IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

VPPFL: Verifiable Privacy-Preserving Federated Learning in Cloud Environment

HUIYONG WANG[1,2,4], TENGFEI YANG[1,3], YONG DING[4,5], SHIJIE TANG [6]and Yujue Wang,[7]

[1]School of Mathematics and Computing Science, Guilin University of Electronic Technology, Guilin 541004, China
[2]Center for Applied Mathematics of Guangxi, Guilin University of Electronic Technology, Guilin 541004, China
[3]Guangxi Engineering Research Center of Industrial Internet Security and Blockchain, Guilin university of electronic technology, Guilin 541004, China
[4]Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, Guilin 541004, China
[5]Institute of Cyberspace Technology, HKCT Institute for Higher Education, Hong Kong, China
[6]School of Electronic Engineering and Automation, GuiLin University of Electronic Technology, Guilin, 541004, China
[7]Hangzhou Innovation Institute, Beihang University, Hangzhou, 310052, China

Corresponding author: Shijie Tang (e-mail: tangsj@guet.edu.cn).

This article is supported in part by the Guangxi Natural Science Foundationunder grant 2023GXNSFAA026236, the National Natural Science Foundation of China under project 61962012, and the National Key R & D Program of China under project2020YFB1006003, and the Science and Technology Project of Guangxi (Guike AD23023002).'

ABSTRACT As a distributed machine learning paradigm, federated learning has attracted wide attention from academia and industry by enabling multiple users to jointly train models without sharing local data. However, federated learning still faces various security and privacy issues. First, even if users only upload gradients, their privacy information may still be leaked. Second, when the aggregation server intentionally returns fabricated results, the model's performance may be degraded. To address the above issues, we propose a verifiable privacy-preserving federated learning scheme VPPFL against semi-malicious cloud server. We use threshold multi-key homomorphic encryption to protect local gradients, and construct a one-way function to enable the users to independently verify the aggregation results. Furthermore, our scheme supports a small portion of users dropout during the training process. Finally, we conduct simulation experiments on the MNIST dataset, demonstrating that VPPFL can correctly and effectively complete training and achieve privacy protection.

INDEX TERMS Privacy protection; Federated learning; Verifiable; Threshold multi-key homomorphic encryption.

I. INTRODUCTION

RECENTLY machine learning technology has played a key role in numerous fields. For example, it has achieved significant results in medical prediction [1] [2], autonomous driving technolog-ies [3] [4], and image recognition [5]. In machine learning, data privacy and security are key concerns. For example, medical data often contains a lot of sensitive personal information. If an unauthorized third party accesses medical data, it can lead to a serious privacy breach that affects the interests of patients [6] [7]. Additionally, since traditional machine learning typically operates on unencrypted data, this also poses a serious risk of privacy leakage.

To address these issues, Google proposed federated learning in 2016 [8]. In federated learning, users only need to share the local gradients instead of original valuable data. This method conveniently utilizes sensitive information while mitigating the risk of privacy breaches that may arise from collecting data from different users.

However, current research indicates that even if users only upload gradient information, their privacy could still be compromised [9] [10]. Attackers might exploit vulnerabilities in cloud server to reveal specific attributes of training samples, or fabricate aggregation results to induce users to leak more valuable information. In some extreme cases, attackers could even use the leaked data to reconstruct users' original data. On the other hand, motivated by illicit profits, malicious cloud server might return incorrect aggregation results to users. For example, in order to reduce its computation costs, the cloud server may use a more simplified and less accurate model to process the uploaded gradients, or even directly modify the aggregation results [11] [12].

To address these issues, we propose the Verifiable Privacy-Preserving Federated Learning (VPPFL). Our contributions are outlined as follows:

- We design a verifiable federated learning mechanism to

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3472467

deal with the semi-malicious cloud server. This scheme enables users to independently verify the aggregation results without the intervention of a trusted third party, and can effectively prevent collusion attacks initiated by the cloud server in collaboration with a small portion of users.

- Our scheme employs multi-key threshold homomorphic encryption to protect users' privacy data, and allows a small portion of users dropout during training without adding additional burden to the server. Even if several users fail to upload their data, the training process won't be interrupted.
- We provide security analysis and simulation experiments to validate the security and efficiency of VPPFL.

The rest of this paper is organized as follows. Section 2 reviews the current relevant research and the basic concepts and techniques involved. Section 3 introduces the system model and security requirement. Section 4 elaborates in detail the teshnical specific of VPPFL. We analyze the security and performance of VPPFL in Section 5. Section 6 presents the experimental analysis results. Finally, section 7 concludes the work.

II. RELATED WORK AND RELEVANT CONCEPTS AND TECHNOLOGIES

A. RELATED WORK

When constructing privacy-preserving federated learning, there are two commonly used cryptographic tools, i.e. differential privacy [13] and homomorphic encryption [14].

Differential privacy is a privacy protection technique with provable security, which protects data by adding random noise. In 2006, Microsoft's Dwork [15] first proposed the differential privacy technology, and later in [16], a new differential privacy mechanism Propose-Test-Release(PTR) was used to achieve high-quality differential privacy results. Geyer et al. [17] used differential privacy for the first time in federated learning to protect participants' data by adding Gaussian noise on the server side. K. Wei et al. [18] employed a local differential privacy strategy during the local model updates of deep neural networks. They protect the local gradient by adding noise before uploading the local model. However, this method does not take users dropout into consideration.

