A Privacy-Preserving and Verifiable Federated Learning Scheme

Xianglong Zhang*†, Anmin Fu*†, Huaqun Wang†, Chunyi Zhou* and Zhenzhu Chen*
*School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, 210094, PR China
†Jiangsu Key Laboratory of Big Data Security and Intelligent Processing,
Nanjing University of Posts and Telecommunications, Nanjing, 210023, PR China
fuam@njust.edu.cn

***Abstract*—Due to the complexity of the data environment, many organizations prefer to train deep learning models together by sharing training sets. However, this process is always accompanied by the restriction of distributed storage and privacy. Federated learning addresses this challenge by only sharing gradients with the server without revealing training sets. Unfortunately, existing research has shown that the server could extract information of the training sets from shared gradients. Besides, the server may falsify the calculated result to affect the accuracy of the trained model. To solve the above problems, we propose a privacy-preserving and verifiable federated learning scheme. Our scheme focuses on processing shared gradients by combining the Chinese Remainder Theorem and the Paillier homomorphic encryption, which can realize privacy-preserving federated learning with low computation and communication costs. In addition, we introduce the bilinear aggregate signature technology into federated learning, which effectively verifies the correctness of aggregated gradient. Moreover, the experiment shows that even with the added verification function, our scheme still has high accuracy and efficiency.**

Index Terms*—Federated learning, Privacy-preserving, Paillier encryption, Chinese Remainder Theorem, Bilinear aggregate signature

I. Introduction

With the increasing demand for high-performance algorithms for processing big data, deep learning has aroused the research interest of scholars. Nowadays, deep learning has been applied extensively in computer vision [1], speech recognition [2], disease diagnosis [3] and etc. Companies such as Google and Apple use massive amounts of data collected from multiple users to train models, and the resulting model has unprecedented accuracy.

However, it is not easy for small businesses and institutions to collect massive data due to privacy issues. Especially, privacy disclosure in users' data sharing has attracted media and govern attention. In response, many countries have passed new laws to strengthen the protection of data privacy [4]. An example is that the European Union enforced the General Data Protection Regulation (GDPR) on May 25, 2018. Although the law prohibits service providers from sharing users' data with each other, it results in data island problem. Consequently, companies or institutions only perform deep learning on their datasets, which leads to model overfitting. To solve the above problems, Google proposed federated learning [5] in 2016. The idea is to build machine learning based on datasets distributed across multiple participants. Participants train models locally and share gradients instead of training sets. Because federated learning can train highly accurate models while protecting privacy, it has been well received by industry and researchers since its inception.

However, existing studies [6,7,18,19] have shown that direct sharing gradients still create privacy issues. Besides, federated learning involves distributed participant interaction, resulting in high communication costs. Protecting privacy and reducing communication costs have become important research directions in federated learning.

A. Motivation

To ensure the security and efficiency of federated learning, we must consider the three cases as below:

First, driven by illegal interests [8], a malicious aggregation server may return incorrect aggregated result to participants [14]. For example, the aggregation server falsifies the aggregated result to affect the accuracy of the model. In such cases, participants train a bad model. For medical institutions, low accuracy models could cause medical accidents.

Second, the distributed structure and the complex deep learning model introduce high communication costs for federated learning. This could make companies or institutions less enthusiastic about federated learning.

Third, the aggregation server attempts to infer the original training data from the uploaded gradients. More and more researches prove it is feasible. The work [6] trains the mGAN-AI model to obtain participants' original training pictures. Through the analysis of the deep learning algorithm, Phong et al. [7] indicated that numerical calculations could approximate the training sets. The main reason for the privacy leak is that the participants upload their gradients to the aggregation server in clear text. Therefore, how to implement aggregation without revealing the gradient to the aggregation server is the key to protecting privacy and secure aggregation.

B. Related Work

Privacy-preserving deep learning has aroused many cryptographic groups' interests. They applied homomorphic encryption for data protection in deep learning, including predicting individual data [9] and shared gradients [7]. Especially, Li et al. [11] proposed a multi-key privacy-preserving deep learning

978-1-7281-5089-5/20/$31.00 ©2020 IEEE

scheme by combining multi-key fully homomorphic encryption [17] with a double decryption mechanism. The scheme realized the protection of participants private data by converting the two encryption methods into each other. However, due to the high computation costs of fully homomorphic encryption, the utility of the scheme was significantly reduced. Except cryptography, secure multiparty computation is also used for privacy protection in deep learning. Mohassel et al. [12] used ABY protocol [13] to design a stochastic gradient descent (SGD) algorithm, which was implemented by two servers through secure two-party computation.

