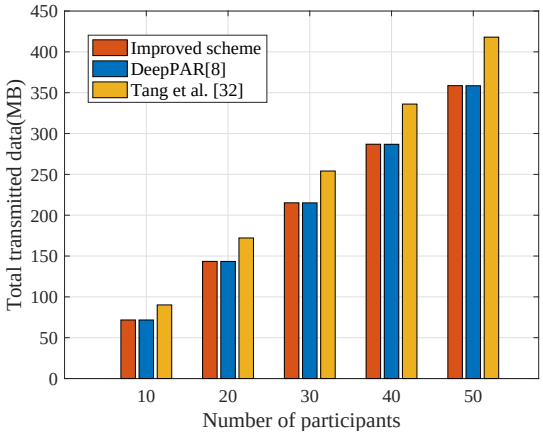


Fig. 2: Comparison of communication overhead between the improved and comparison schemes under different number of participants

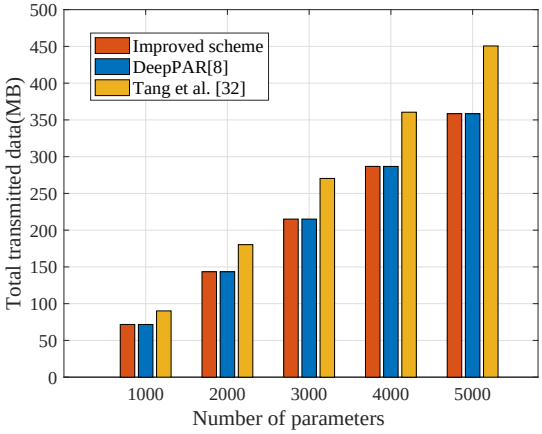


Fig. 3: Comparison of the communication overhead between the improved and comparison schemes under different number of parameters

B. Computational Cost

We analyze the computational cost of the improved and comparison schemes in the subsection. To simplify the representation, we denote a/an multiplication/exponentiation as Mul/Exp, a homomorphic re-encryption as Enc, and a hash as Hash.

Improved scheme: In the **KeyGen** phase, the parameter server gets the corresponding re-encryption key for each participant, in which the parameter server performs $3n$Exp + $2n$Mul costs, each participant takes 4Exp + 3Mul costs, and the proxy server executes $2n$Exp + $3n$Mul costs. In the **Compute** phase, participants encrypt the gradient vectors and upload them to the parameter server, and this phase requires $3M$Exp + MMul costs for each participant. In the **Re-encryption** phase, the parameter server needs to convert all ciphertexts into another set of ciphertexts with the same public key pk of the proxy server, in this phase, the parameter server performs MExp costs. In the **Aggregate** phase, the parameter server needs to take $2M$Mul costs. In the **Decrypt** phase, the participants send the blinded model parameters to the proxy server for decryption. This phase requires MExp + MMul computational costs for each participant and MExp + MMul computational costs for the proxy server.

Original scheme: In the **KeyGen** phase, the parameter server gets the corresponding re-encryption key for each participant. In this phase, the parameter server performs nMul costs, each participant takes nMul costs, and the proxy server spends nMul computational costs. In the **Compute** phase, participants encrypt the gradient vectors and upload them to the parameter server, this phase requires $3M$Exp + MMul computational costs for each participant. In the **Re-encrypt** phase, the parameter server needs to convert all ciphertexts into another set of ciphertexts with the same public key pk of the proxy server, in this phase, the parameter server needs to perform MExp costs. In the **Aggregate** phase, the parameter server needs to perform $2M$Mul costs. In the **Decrypt** phase, the participant sends the blinded parameters to the proxy server for decryption, among them, the participant performs MExp + MMul costs and the proxy server takes MExp + MMul computational costs.

Tang *et al.*'s scheme: In the **KeyGen** phase, key transform server (KTS) takes Mul + 2Exp computational costs, participants performs Exp + MMul computational costs, and data service provider (DSP) provides 2Exp costs. In the **Compute** phase, participant takes Mul + MEnc costs to compute the ciphertext gradients, and KTS costs MEnc + n(Hash + 2Enc) to compute the first phase of re-encryption. In the **Re-encrypt** phase, DSP performs n(Hash + 2Enc + Exp) costs to compute the second phase of re-encryption. In the **Aggregate** phase, KTS takes nEnc costs to aggregate the weights. In the **Decrypt** phase, participant takes M(2Hash + 3Mul + 6Exp) computational costs to decrypt.