Homomorphic encryption is now widely used in the construction of privacy-preserving federated learning. Phong et al. [19] introduced homomorphic encryption in the asynchronous stochastic gradient descent training. However, all users utilize the same private key, leading to a potential risk: if the server colludes with some users, the data privacy of other users cannot be guaranteed. B. Wang et al. [20] in their research adopted homomorphic encryption to protect users' local data and implemented access control to verify the credibility of user identities, effectively defending against threats from internal attacks. J. Ma et al. [21] used multi-key homomorphic encryption to encrypt the model before updating the local gradients. Decryption requires the collaborative participation of all users to prevent unauthorized access to participant's data. As a result, if users dropout in the middle of the training process, decryption cannot be achieved, which is impractical for real-world federated learning.

Recently, the research community has proposed various schemes to address the data integrity challenge in federated learning. G. Xu et al. [22] introduced an innovative verifiable privacy-preserving federated learning architecture. By employing homomorphic hash functions and zero-knowledge proof, they construct a verifiable and secure aggregation mechanism. X. Guo et al. [23] modified this framework to reduce the communication cost, while they also pointed out that if malicious cloud server colluded with users, the scheme in [22] would still face certain security vulnerabilities. L. Lin and X. Zhang [24] used differential privacy and one-way function to allow users to verify aggregation results returned by a lazy server, but the approach does not support user dropout during training. Y. Ren et al. [25] adopted linear homomorphic hash functions and digital signature to achieve traceable verification of aggregation results and identification of erroneous cycles. However, this approach inevitably increases communication cost.

B. CONCEPTS AND TECHNOLOGIES

We now introduce some relevant conceptions and technologies. Some of the symbols used in this paper are listed in Table 1.

TABLE 1. List of symbols

Symbol	Meaning
N	number of users
K	security parameter
P_n	the n-th user
θ	learning rate
g_n	gradient of the n-th user
ω	weigth
D_n	dataset of the n-th user
D_n^j	dataset of the n-th user in the j-th iteration
t	number of users for collaborative decryption
$sum(g)$	If g is a vector, it represents the sum of vector elements; If g is a matrix, it represents the sum of all elements in the matrix

1) Federated learning

Different from traditional machine learning, federated learning has made significant strides in protecting user's privacy. In federated learning, users do not upload personal data, but only need to share the local gradients, significantly reducing the risk of personal information leakage. As shown in Figure 1, users upload these gradients to the cloud server, which then aggregates the data and feeds the results back to users. By this method, users and server collaborate to cultivate a comprehensively optimized global model, ensuring the security of personal data while achieving efficient model training.

2) Neural network

We now introduce a classic deep neural network - fully connected neural network (FCNN). Figure 2 shows the ar-

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3472467

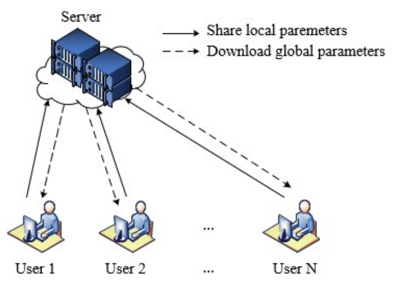

FIGURE 1. Federated learning architecture.

chitecture of FCNN. The neurons in each layer are densely connected to the neurons in the preceding and following layers by weight ω.

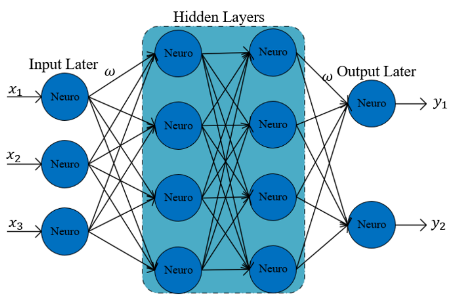

FIGURE 2. The architecture of FCNN.

FCNN can be represented by $f(x, \omega) = \tilde{y}$, where x is the input and $\tilde{y}$ is the corresponding output. Assuming the entire dataset $D = \{< x_i, y_i >, i = 1, \cdots, T\}$, the loss function be defined as:

$$L_f(D, \omega) = \frac{1}{|D|} \sum_{(x_i, y_i) \in D} L_f((x_i, y_i), \omega), \tag{1}$$

where $L_f((x_i, y_i), \omega) = l(y, \tilde{y}) = ||y, \tilde{y}||_2$.

The objective of neural network training is to achieve an optimal set of parameters ω, which minimizes the value of the loss function. To achieve this goal, we use Algorithm 1: the mini-batch gradient descent method (SGD).

input : Dataset $D = \{(x_i, y_i) : i = 1, \ldots, N\}$,
Learning rate θ,
Loss function
$L_f(D, \omega) = \frac{1}{|D|} \sum_{(x_i, y_i) \in D} L_f((x_i, y_i), \omega)$.
output: The optimal model parameters ω.

Randomly select an initial ω^0;
At the j-th iteration, randomly select a small batch of data $D^j \subseteq D$;
for $(x_i, y_i) \in D^j$ do
Calculate $g^j_{(x_i, y_i)} \leftarrow \nabla L_f((x_i, y_i), \omega_j)$;
end
Calculate $g^j_* \leftarrow \frac{1}{|D^j|} \sum_{(x_j, y_j) \in D^j} g^j_{(x_i, y_i)}$;
Update weight $\omega^{j+1} \leftarrow \omega^j - \theta \cdot g^j_*$;
until convergence is satisfied;
return ω.