After federated learning was proposed, some follow-up research focus on its security issues. Zheng et al. [10] paid close attention to malicious participants and proposed a collaborative learning scheme defending against malicious participants. They used the ADMM algorithm to train linear models but not support training more complex neural network models. Ma et al. [15] first proposed a verifiable scheme against a malicious adversary. Unfortunately, participants need to solve the discrete logarithm problem when decrypting, so the scheme is not practical. Besides, the above two schemes have high communication costs due to involving multi-party computation protocol.

From the description above, most schemes ignore verification of the correctness of the results. In addition, the scheme based on secure multiparty computation generates high communication costs.

C. Our Contributions

In this paper, we propose a privacy-preserving and verifiable federated learning scheme. Our contributions are mainly in three-fold:

1) We propose a gradient processing method, which can realize privacy-preserving and reduce the computation and communication costs effectively. Moreover, we confirm that this method will not affect the accuracy of the model.
2) We design a verification mechanism, which can detect whether the aggregation server falsifies the aggregated result. And the validity of the mechanism is proved.
3) We prove the efficiency of our scheme and the accuracy of the trained model through experiments.

Organization. The rest of this paper is organized as follows. We introduce the preliminaries in Section II. Section III explains the problem statement. And our federated learning scheme is shown in Section IV. In Section V, we give the security analysis. Followed by Section VI is the experiments of the scheme. Section VII is the conclusion of this paper.

II. Preliminaries

In this section, we will give some preliminaries related to our scheme.

A. Paillierr Encryption

Paillier encryption [20] is an additive homomorphic encryption, which has been widely used in data processing. In this paper, we define the public key as pk and the secret key as sk. The ciphertext of plaintext c is defined as $[\![c]\!]_{pk}$. And the ciphertexts $[\![c_1 + c_2]\!]_{pk}$ and $[\![c_1 \cdot c_2]\!]_{pk}$ satisfy:

$$[\![c_1 + c_2]\!]_{pk} = [\![c_1]\!]_{pk} \cdot [\![c_2]\!]_{pk} \tag{1}$$

$$[\![c_1 \cdot c_2]\!]_{pk} = [\![c_1]\!]_{pk}^{c_2} \tag{2}$$

B. Chinese Remainder Theorem

The Chinese Remainder Theorem is a method to solve the system of linear congruences. Suppose $m_1, m_2, \cdot\cdot\cdot, m_k$ are positive integers that are pairwise co-prime $gcd(m_i, m_j) = 1 (i \neq j)$. Let $M = m_1 \cdot m_2 \cdot\cdot\cdot m_k$. Then there is a unique solution to the following system of congruences in the finite field $\mathbb{F}_M$:

$$\begin{array}{c} y \equiv a_1(mod\ m_1) \\ y \equiv a_2(mod\ m_2) \\ \cdots \\ y \equiv a_k(mod\ m_k) \end{array} \tag{3}$$

the solution of equation (3) is $y \equiv (a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1} + \cdot\cdot\cdot + a_k M_k M_k^{-1}) mod\ M$. Where the $M_i = M/m_i$, and M_i^{-1} is the inverse element of M_i in the finite field $\mathbb{F}_M$.

C. Bilinear Aggregate Signature

Bilinear aggregate signature [16] can verify many signatures on different messages generated by different participants. Suppose g_1 and g_2 are generators of multiplicative cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. The bilinear map e: $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$. For a message c, the hash function h: $c \rightarrow \mathbb{G}_2$. Generally, it is a tuple of five algorithms:

1) **KeyGen**: The participant randomly selects secret key x, and computes public key g_1^x.
2) **Sign**: Input x and a message c. Output the signature σ of c: $h(c) \rightarrow h'$, $(h')^x \rightarrow \sigma \in \mathbb{G}_2$.
3) **Verify**: Input g_1^x, c and σ. Then compute $h(c) \rightarrow h'$. Accept the result if $e(g_1, \sigma) \overset{?}{=} e(g_1^x, h')$ holds.
4) **AggregateSign**: Suppose there are n participants $P_1, P_2, \cdot\cdot\cdot, P_n$ in the system. And P_i has a message c_i and its signature σ_i. Output the aggregate signature $\sigma = \prod_{i=1}^{n} \sigma_i$.
5) **AggregateVerify**: Input $h'_i = h(c_i) (i = 1, 2, \cdot\cdot\cdot, n)$ and the aggregate signature σ. Suppose P_i's secret key is x_i. Then accept the result if $e(g_1, \sigma) = \prod_{i=1}^{n} e(g_1^{x_i}, h'_i)$ holds.