As a summary, in the improved scheme, each participant needs to perform $4(M + 1)$Exp + $(2M + 3)$Mul costs, the

parameter server needs to take $(3n+M)$Exp $+2(n+M)$Mul computational costs, and the proxy server needs to compute $(2n+M)$Exp $+(3n+M)$Mul costs. In the original scheme, each participant performs $4M$Exp $+(2M+1)$Mul costs, the parameter server takes MExp $+(2M+n)$ Mul computational costs, and the proxy server needs to compute MExp $+(M+n)$Mul costs. In the Tang *et al.*'s scheme, each participant spends $(6M+1)$Exp + $4M$Mul + MEnc + $2M$Hash computational costs, DSP computes $(n$ + 2)Exp + nHash + $2n$Enc costs, and KTS takes Mul + 2Exp + $(M+3n)$Enc + nHash costs. The specific statistics are shown in Table III.

As can be seen from Table III, the computational cost of participants in the improved scheme is only slightly increased compared to the original scheme; the main increase in computational cost is in the parameter server and the proxy server, which have a powerful computational power. Besides, Tang *et al.*'s scheme is distinctly more computational cost than the other two schemes. Therefore, our scheme successfully solves the privacy leakage problem of the original scheme with a slight increase in participant overhead.

C. Experimental Performance

In this subsection, we present the results of a series of experiments on a real dataset. The experiments were implemented using Python3.7, and the benchmarks are over on an Xeon CPU Sliver 4110 @2.10 GHz server.

Dataset. We use the MNIST dataset, which consists of 60,000 training handwritten digits and 10,000 test digits from "0" to "9". Each dataset contains 784 features, representing 28 $*$ 28 pixels in the image. In this paper, we use PyTorch 1.7.0 to build and train a LeNet deep learning model to evaluate the performance of our scheme.

Based on the above theoretical analysis, we can know that there is a large performance difference between the generation process of the re-encryption key rk_i in the **KeyGen** phase of the improved scheme and the comparison schemes. Therefore, we tested the re-encryption key generation time of the improved scheme and the comparison schemes [8, 32] under different numbers of participants, and the specific test results are shown in Fig. 4.

In Fig. 4, the improved scheme has more running time than the comparison schemes with the number of participants $\{100, 200, \ldots, 1000\}$ growing in the re-encryption key generation process. However, the key generation time is only an initialization step performed before training and does not need to be initialized in each iteration during iterative training, so it has little impact on the running time of training. In addition, due to the change in the re-encryption key generation process, the security of the improved scheme is a huge improvement over the original scheme [8].

Since the main purpose of the scheme is to complete the training of the deep learning model and guarantee the accuracy of the model, we again tested the running time of the improved scheme and the comparison schemes with different training parameters, and the specific test results are shown in Fig. 5.

As shown in Fig. 5, the running time of the improved scheme is almost the same as that of the comparison schemes

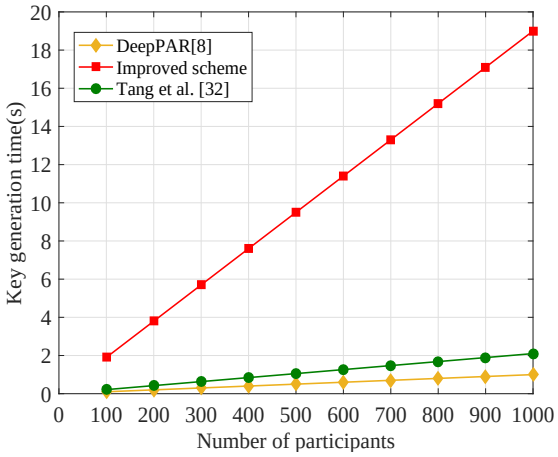


Fig. 4: Comparison of key generation times

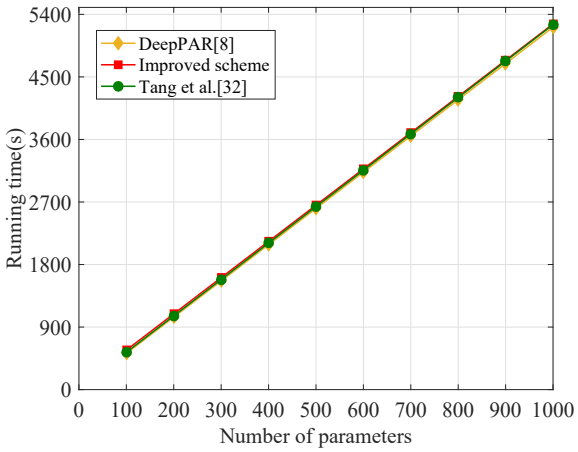


Fig. 5: Comparison of the running times of the schemes with different number of parameters

[8, 32] with the number of parameters $\{100, 200, \ldots, 1000\}$ growing. The results demonstrate that their running time is approximately proportional to the number of parameters. This is because the improved scheme has finished the initialization of the parameters during the iterative training of the model, and there is no need to repeat the initialization during the training process, so the training time of the model is basically the same as that of the original scheme [8].