Algorithm 1: SGD

3) Threshold paillier cryptosystem

In VPPFL, we use the threshold Paillier cryptosystem [26] to construct a secure framework since it has two important features: 1) Threshold property: Each user cannot decrypt the ciphertext alone, at least t users are need to work together to decrypt the ciphertext; 2) Homomorphic additivity: Multiplying ciphertexts equals adding plaintexts, enabling operations on plaintexts through ciphertext calculations. These two features provide sufficient functionality and privacy protection for our scheme.

In the threshold Paillier cryptosystem, the public key $pk = (G, K)$ is openly shared with all participants, where $G = 1 + K$, $K = pq$, with p and q are two large primes. The private key is split into N keys, denoted as $(sk_1, sk_2, \ldots, sk_N)$, with each user holding his own private key.

For a plaintext M, encrypting it using the public key pk will yield the ciphertext

$$c = E_{pk}(M) = G^M x^K \ mod \ K^2, \tag{2}$$

where x is a random positive integer in the multiplicative group Z_{K^2}.

This cryptosystem has homomorphic additivity, which can be described as: $c = E_{pk}(M_1 + M_2) = G^{(M_1+M_2)}(x_1 x_2)^K \ mod \ K^2 = E_{pk}(M_1) \cdot E_{pk}(M_2)$,where M_1 and M_2 are the two plaintexts, and x_1 and x_2 are random positive integers in $\mathbb{Z}^*_{K^2}$.

4) One-way function

The generation of one-way function is based on hardness of the irreversible logarithm problem, which ensures that the semi-malicious cloud server cannot infer the user's privacy information through the function values of gradients. The specific process is as follows:

Assume a is a generator of order k, and b is a large prime number. Construct a one-way function h: $\mathbb{Z} \rightarrow \mathbb{Z}_p$ as:

$$h(M) = a^M \ mod \ b, \ M \in \mathbb{Z}. \tag{3}$$

It satisfies homomorphic addition, i.e.$\forall x_1, x_2 \in \mathbb{Z}, h(x_1) = a^{x_1} \ mod \ b, h(x_2) = a^{x_2} \ mod \ b$, then $h(x_1 + x_2) = a^{x_1+x_2} \ mod \ b = (a^{x_1} \mod b) \cdot (a^{x_2} \mod b) \ mod \ b = h(x_1)h(x_2)$.

III. SYSTEM MODEL AND SECURITY REQUIREMENTS

A. SYSTEM ARCHITECTURE

As shown in Figure 3, our system consists of three parts: semi-malicious cloud server (CS), trusted authority (TA) and semi-honest users (Users).

Semi-malicious cloud server (CS): The main task of CS is to aggregate the gradients uploaded by users, broadcast

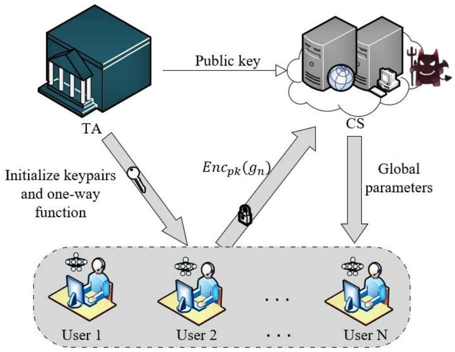

FIGURE 3. System architecture.

the function values of gradients to all users, decrypt the aggregation results, and send them to users. We require CS to only obtain ciphertexts and the final aggregation results, without knowing any other information.

Trusted authority (TA): TA is an authoritative and trustworthy entity (e.g. a government agency). It does not collude with any party. Its main task is to initialize model parameters, generate a one-way function, and generate key pairs for all users. Then, it sends the one-way function and key pairs to users through secure channel and broadcasts public key. After that, unless there is a dispute, it will go offline.

Semi-honest users (Users): Users are the owner of the data, who participate in the training process, and ultimately obtain the global model. Each user sends his encrypted local gradient and the function value of gradient to CS during each round, and cooperates with CS to decrypt the aggregation results. Finally, all users verify the aggregation results returned by CS.

B. THREAT MODEL

Semi-malicious CS attack: CS may deceive users by reducing the gradient aggregation of one or more users in order to save costs.

Half-honest user attack: He may try to use the information he has to infer the private data of other users.

Collusion attacks: There may be collusion attacks between a small number of users and CS, and the private information of other users can be inferred by sharing information such as model parameters.

External malicious attack: There is a malicious adversary, denoted as $\mathbb{A}$, who will use any means to obtain useful information from users. For example, $\mathbb{A}$ can launch active attacks by infiltrating CS, modifying or injecting false data, returning incorrect aggregation results to deceive users into revealing more privacy data.

C. SECURITY OBJECTIVES

We aim to propose an efficient, secure, and verifiable privacy-preserving federated learning scheme. Specifically, the following objectives should be achieved:

1) Privacy of user's data: No entity other than the user himself should be able to access sensitive information of the user, including an external adversary $\mathbb{A}$ and CS.

2) Every user should be able to independently verify the aggregation results. If CS returns incorrect aggregation results, users should have the right to deny the results and request CS to reaggregate the results.