III. Problem statement

In this section, we first present the architecture of our scheme, then introduce the design goals in our work.

A. Scheme Model

There are three entities in our scheme: public key generator (PKG), participants, and aggregation server. The main work PKG handles with are to generate parameters, secret keys, and functions, then these parameters are distributed to participants. Participants need to train and update models and perform encryption, signature, decryption, and verification operations. The aggregation server needs to aggregate data and perform other calculations. Figure 1 shows the architecture of our scheme. Suppose there are n participants $(P_1, P_2, \cdot\cdot\cdot, P_n)$ in our scheme, and they follow the scheme faithfully. However, due to the aggregation process is unsupervised and opaque, the aggregation server may attempt to extract information from uploaded data or falsify the aggregated results to affect the accuracy of the trained model.

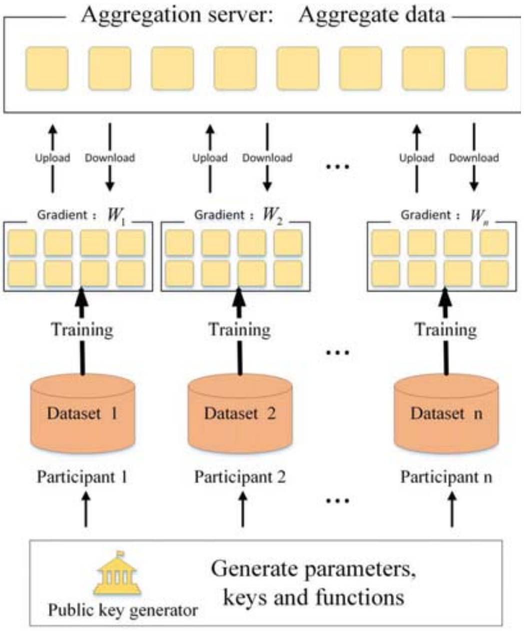

Fig. 1. Architecture of our federated learning scheme.

B. Design Goals

Our scheme needs to achieve the following security and performance guarantees to construct a practical federated learning scheme:

- ***Correctness.*** When the aggregation server and participants follow our system strictly, the final aggregated result should be correct.
- ***Verifiability.*** The aggregation server may falsify the aggregated gradient to impact model updates, and our scheme can detect this malicious behavior.
- ***Data Privacy.*** Data privacy refers to the privacy of each participant's original gradient and the aggregated gradient. The aggregation server cannot obtain these data privacy from the data it receives.

IV. The proposed scheme

Generally, our scheme can be divided into five phases. Figure 2 shows the workflow of a participant and the aggregation server in these phases. The following is a basic introduction to these phases:

1) **Initialization phase**: PKG generates and distributes secret keys to each participant.
2) **Model training phase**: $P_i(i = 1, 2, \cdot\cdot\cdot, n)$ trains model locally to get the gradient $W_i(i = 1, 2, \cdot\cdot\cdot, n)$.
3) **Data processing phase**: W_i is processed, encrypted and signed. P_i uploads ciphertext $[\![\widetilde{W_i}]\!]_{pk}$ and signature σ_i to aggregation server.
4) **Aggregation phase**: The aggregation server aggregates ciphertexts and signatures to get $[\![\widetilde{W}]\!]_{pk}$ and σ.
5) **ResDecVerify phase**: Each participant downloads the aggregated results to decrypt and verify. Finally updating the model parameter $\mathbf{M}$.