Then we tested the effect of different number of model parameters on the model training accuracy, and the experimental results are shown in Fig. 6.

From Fig. 6, it can be seen that as the number of parameters $\{10000, 20000, \ldots, 60000\}$ increases, the training accuracy of the model also grows. However, with the increase of the number of parameters, the running time of the model also rises. Therefore, in order to train the model more efficiently, participants can make a compromise between runtime and accuracy by selecting the appropriate model parameters for training.

Finally, we tested the accuracy of the model for learning rates $\alpha = 0.01$, $\alpha = 0.001$ and $\alpha = 0.0001$, and the specific

TABLE III: Computational cost comparison

Phase	Our scheme	DeepPAR [8]	Tang *et al.* [32]
Participant	$4(M+1)$ Exp + $(2M+3)$Mul	$4M$Exp + $(2M+1)$Mul	$(6M+1)$Exp + $4M$Mul + MEnc + $2M$Hash
Parameter Server (DSP)	$(3n+M)$Exp + $2(n+M)$Mul	MExp + $(2M+n)$Mul	$(n+2)$Exp + nHash + $2n$Enc
Proxy Server (KTS)	$(2n+M)$Exp + $(3n+M)$Mul	MExp + $(M+n)$Mul	Mul + 2Exp + $(M+3n)$Enc + nHash

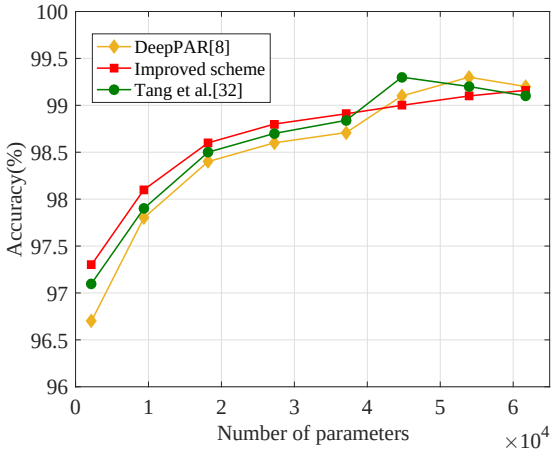


Fig. 6: Model accuracy with different number of parameters

experimental results are shown in Fig. 7. As shown in Fig. 7, we can see that when $\alpha = 0.01$, the accuracy of the model varies irregularly with the growth of the epochs, which is due to the poor choice of the learning rate causing the model to oscillate constantly at the local optimal point and fail to converge to the optimal point. The overall accuracy of the model at $\alpha = 0.001$ and $\alpha = 0.0001$ is increasing with the epochs $\{0, 5, 10, \ldots, 40\}$ growing, and the model reaches its highest accuracy of 99.06 % at $\alpha = 0.0001$.

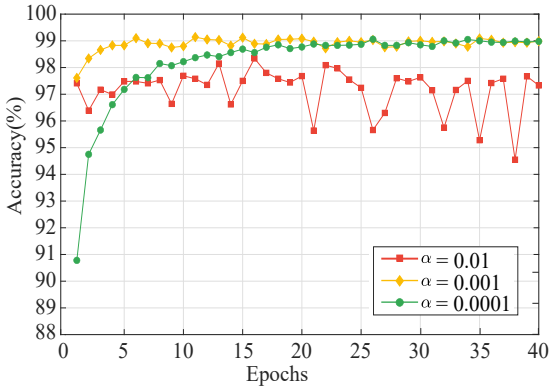


Fig. 7: Accuracy of the improved scheme at different learning rates

VIII. Conclusion

In this paper, we propose an analysis and improvement scheme based on the DeepPAR asynchronous deep learning scheme designed by Zhang *et al*. to address the security issues in the DeepPAR asynchronous deep learning scheme. First, this paper analyses the security problems in the original DeepPAR scheme. Then we propose an improved scheme to address these security problems, and provide a detailed analysis and discussion of the security of the improved scheme. Finally, we compare the improved scheme with the original DeepPAR scheme in terms of communication overhead, computational cost and experimental performance. The runtime of the improved scheme is higher than that of the original scheme in the initialization phase, and the runtime and training accuracy are basically the same as those of the original scheme, but the security of the improved scheme is higher, and this scheme is more feasible in a high privacy-preserving scenario like deep learning. In future work, we will focus on other attacks method such as reconstruction attacks, model inversion attacks and attribute-inference attacks in the privacy-preserving FL framework.