3) The scheme should allow a small portion of users to join or dropout the training process without interrupting the overall training of the model.

IV. VPPFL

In this section, we provide the detailed design of VPPFL. Our scheme consists of four main stages: 1) Initialization; 2) Encryption; 3) Decryption and 4) Verification. Figure 4 illustrates the process flow of VPPFL.

1) Initialization

TA takes on the role of initializing the system parameters and generating key pairs. The specific process for generating the public and private keys is as follows, as shown in Algorithm 2.

Parameter Generation: The parameters that need to be initialized include the global weight ω, learning rate θ, training epoch, safety parameter L, and the one-way function h.

Key generation and distribution: First, TA randomly generates two large prime numbers $p = 2p' + 1$, $q = 2q' + 1$, where $p', q' < L$. Second, TA generates the RSA modulus $K = pq$, ensuring that $\gcd(K, \psi(K)) = 1$. Then, TA randomly selects $\beta \in \mathbb{Z}_K^*$ and calculates $m = p'q'$ and $\Delta = N!$. Next, TA disguise m as $\alpha = m\beta \ mod \ K$.

TA sets public key $pk = (K, G, \alpha)$ and private key $SK = \beta m$, where $G = K+1$. TA splits the private key SK as follows: selects t random numbers $a_1, a_2, \dots, a_t \in \{0, 1, \dots, Km - 1\}$, then generates the polynomial $f(x) = \beta m + a_1 x + \dots + a_t x^{t-1} \ mod \ Km$. Finally, TA sends $f(n)$ to each participant $P_n (1 \leq n \leq N)$ through secure channel.

2) Encryption

Each user $P_n (1 \leq n \leq N)$ encrypts his own gradient: for the gradient vector $g_n = [g_{n1}, g_{n2}, \dots, g_{nm}]$, P_n chooses a random number $x_n \in Z_K^*$, uses the public key pk to calculate ciphertext $Enc_{pk}(g_n) = G^{g_n} x_n^K \ mod \ K^2$, and the one-way function value of gradient $h(sum(g_n)) = a^{sum(g_n)} \ mod \ b$.

Each user P_n sends the ciphertext $Enc_{pk}(g_n)$ and the one-way function value of his gradient $h(sum(g_n))$ to CS. CS broadcasts the received one-way function values of gradients $\{h(sum(g_1)), h(sum(g_2)), \dots, h(sum(g_N))\}$, and aggregates the ciphertexts to obtain the encrypted gradient ciphertext,

$$c = \prod_{n=1}^{N} Enc_{pk}(g_n). \tag{4}$$

3) Decryption

For the ciphertext c, CS randomly selects $t (1 \leq t \leq N)$ users to send decryption requests. Suppose the selected participants form a set S. The selected participant computes the

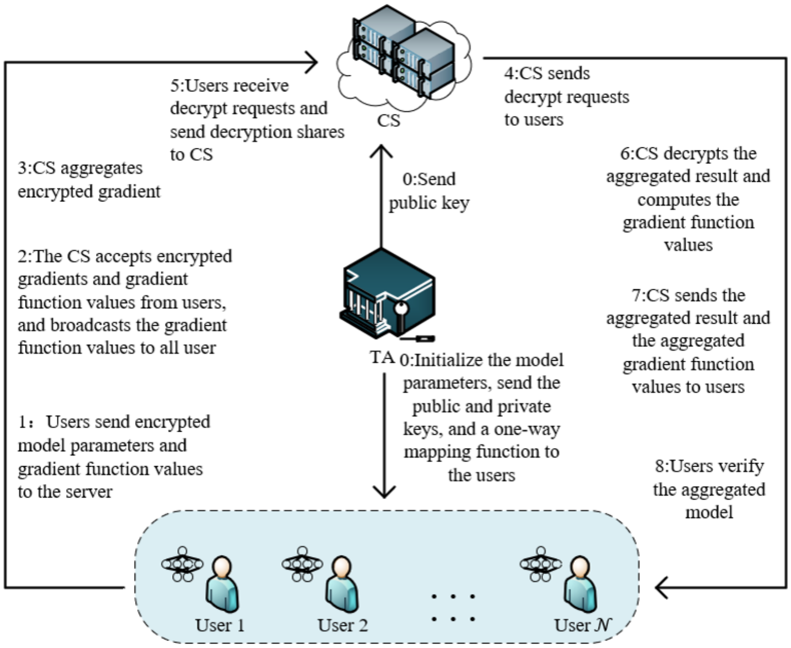

FIGURE 4. The process flow of VPPFL.

output: Private key $(sk_1, sk_2, \ldots, sk_N)$.

Randomly generate two prime numbers p' and q', where $p', q' < L$;
Calculate $p = 2p' + 1$, $q = 2q' + 1$, p and q are also prime numbers;
Calculate RSA modulus $K = pq$, and ensure that $\gcd(K, \varphi(K)) = 1$, where $\varphi(K) = (p-1)(q-1)$;
Calculate decryption key $m = p'q' = \frac{\varphi(K)}{4}$;
Randomly choose a $\beta \in \mathbb{Z}_K^*$, calculate $\alpha = m\beta \bmod K$;
Calculate $\Delta = N!$;
Set private key $SK = \beta m$, public key $pk = (K, G, \alpha, \Delta)$, where $G = 1 + K$;
Split the private key SK: select t random numbers $a_1, a_2, \ldots, a_{t-1} \in \{0, 1, \ldots, Km-1\}$, generate a polynomial $f(x) = \beta m + a_1 x + \ldots + a_t x^{t-1} \bmod Nm$, calculate $sk_n = f(n)$ and send sk_n to the corresponding participant $P_n (1 \leq n \leq N)$.