A. Initialization Phase

In this phase, all participants agree on the trained neural network model architecture. The PKG generates parameters, keys and functions, then distributes them to participants. The parameters include initial model parameter $\mathbf{M}$, learning rate η, k positive integers and generator g_1 of a multiplicative cyclic group $\mathbb{G}_1$. The k positive integers $m_1, m_2, \cdots, m_k$ are pairwise co-prime $gcd(m_i, m_j) = 1(i \neq j)$. And they are large enough so that the aggregated result does not generate overflow error. Meanwhile we set $M = \prod_{i=1}^{k} m_i$. The keys include keys of paillier encryption (pk, sk) and bilinear aggregate signature x. Note that all participant receive the same keys, then calculate g_1^x. In addition, the functions include a homomorphic hash function h and a bilinear map e: $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$. For two messages c_1 and c_2, the function h satisfies: $h(c_1) \in \mathbb{G}_2$ and $h(c_1 + c_2) = h(c_1) \cdot h(c_2)$. What's more, it is important to emphasize that x, g_1^x, sk and h are confidential to the aggregation server.

B. Model Training Phase

In this phase, each participant uses SGD algorithm to train the model locally. The gradient of P_i is $W_i = [W_i^{(1)}, W_i^{(2)}, \cdot\cdot\cdot, W_i^{(l)}](i = 1, 2, \cdot\cdot\cdot, n)$, where $W_i^{(j)}(j = 1, 2, \cdot\cdot\cdot, l)$ is the vector representing the jth layer parameter of gradient W_i and l is the number of layers. Define the length of $W_i^{(j)}$ as $\left|W_i^{(j)}\right|$ and the size of gradient as $|W_i|$. Obviously, $|W_i|$ is equal to $|\mathbf{M}|$. We set $|\mathbf{M}| = D$ and $\left|W_i^{(j)}\right| = D_j$. In our scheme, the final aggregated result is $W = \sum_{i=1}^{n} W_i$.

C. Data Processing Phase

In this phase, each participant processes and encrypts private gradient. Take P_i as an example. First, P_i uses the Chinese Remainder Theorem to process private gradient W_i, then utilizes bilinear aggregate signature to sign the processed data while encrypting it by Paillier encryption. We will present this phase in details next.

First, participant P_i processes the private gradient W_i layer by layer. P_i partitions $W_i^{(j)}$ into $r = \left\lceil \frac{D_j}{k} \right\rceil$ pieces equally $W_i^{(j)} = w_1|| \cdots ||w_r$, where $\lceil z \rceil$ is the smallest integer lager than or equal to z and $\cdot||\cdot$ denotes vectors concatenation. If the length of $W_i^{(j)}$ is not divisible by k, $W_i^{(j)}$ should be padded

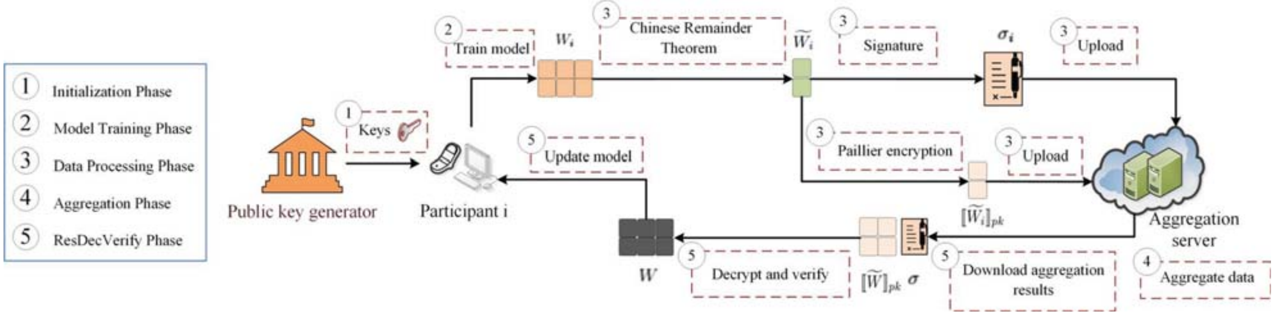

Fig. 2. The workflow of a participant and the aggregation server in different phases.

with 0. Suppose the vector $w_i = [a_1^{(i)}, a_2^{(i)}, \cdots, a_k^{(i)}]$, $(i = 1, 2, \cdots, r)$, then P_i solves the system of linear congruences:

$$\begin{array}{l} y \equiv a_1^{(i)}(mod\ m_1) \\ y \equiv a_2^{(i)}(mod\ m_2) \\ \cdots \\ y \equiv a_k^{(i)}(mod\ m_k) \end{array} \tag{4}$$

where $y = \widetilde{w_i} \in \mathbb{F}_M$ is the unique solution of equation (4). By processing W_i in this way, we will obtain the processed data $\widetilde{W_i}$. Figure 3 shows parameter $W_i^{(j)}$ processing using Chinese Remainder Theorem.