References

[1] A. A. Moghaddam, A. Seifi, and T. Niknam, "Multi-operation management of a typical micro-grids using particle swarm optimization: A comparative study," *Renewable and Sustainable Energy Reviews*, vol. 16, no. 2, pp. 1268–1281, 2012.

[2] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Trans. Ind. Informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[3] D. He, M. Ma, S. Zeadally, N. Kumar, and K. Liang, "Certificateless public key authenticated encryption with keyword search for industrial internet of things," *IEEE Trans. Ind. Informatics*, vol. 14, no. 8, pp. 3618–3627, 2018.

[4] B. Liu, Y. Li, Y. Liu, Y. Guo, and X. Chen, "PMC: A privacy-preserving deep learning model customization framework for edge computing," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 4, no. 4, pp. 139:1–139:25, 2020.

[5] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[6] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1310–1321.

[7] 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, 2018.

[8] X. Zhang, X. Chen, J. K. Liu, and Y. Xiang, "Deeppar and deepdpa: Privacy preserving and asynchronous deep

learning for industrial iot," *IEEE Trans. Ind. Informatics*, vol. 16, no. 3, pp. 2081–2090, 2019.

[9] B. Liu, Y. Guo, and X. Chen, "PFA: privacy-preserving federated adaptation for effective model personalization," in *Proc. WWW '21*, 2021, pp. 923–934.

[10] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Proc. ESORICS 2020*, vol. 12308, 2020, pp. 480–501.

[11] Y. Chen, Y. Ping, Z. Zhang, B. Wang, and S. He, "Privacy-preserving image multi-classification deep learning model in robot system of industrial iot," *Neural Comput. Appl.*, vol. 33, no. 10, pp. 4677–4694, 2021.

[12] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS 2017*, vol. 54, 2017, pp. 1273–1282.

[13] C. Dwork, "Differential privacy: A survey of results," in *Proc. International conference on theory and applications of models of computation*, 2008, pp. 1–19.

[14] N. Phan, X. Wu, and D. Dou, "Preserving differential privacy in convolutional deep belief networks," *Machine learning*, vol. 106, no. 9-10, pp. 1681–1704, 2017.

[15] R. Han, D. Li, J. Ouyang, C. H. Liu, G. Wang, D. Wu, and L. Y. Chen, "Accurate differentially private deep learning on the edge," *IEEE Trans. Parallel Distributed Syst.*, vol. 32, no. 9, pp. 2231–2247, 2021.

[16] N. Phan, X. Wu, H. Hu, and D. Dou, "Adaptive laplace mechanism: Differential privacy preservation in deep learning," in *Proc. 2017 IEEE ICDM*, 2017, pp. 385–394.

[17] 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, 2016.

[18] M. Gong, J. Feng, and Y. Xie, "Privacy-enhanced multi-party deep learning," *Neural Networks*, vol. 121, pp. 484–496, 2020.

[19] M. Kim, J. Lee, L. Ohno-Machado, and X. Jiang, "Secure and differentially private logistic regression for horizontally distributed data," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 695–710, 2019.

[20] 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*, 2016, pp. 308–318.

[21] L. Xiang, J. Yang, and B. Li, "Differentially-private deep learning from an optimization perspective," in *Proc. INFOCOM 2019*, 2019, pp. 559–567.

[22] A. C. Yao, "Protocols for secure computations (extended abstract)," in *Proc. 23rd Annual Symposium on Foundations of Computer Science*, 1982, pp. 160–164.

[23] X. Zhang, T. Jiang, K. Li, A. Castiglione, and X. Chen, "New publicly verifiable computation for batch matrix multiplication," *Inf. Sci.*, vol. 479, pp. 664–678, 2019.

[24] O. Goldreich, "Secure multi-party computation," *Manuscript. Preliminary version*, vol. 78, 1998.

[25] Y. Chen, B. Wang, and Z. Zhang, "Pdlhr: Privacy-preserving deep learning model with homomorphic re-encryption in robot system," *IEEE Syst. J.*, pp. 1–12, 2021.

[26] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

[27] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, and H. Ludwig, "Hybridalpha: An efficient approach for privacy-preserving federated learning," in *Proc. AISec@CCS 2019*, 2019, pp. 13–23.

[28] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *Proc. 2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 19–38.

[29] A. Bansal, T. Chen, and S. Zhong, "Privacy preserving back-propagation neural network learning over arbitrarily partitioned data," *Neural Comput. Appl.*, vol. 20, no. 1, pp. 143–150, 2011.