Algorithm 2: KGA(TA)

decryption share $s_n = c^{2\Delta sk_n}\ mod\ K^2$ and sends it to CS. CS can then compute the aggregation results

$$g_* = L(\prod_{n \in S} s_n^{2\mu_n}\ mod\ K^2) \times \frac{1}{4\Delta^2 \alpha}\ mod\ K, \qquad (5)$$

where $\mu_n = \Delta \times \lambda_{0,n}^S \in \mathbb{Z}$, $\lambda_{0,n}^S = \prod_{n' \in S \setminus \{n\}} \frac{-n'}{n-n'}$, $L(u) = \frac{u-1}{K}$.

4) Verification

Each user $P_n (1 \leq n \leq N)$ receives the one-way function values of gradients $\{h(sum(g_1)), h(sum(g_2)), \ldots, h(sum(g_N))\}$ from CS, then he calculates $h(sum(g'_*)) = \prod_{n=1}^N h(sum(g_n)$, and $h(sum(g_*)) = a^{sum(g_*)}\ mod\ b$ based on the aggregation results g_* returned by CS. If $h(sum(g'_*)) = h(sum(g_*))$, the next round of training will begin. Otherwise, CS is required to reaggregate the results.

Algorithm 3 provides a detailed description of the VPPFL process.

.

Next, we provide a proof of correctness for our scheme.

Theorem 1: If CS honestly performs the aggregation operations in the VPPFL, the aggregation results will pass verification.

Proof: The encrypted gradients uploaded by the users to CS are $\{Enc_{pk}(g_1), Enc_{pk}(g_2), \ldots, Enc_{pk}(g_N)\}$. If CS honestly performs the aggregation operation, it will get the encrypted aggregation results as $Enc_{pk}(g_*) = \prod_{n=1}^N Enc_{pk}(g_n)$. Subsequently, CS randomly sends decryption requests to t users, and the numbers of the selected participants form a set S to perform the decryption operation. After receiving the decryption request from CS, each user in S sends the decryption share $s_n = c^{2\Delta sk_n}\ mod\ K^2\ (n \in S)$ to CS.

We have $\prod_{n \in S} s_n^{2\mu_n} = c^{4\Delta \sum_{n \in S} f(n)\mu_n}$

$= c^{4\Delta \sum_{n \in S} \Delta f(n) \lambda_{0,n}^S}$

$= c^{4\Delta^2 m\beta} = (G^{g_*} x^K\ mod\ K^2)^{4\Delta^2 m\beta}$

$= G^{4\Delta^2 m\beta g_*}\ mod\ K^2 (\because \forall x,\ x^{2Km} = 1\ mod\ K^2)$

$= 1 + 4\Delta^2 m\beta g_* K\ mod\ K^2 (\because G = 1 + K)$.

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3472467

Round 0 (Initialization)
TA:
Generate a set of private keys $\{sk_1, sk_2, \ldots, sk_N\}$ and a public key $pk = (K, G, \alpha, \Delta)$ based on Algorithm 2. Broadcast the public key to all participants;
Send the private key $sk_n = f(n)(1 \leq n \leq N)$ to the corresponding user P_n via a secure channel;
Select a large prime number b and a generator a of order k to create a one-way function. Send the function to all users and then go offline.
Round 1(Encryption)
Users:
Each user P_n selects a mini-batch subset $D_n^j \subseteq D_n$ and calculates $g_n \leftarrow \sum_{(x_i,y_i)\in D_n^j} \nabla L_f((x_i, y_i), \omega^j)$;
Calculate the encryption of gradient $Enc_{pk}(g_n)$ and the one-way function of the gradient $h(sum(g_n))$ and then upload them to CS.
CS:
Receive $Enc_{pk}(g_n)$ and $h(sum(g_n))$ from user P_n;
Calculate the aggregated encrypted gradient $c \leftarrow \prod_{n=1}^{N} Enc_{pk}(g_n)$;
Broadcast the received $h(sum(g_n))$ to all users.
Round 2(Decryption)
CS randomly selects t users and sends the decryption requests to them;
After receiving the decryption request, user $P_n(1 \leq n \leq t)$ calculates decryption share $s_n = c^{2\Delta f(n)} \ mod \ K^2$ and sends it to CS;
CS:
Receive the decryption shares $s_n(1 \leq n \leq t)$ from the users ;
Calculate $\lambda_{0,n}^S \leftarrow \prod_{n' \in S\setminus\{n\}} \frac{-n'}{n-n'}$;
Calculate $\mu_n \leftarrow \Delta \times \lambda_{0,n}^S$;
Decrypt the aggregation gradient $g_* \leftarrow L\left(\prod_{n\in S} s_n^{2\mu_n} \mod N^2\right) \times \frac{1}{4\Delta^2\alpha} \ mod \ K$;
Send $g_* = [g_*^1, g_*^2, \ldots, g_*^m]$ to the users.
Round 3(Verification)
Users:
Each user receives g_* from CS;
Calculate $h\left(sum(g_*)\right) \leftarrow a^{sum(g_*)} \ mod \ b$;
Calculate $h\left(sum(g_*^{'})\right) \leftarrow \prod_{n=1}^{N} h\left(sum(g_n)\right)$;
if $h\left(sum(g_*^{'})\right) = h\left(sum(g_*)\right)$
 Users perform parameter update $\omega^{j+1} \leftarrow \omega^j - \theta \cdot \left(\sum_{n\in N} g_*\right) / \left(\sum_{n\in N} |D_n^j|\right)$;
else
 Users request CS to recompute the aggregation results.