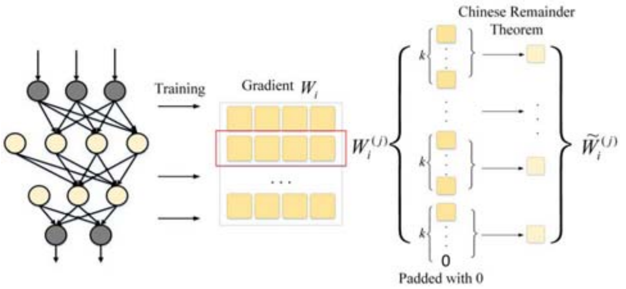

Fig. 3. The schematic diagram of processing gradient W_i using Chinese Remainder Theorem, where $W_i^{(j)}$ is the jth layer parameter of W_i.

We can conclude that $\left|\widetilde{W_i}\right| \approx \frac{D}{k}$. Then P_i encrypts $\widetilde{W_i}$ to get ciphertext $[\![\widetilde{W_i}]\!]_{pk}$. Compared to directly encrypting the gradient W_i, since $\left|\widetilde{W_i}\right|$ is about $\frac{1}{k}$ of $|W_i|$, the encryption operation is correspondingly reduced. Meanwhile P_i calculates the signature $\sigma_i = (h(\widetilde{W_i}))^x$. Finally, P_i uploads ciphertext $[\![\widetilde{W_i}]\!]_{pk}$ and signature σ_i to the aggregation server. Due to the processing of the Chinese Remainder Theorem, the size of the ciphertext is reduced, and the communication costs are correspondingly reduced, which is about $\frac{k-1}{k}$ lower than the case without processing.

D. Aggregation Phase

The aggregation server receives ciphertexts and signatures from all participants, then aggregates them. In this phase, the aggregation server needs to calculate:

$$[\![\widetilde{W}]\!]_{pk} = \prod_{i=1}^{n}[\![\widetilde{W_i}]\!]_{pk}, e(g_1, \sigma) = e(g_{1,} \prod_{i=1}^{n} \sigma_i) \tag{5}$$

where the former of equation (5) realizes gradients aggregation, the latter realizes signatures aggregation. In this phase, the aggregation server may falsify the aggregated results. After data aggregation, each participant downloads the above two results from the aggregation server.

E. ResDecVerify Phase

In this phase, firstly, each participant decrypts ciphertext $[\![\widetilde{W}]\!]_{pk}$ to get plaintext $\widetilde{W}$, then verifies whether the aggregated result $\widetilde{W}$ is reliable. Accept $\widetilde{W}$ if the follow equation holds:

$$e(g_1, \sigma) = e(g_1^x, h(\widetilde{W})) \tag{6}$$

Otherwise, reject.

Since processing the gradient by Chinese Remainder Theorem before encryption, participant needs to perform the following calculation after verifying $\widetilde{W}$ is reliable:

$$\widetilde{W}\ mod\ m_i (i = 1, 2, \cdots, k) \tag{7}$$

Finally, participants combine the results calculated by equation (7) to obtain the final aggregated gradient W. At the end of the phase, they update the model parameters $\mathbf{M}$=$\mathbf{M}$−$\frac{\eta}{n} \times W$. If the termination condition is not satisfied, participants continue to train model locally in preparation for the next round of federated learning.

V. Security Analysis

This section gives the security analysis of scheme, which provides the guarantee of correctness, verifiability and privacy.

A. Correctness

If participants and the aggregation server follow our scheme loyally, participants can get the correct aggregated gradient W.