[30] D. He, N. Kumar, N. Chilamkurti, and J. H. Lee, "Lightweight ecc based rfid authentication integrated with an id verifier transfer protocol," *Journal of Medical Systems*, vol. 38, no. 10, pp. 1–6, 2014.

[31] R. Amin, S. H. Islam, G. P. Biswas, M. K. Khan, and N. Kumar, "An efficient and practical smart card based anonymity preserving user authentication scheme for TMIS using elliptic curve cryptography," *J. Medical Syst.*, vol. 39, no. 11, pp. 1–18, 2015.

[32] F. Tang, W. Wu, J. Liu, H. Wang, and M. Xian, "Privacy-preserving distributed deep learning via homomorphic re-encryption," *Electronics*, vol. 8, no. 4, pp. 1–21, 2019.

[33] X. Ma, F. Zhang, X. Chen, and J. Shen, "Privacy preserving multi-party computation delegation for deep learning in cloud computing," *Inf. Sci.*, vol. 459, pp. 103–116, 2018.

[34] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, 1985.

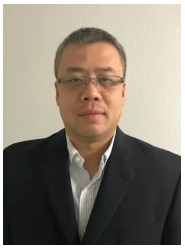
[35] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Proc. International Conference on the Theory and Applications of Cryptographic Techniques*, 1998, pp. 127–144.

[36] M. Avriel, *Nonlinear programming: analysis and methods*. Courier Corporation, 2003.

Yange Chen is currently learning for a Ph.D. in School of Telecommunications Engineering, Xidian University, and she is an associate professor in School of Information Engineering, Xuchang University. She received her MS and BS degrees in computer application technology from Henan Polytechnic University in 2008 and 2006, respectively. Her main research interests include deep learning security, Internet of Things security, homomorphic encryption.

Suyu He received the bachelor's degree from the School of Telecommunications Engineering, Xidian University in 2018. He is currently pursuing the Master degree supervised by Prof.Baocang Wang. His research interests include homomorphic encryption, machine learning and their applications to practical problems.

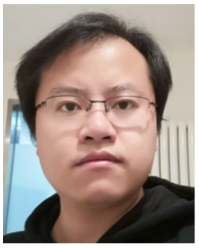
Baocang Wang is a professor in the School of Telecommunications Engineering, Xidian University. He received his Ph.D. degree in cryptography from Xidian University in 2006, and received his MS and BS degrees in mathematics from Xidian University in 2004 and 2001, respectively. His main research interests include public key cryptography, encryption data processing, data mining security.



Pu Duan has been a twenty-year veteran on cryptography, information security and networking security. He has published 20+ papers on areas of cryptography, networking security and system security. He joined Cisco after obtaining his Ph.D. degree from Texas A&M University in 2011. At Cisco he led the research and development of new cryptographic algorithms for TLS.13 on cisco firewall product. Currently Dr. Duan works in Secure Collaborative Lab (SCI) lab at Ant Group as a senior staff engineer, leading the team on research and implementation of privacy-preserving technologies.

Benyu Zhang has been engaged in AI research and development for two decades. He published 49 peer-reviewed papers with more than 11,000 citations, has 70 US patents approved and 84 US patents pending. Since he went into industry, he has Initiated and led numbers of core AI systems R&D in advertising, search, and recommendation at Google and FB. Currently he leads the Secure Collaborative Intelligence Lab (SCI Lab) at Ant Group to build the next generation privacy-preserving data mining and AI platform. The vision is to form the "data internet" and enable data mining and AI applications on data from multiple parties securely, efficiently, and effectively.

Zhiyong Hong received the B.S. and M.S. degrees in computer science and technology from the Shenyang Institute of Technology, Shenyang, China, in 2001 and 2004, respectively, and the Ph.D. degree in computer science and technology from Southwest Jiaotong University, Chengdu, China, in 2014. From 2006 to 2014, he was a Lecturer with the School of Computer, Wuyi University, Jiangmen, China. Since 2015, he has been an Associate Professor. His research interests include intelligent information processing, block chain, and rough set theory.



Yuan Ping received the B.S. degree in electronics and information engineering from Southwest Normal University, in 2003, the M.S. degree in mathematics from Henan University, in 2008, and the Ph.D. degree in information security from the Beijing University of Posts and Telecommunications, in 2012. He was a Visiting Scholar with the School of Computing and Informatics, University of Louisiana at Lafayette. He was a Visiting Scholar with the Department of Computing Science, University of Alberta. He is currently an Professor with Xuchang University. His research interests include machine learning, data privacy and security, and cloud and edge computing.