Algorithm 3: VPPFL

Therefore, $L\left(\prod_{n\in S} s_n^{2\mu_n} \ mod \ K^2\right) = 4\Delta^2 m\beta g_* = g_* \times 4\Delta^2\alpha \ mod \ K$.

Given that Δ and α are components of the public key, CS is thus able to obtain the aggregation results $g_* = L\left(\prod_{n\in S} s_n^{2\mu_n} \ mod \ K^2\right) \times \frac{1}{4\Delta^2\alpha} \ mod \ K$.

Participants will receive the aggregation gradient g_* returned by CS and the broadcasted one-way function values of gradients $\{h\left(sum(g_1)\right), h\left(sum(g_2)\right), \ldots, h\left(sum(g_N)\right)\}$, then they will obtain $sum(g_*)$ by summing up all elements within g_*. Since threshold Paillier encryption satisfies homomorphic addition, i.e. $Enc_{pk}(g_*) = \prod_{n=1}^{N} Enc_{pk}(g_n)$, the following equation holds: $g_* = g_1 + g_2 + \cdots + g_N$.

Based on the homomorphic property of the one-way function, if the following equation holds, then the aggregation gradients will pass verification: $\prod_{i=1}^{N} h\left(sum(g_i)\right) = \prod_{i=1}^{N} a^{sum(g_i)} \ mod \ b = a^{\sum_{i=1}^{N}(sum(g_i))} \ mod \ b = a^{sum(g_*)} \ mod \ b = h\left(sum(g_*)\right)$.

Therefore, the aggregation results will pass verification.

V. SECURITY ANALYSIS AND PERFORMANCE EVALUATION

A. SECURITY ANALYSIS

In this section, we conduct theoretical analysis and security proof of the VPPFL, including data privacy and the verifiability of the aggregation gradients.

We first introduce some notations. Consider a server CS interacting with a set of N users, and let the security parameter be L. We use U_i to denote the set of users that successfully uploaded their local gradients in round $i-1$, such that $U_4 \subseteq U_3 \subseteq U_2 \subseteq U_1$. Users from these sets can dropout at any time during the process.

Given a subset $W \subseteq U$, the collective perspective

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3472467

of users in W can be represented as a random variable $REAL_W^L(g, U_1, U_2, U_3, U_4)$, where L is the security parameter. To prove that our scheme is secure, we first introduce a definition; only schemes that meet the conditions of this definition are considered secure.

Definition 1: If any adversary $\mathbb{A}$ has a negligible advantage over the following game in polynomial time for security parameter L, then the scheme is indistinguishable under the choice of plaintext attack, and the scheme is said to be ICD-CPA secure.

Initialization stage: Enter the security parameter L, challenger $\mathbb{C}$ generates the system parameter para and private key sk, and sends the system parameter para to opponent $\mathbb{A}$.

Challenge Phase: The adversary chooses two messages m_0 and m_1 and sends them to the challenger $\mathbb{C}$. Here, the two messages are of equal length, i.e. $|m_0| = |m_1|$. Upon receiving these two messages, the challenger $\mathbb{C}$ randomly selects $b \in \{0, 1\}$, computes $C^* = ENC(param, m_b)$, and then sends C^* to the adversary $\mathbb{A}$.

Output: The adversary $\mathbb{A}$ outputs a guess b' for b. If $b' = b$, the adversary $\mathbb{A}$ wins the challenge; otherwise, the adversary $\mathbb{A}$ loses the challenge.

The advantage of the adversary $\mathbb{A}$ in winning the above game is defined as $Adv_{\epsilon,A}(L) = \left|\Pr(b = b') - \frac{1}{2}\right|$.

1) Data privacy

Theorem 2: VPPFL can resist collusion attacks between the CS and fewer than t users. That is, for all $L, g, W \subseteq U$, and $U_4 \subseteq U_3 \subseteq U_2 \subseteq U_1$, there exists a PPT simulator SIM whose output is indistinguishable from the output of $REAL_W^L$.

$$REAL_W^L(g, U_1, U_2, U_3, U_4) \equiv SIM_W^L(g, U_1, U_2, U_3, U_4).$$

Proof: We assume the set of participants colluding with CS is $P_{collude} = \{P_1, P_2, \ldots, P_{Num}\}$, where $Num < t$. If the adversary $\mathbb{A}$ want to obtain the plaintext gradient g_n from the encrypted gradient $Enc_{pk}(g_n)$, he needs to get the decryption key Sk. However, none of the parties can obtain this decryption key individually, at least t users are required through Shamir's secret sharing. In fact, Shamir's secret sharing scheme has been shown to be semantically secure under the DDH hardness assumption [27]. Therefore, in the case of fewer than t users colluding, the output of the simulator SIM is computationally indistinguishable from the output of $REAL$.