According to equation (1), we can get:

$$[\![\widetilde{W}]\!]_{pk} = \prod_{i=1}^{n}[\![\widetilde{W_i}]\!]_{pk} = [\![\sum_{i=1}^{n} \widetilde{W_i}]\!]_{pk} \tag{8}$$

Decrypting it and getting plaintext $\widetilde{W} = \sum_{i=1}^{n} \widetilde{W_i}$. Suppose $\widetilde{w}_i$, $\widetilde{w}_j \in \mathbb{F}_M (i \neq j)$ are respectively the elements of $\widetilde{W_i}$, $\widetilde{W_j}$

corresponding to the same position. According equation (4), then $\widetilde{w}_i + \widetilde{w}_j \in \mathbb{F}_M$ satisfies:

$$\widetilde{w}_i + \widetilde{w}_j \equiv a_t^{(i)} + a_t^{(j)} (mod\ m_t), (t = 1, 2, \cdot \cdot \cdot, k) \tag{9}$$

Note that $a_t^{(i)}$ is an element in the P_i's original gradient W_i. Therefore, calculating as equation (7) and combining these results, participants can get the correct aggregated gradient $W = \sum_{i=1}^{n} W_i$.

B. Verifiability

The aggregation server may falsify the aggregated gradient to affect the accuracy of the trained model. This malicious behavior can be detected in our scheme. If the aggregation server attempts to falsify $[\![\widetilde{W}]\!]_{pk}$ without being detected, it should modify σ accordingly to make the following equation hold:

$$\begin{aligned} e(g_1, \sigma) &= e(g_1, \prod_{i=1}^{n} (h(\widetilde{W_i}))^x) \\ &= e(g_1, (\prod_{i=1}^{n} h(\widetilde{W_i})))^x \\ &= e(g_1^x, h(\sum_{i=1}^{n} \widetilde{W_i})) \\ &= e(g_1^x, h(\widetilde{W})) \end{aligned} \tag{10}$$

Next, we consider whether the aggregation server could evade detection by forging signatures. Review IV-D, the aggregation server can obtain ciphertext $[\![\widetilde{W_i}]\!]_{pk}$ and signature σ_i. Note that h and x are confidential to the aggregation server. Hence, for a false data $[\![\widetilde{W_i'}]\!]_{pk}$, the aggregation server cannot forge its signature $\sigma_i' = (h(\widetilde{W_i'}))^x$. Likewise, the aggregation server cannot forge the aggregated signature $\sigma' = \sum_{i=1}^{n} \sigma_i'$. And the right-hand side of equation (10) contains g_1^x, which is also confidential to the aggregation server. Therefore, our scheme is reliable in verifying the correctness of the aggregated gradient.

C. Data Privacy

Through references [6,7,18,19], it can be known that the main cause of privacy leakage is the aggregation server can obtain the private gradient of participants. Our scheme can ensure that the aggregation server does not get the gradient plaintext. Meanwhile, we consider that the aggregated gradient belongs to the private property of the participants and should also be protected. In our scheme, each participant encrypts his own gradient using Paillier encryption before uploading. Paillier et al. [20] have proved that Paillier cryptosystem is a semantically secure encryption algorithm. And sk is confidential to the aggregation server. So the aggregation server cannot calculate $\widetilde{W_i}$, $\widetilde{W}$ through $[\![\widetilde{W_i}]\!]_{pk}$, $[\![\widetilde{W}]\!]_{pk}$. In addition, because hash function is irreversible and x is confidential to the aggregation server, it cannot calculate $\widetilde{W_i}$ through $\sigma_i = (h(\widetilde{W_i}))^x$.

Likewise, each participant cannot calculate others' private gradient through the results of equation (5). So the privacy security of $\widetilde{W_i}$ and $\widetilde{W}$ can be guaranteed in our scheme.

VI. Experiment

In this section, we test the accuracy of the trained model and the time costs of our scheme. The experiments prove that even with the added verification function, our scheme still has high accuracy and efficiency.

A. Experiment Environment and Datasets

The simulation experiment is conducted on Windows Server 2008 R2 Enterprise, Intel Xeon CPU E5-2640, 2.60 GHz, and 64 GB memory. The efficiency of the scheme is related to the parameter k. However, the selected k non-negative integers should be large enough, so k should not be too large. We set k as 7 and the size of the key as 64bit, the channel as 1Gbps, and the number of participants as 20. The training set is the MNIST dataset, which is composed of 70,000 images of 2828 pixels, handwritten digital grayscale images with labels, among which 60,000 are training data, and 10,000 are test data.

B. Accuracy

We compare the accuracy of three schemes: our scheme, original federated learning [5] (Original-FL), and scheme [7] (LWE-FL). Original-FL does not process the shared gradients, and LWE-FL uses Learning with Errors (LWE) encryption to encrypt the shared gradients. Our experiment takes a multi-layer perceptron (MLP) as the training model. And the form of the MLP is 748(input)-128(hidden)-64(hidden)-10(output). Figure 4 shows the experimental results. After multiple rounds of training, the model of our scheme (blue line) can achieve about 97% accuracy, slightly lower than the accuracy of Original-FL (green line), which is 98%. And LWE-FL (red line) has the lowest accuracy, approximately 95%.

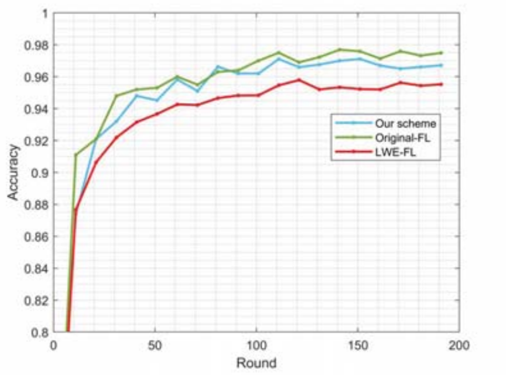

Fig. 4. The accuracy of the models trained by three schemes.

C. The Costs of Our Scheme

This section compares the computation and time costs of two schemes: our scheme and LWE-FL. Since Original-FL does not process the gradients, its costs are almost 0. But existing researches [6,7,18,19] have proven that due to gradients are not processed, Original-FL causes privacy leaks. So we do not make an experimental comparison of it.

According to the solution of equation (3), using the Chinese remainder theorem to process gradients is equivalent to simply multiplying vectors. And the costs of equation (7) are almost negligible. Therefore, the computation costs added by the data processing is much less than that of the reduced Paillier encryption which contains exponential computation.

TABLE I
COMPUTATION COSTS OF A PARTICIPANT

Scheme	Computation costs
Our scheme	$\frac{D}{k}\cdot$ (***Enc1***+ ***Dec1***+ ***Crt***)+***Exp***+2·***Hash***+***Bil***+ $D\cdot$ (***Mod***+***Add***+***Mul***)
LWE-FL [7]	$D\cdot$(***Enc2***+ ***Dec2***+***Mul***)

TABLE II
COMPUTATION COSTS OF THE AGGREGATION SERVER

Scheme	Computation costs
Our scheme	$((n-1)\frac{D}{k}+(n-1))\cdot$***Mul***+***Bil***
LWE-FL [7]	$D\cdot$***Add***

The computation costs of our scheme and LWE-FL are shown in Table I and Table II. In these tables, ***Enc1***, ***Dec1***, ***Enc2***, ***Dec2***, ***Crt***, ***Exp***, ***Hash***, ***Mul***, ***Bil***, ***Mod*** and ***Add*** represent Paillier encryption, Paillier decryption, LWE encryption, LWE decryption, Chinese Remainder Theorem, exponentiation, hash operation, multiplication, bilinear map, modular, and addition respectively. And D is the number of model parameters. Figure 5 shows the time costs of the two schemes in a round of training under the different number of model parameters. We can conclude that our scheme (blue line) is more efficient than LWE-FL (green line). Moreover, note that LWE-FL cannot verify the correctness of aggregated gradient.

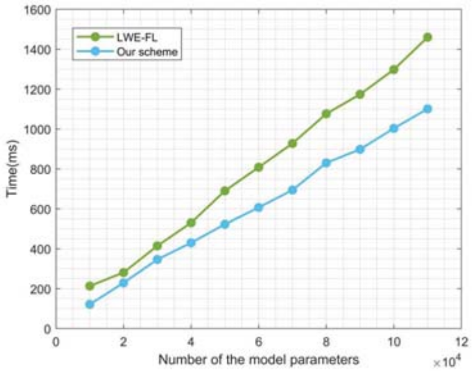

Fig. 5. The time costs of our scheme and scheme [7] (LWE-FL) under different number of model parameters

VII. Conclusion

In this paper, we propose a verifiable and privacy-preserving federated learning scheme. If the aggregation server falsify the aggregated gradient, our scheme can detect this malicious behavior. And the aggregation server cannot infer private gradient from the shared data. In addition, we present a special gradient processing method to reduce communication and computation costs. And the experiment proved that our scheme has high accuracy and efficiency.