Theorem 3: In VPPFL, no party can obtain the private information of other users.

Proof: The data that CS can obtain the encrypted aggregation gradients and the one-way function values of gradients $\{h(sum(g_1)), h(sum(g_2)), \ldots, h(sum(g_N))\}$. The data that a user P_i can obtain include all users' one-way function value of gradient and their own split secret key sk_i.

From Theorem 2, we know that CS colluding with fewer than t users, cannot extract other users' private information from the encrypted gradients. Therefore, a single user also cannot derive any useful information from the encrypted gradients. Both CS and all users can obtain the one-way function values of gradients $\{h(sum(g_1)), h(sum(g_2)), \ldots, h(sum(g_N))\}$. Furthermore, users can also access the one-way function $h(M) = a^M \ mod \ b$. Due to the irreversibility of the one-way function, the plaintext $sum(g_i)$ is secure. Even in the extreme case where $sum(g_i)$ is obtained, users wouldn't get any private information about g_i, since $sum(g_i)$ is only the aggregation value of the gradient.

Theorem 4: If the DDH difficulty question is assumed, VPPFL is IND-CPA safe. That is, the proposed scheme can meet the security definition of data privacy under the selection of plaintext attacks.

Proof: If there exists an external adversary $\mathbb{A}$ who attempts to eavesdrop on the encrypted gradients $Enc_{pk}(g_n)$ uploaded by users to the server. Since $Enc_{pk}(g_n)$ is a valid ciphertext in the Paillier cryptosystem, and this system has been proven to be semantically secure under the DDH hardness assumption [26]. The external adversary $\mathbb{A}$ cannot obtain the corresponding plaintext information from the ciphertexts generated by the users.

At the same time, as a result of theorem 2, our protocol is secure even if a small number of users collude with the server. It can be obtained from theorem 3, even if any party involved in the training calculates based on the input data they obtain, intermediate results, etc. Therefore, the probability that external adversary $\mathbb{A}$ will get the plaintext in polynomial time is negligible.

Through the above proof, neither external adversary nor internal adversary can obtain the private information of a single user. Therefore, our protocol is IND-CPA secure.

2) Verifiability of the aggregation gradient

Theorem 5: If CS returns an incorrect aggregation gradient g_*, it will fail the verification process.

Proof: From Theorem 1, if CS tries to reduce the amount of computation for aggregation, the aggregation gradient ciphertext will become $\prod_{i=1}^{N} h(sum(g_i)) = \prod_{i=1}^{N} a^{sum(g_i)} \ mod \ b = a^{\sum_{i=1}^{N}(sum(g_i))} \ mod \ b = a^{sum(g_*)} \ mod \ b = h(sum(g_*))$. If CS attempts to reduce the computation cost by providing an aggregated gradient ciphertext $Enc_{pk_{less}}(g_*) = \prod_{n=1}^{less} Enc_{pk}(g_n)$, where $less < N$. Each user already knows the one-way function values of gradients $\{h(sum(g_1)), h(sum(g_2)), \ldots, h(sum(g_N))\}$, so they can compute $h\left(sum(g_{*(less)})\right) < \prod_{i=1}^{N} h\left(sum(g_i)\right)$. Therefore, any reduction in the computation cost by CS, indicating laziness or tampering, will certainly be detected.

B. PERFORMANCE EVALUATION

To highlight the advantages of VPPFL, we conducted a detailed comparison with some existing schemes, as shown in Table 2. Moreover, we also implemented the PPVerifier scheme to facilitate a more detailed comparison with our scheme.

The following introduces the computation cost of the scheme. Assuming there is N users and one server, and each user with a model parameter vector of size d. CS randomly selects t users to cooperate in decryption. As the cost depends

TABLE 2. Scheme comparison

	PPML [28]	SafetyNets [11]	PPVerifier [24]	VPPFL
Data Security	✓	×	✓	✓
Verifiability	×	✓	✓	✓
Support users dropout	✓	×	×	✓

on N, d, and t. The cost of VPPFL on the user and CS is shown in Table 3.

TABLE 3. Computation and communication cost

	Computation cost	Communication cost
CS	$O(N+t)$	$O(N+t)$
User	$O(d)$	$O(1)$

The computation cost for CS mainly comes from three main aspects. The first is the computation cost of aggregating gradient ciphertexts uploaded by N users, which is $0(N)$; Second, CS randomly selects t users to cooperate with the decryption of the global aggregation gradients, and the computation cost is $O(t)$; Third, CS calculats the one-way function values of the global aggregation gradient, where the computation cost is $O(1)$. Therefore, the total computation cost for CS is $O(N+t)$. The communication cost of CS mainly comes from sending the decrypted global model to N users, accepting the encryption gradient and the one-way function values sent by N users, and selecting t users to send decryption requests. Therefore, the total communication cost for CS is $O(N+t)$.

The computation cost for the user mainly comes from the computation cost on the user mainly comes two aspects. First, the computation cost of the local gradient, which is $O(d)$. Second, the computation cost for verification, which is $O(1)$. The communication cost of the user mainly comes from sending the one-way function value of the gradient to CS and encrypting the local gradient, which is $O(1)$. Therefore, the communication cost for the user is $O(1)$.

VI. EXPERIMENT EVALUATION

In this section, we conduct all-round experiments on VPPFL to evaluate its performance.