Acknowledgment

This work is supported by National Natural Science Foundation of China (61572255, 61941116), Six talent peaks project of Jiangsu Province, China (XYDXXJS-032), Jiangsu Key Laboratory of Big Data Security & Intelligent Processing, NJUPT(BDSIP1909), and CERNET Innovation Project (NGII20190405).

References

[1] Y. LeCun, K. Kavukcuoglu, C. Farabet, “Convolutional networks and applications in vision,” ISCAS, Paris, pp. 253-256, May 2010.

[2] A. Graves, A. Mohamed, G. Hinton, “Speech recognition with deep recurrent neural networks,” ICASSP, Vancouver, pp. 6645-6649, May 2013.

[3] Y. Liu, J. Ling, Z. Liu, “Finger vein secure biometric template generation based on deep learning,” Soft Computing, vol. 22, no. 7, pp. 2257-2265, 2018.

[4] S. Yu, “Big privacy: Challenges and opportunities of privacy study in the age of big data.” IEEE Access, vol. 4, pp. 2751-2763, 2016.

[5] H. B. McMahan, E. Moore, D. Ramage, et al., “Federated learning of deep networks using model averaging,” CoRR, vol. abs/1602.05629, 2016.

[6] Z. Wang, M. Song, Z. Zhang, et al., “Beyond Inferring Class Representatives: User-Level Privacy Leakage From Federated Learning,” INFOCOM, Paris, pp. 2512-2520, April 2019.

[7] L. T. Phong, Y. Aono, T. Hayashi, et al., “Privacy-preserving deep learning via additively homomorphic encryption,” IEEE Transactions on Information Forensics and Security, vol. 13, no. 5, pp. 1333-1345, 2017.

[8] B. Kuang, A. Fu, S. Yu, et al., “ Esdra: An efficient and secure distributed remote attestation scheme for iot swarms,” IEEE Internet of Things Journal, vol. 6, no. 5, pp. 8372-8383, 2019.

[9] R. Gilad-Bachrach, N. Dowlin, K. Laine, et al., “Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy,” ICML, New York, pp. 201-210, June 2016.

[10] W. Zheng, R. A. Popa, J. E. Gonzalez, et al., “Helen: Maliciously Secure Coopetitive Learning for Linear Models,” S&P, San Francisco, May 2019.

[11] P. Li, J. Li, Z. Huang, et al., “Multi-key privacy-preserving deep learning in cloud computing,” Future Generation Computer Systems, vol. 74, pp. 76-85, 2017.

[12] P. Mohassel, Y. Zhang, “Secureml: A system for scalable privacy-preserving machine learning,” S&P, San Jose, pp. 19-38, May 2017.

[13] D. Daniel, T. Schneider, M. Zohner, “ABY-A Framework for Efficient Mixed-Protocol Secure Two-Party Computation,” NDSS, February 2015.

[14] A. Fu, Z. Chen, Y. Mu, W. Susilo, Y. Sun, J. Wu, “Cloud-based Outsourcing for Enabling Privacy-Preserving Large-scale Non-Negative Matrix Factorization,” IEEE Transactions on Services Computing, DOI: 10.1109/TSC.2019.2937484, August. 2019.

[15] X. Ma, F. Zhang, X. Chen, et al., “Privacy preserving multi-party computation delegation for deep learning in cloud computing,” Information Sciences, vol. 459, pp. 103-116, 2018.

[16] D. Boneh, C. Gentry, B. Lynn, et al., “Aggregate and verifiably encrypted signatures from bilinear maps,” Advances in Cryptology-EUROCRYPT, Warsaw1, pp. 416-432, May 2002.

[17] P. Mukherjee, D. Wichs, “Two round multiparty computation via multi-key FHE,” Advances in Cryptology EUROCRYPT, Berlin, pp. 735-763, 2016.

[18] J. Hayes, L. Melis, G. Danezis, E. De Cristofaro, “LOGAN: Membership inference attacks against generative models,” Proceedings on Privacy Enhancing Technologies, vol. 2019, no. 1, pp. 133-152, 2019.

[19] L. Melis, C. Song, E. D. Cristofaro, et al., “Exploiting unintended feature leakage in collaborative learning,” S&P, San Francisco, May 2019.

[20] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” International Conference on the Theory and Applications of Cryptographic Techniques, pp. 223-238, 1999.