A. EXPERIMENTAL ENVIRONMENT

We implemented VPPFL using MATLAB2016a. The algorithm was implemented on the MNIST dataset (http://yann.lecun.com/exdb/mnist/). The dataset includes 70,000 grayscale images, each 28x28 pixels, depicting handwritten digits, segmented into 60,000 images for training and 10,000 for testing. The experiment utilized a neural network as the training model, consisting of an input layer, an output layer and two hidden layers. We set the learning rate is 0.2. The experiments were conducted on a computer with Intel Core i5-1035G1, 1.0GHz CPU, and 8GB of memory.

B. CLASSIFICATION ACCURACY

We implemented the PPVerifier protocol [24] as well as the unencrypted original algorithm FedSGD [8] to analyze the accuracy of our scheme in neural network training. In practical use cases, as gradient vectors typically exist in floating-point format, our approach requires preprocessing them into integers before encryption. This is why our VPPFL and PPVerifier slightly lag behind FedSGD in terms of accuracy. We kept all conditions the same.

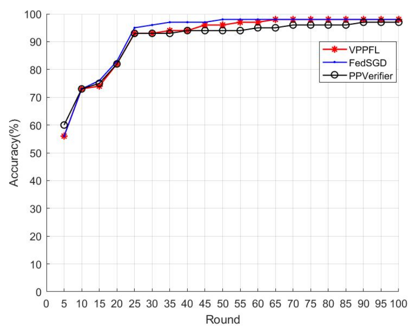

FIGURE 5. Compares the accuracy of VPPFL, PPVerifier and FedSGD.

As shown in Figure 5, with a total gradient count set to 100,000 and 100 training rounds respectively. After 100 training rounds, both VPPFL and FedSGD achieved nearly the same 98% accuracy. This indicates that when using VPPFL to protect the gradients, it can still maintain the model's accuracy. This is because only a small portion of gradient information is lost.

C. COMPUTATION COST

In this section, we will delve into the total computation cost, the impact of the number of gradients on computation cost, the computation cost on CS and users, as well as the analysis of computation cost when users dropout during the training process.

1) Total computation cost

As shown in Figure 6, we set the total number of gradients involved in training to 100,000. Although we incurred some additional time cost compared to PPVerifier, this cost is acceptable, and our scheme supports involvement and dropout of users during the training process, which is more aligned with practical applications. PPVerifier does not support this feature. Therefore, the cost we incurred is worthwhile.

2) Computation cost between CS and users

To facilitate observation, we set the number of users participating in training to 10. As shown in Figure 7, with an increase

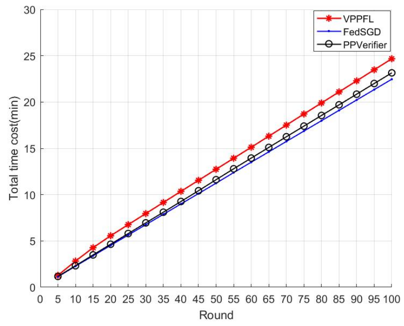

FIGURE 6. Total computation cost.

in the number of gradients, the computation cost on the client side increases linearly. When each user has 10,000 gradients, the users' computation cost will surpass the CS computation cost. This places higher demands on the computing power of users participating in training. Therefore, each user needs to appropriately manage the amount of data they participate in training each round, thus to avoid training too much data at once to prevent excessive burden on themselves. The server's computation cost does not significantly increase because the server does not participate in the users' training process, which ensures user privacy and security.

3) The computation cost in different stages.

We analyze in detail the computation cost of different stages in one round of training. We set the number of users participating in training to 100. As shown in Table 4, it is evident that the users' computation cost are primarily composed of encryption and verification stages, with this portion of the expenditure occupying a very low proportion of the total computation cost. The part that incurs the highest cost is the decryption process, which is handled by CS, making it more inclusive for users with weaker computing capabilities.

4) Computation cost on CS when users dropout

As shown in Figure 8, we set the number of users participating in training to 100. As the number of dropout users increases, the computation cost on CS does not increase. The reason for this is that the cloud server's computation cost are mainly concentrated in the decryption process, which involves sending decryption requests to t users. Even if some users dropout, CS only needs to send decryption requests to the remaining users who are still online, thus not adding extra computation for CS.

VII. CONCLUSION

In this paper, we proposed VPPFL, a privacy-preserving federated learning scheme for the semi-malicious server. VPPFL supports users dropout and provides verifiability for each user during the training process while preserving user privacy. Furthermore, we proved the security of the scheme and validated the practical performance of our scheme through simulated experiments on real data theoretically.The scheme proposed in this paper solves some problems in federated learning to a certain extent, but there is still room for improvement. For example, this scheme relies on a trusted third party, and in the actual deployment of federated learning, finding such a trusted entity is a huge challenge. How to achieve verifiable privacy-preserving federated learning without the participation of a trusted third party is an important direction of follow-up research.

Data availability

The datasets are available online. The URL is as follows: MNIST database: http://yann.
lecun.com/exdb/mnist.

REFERENCES

[1] K. Ahiska, M. K. Ozgoren, and M. K. Leblebicioğlu, "Autopilot design for vehicle cornering through icy roads," *IEEE Transactions on Vehicular Technology*, vol. 67, pp. 1867–1880, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:3964921

[2] W. Bo, L. Hongtao, and W. Jie, "Privacy protection federal learning architecture for medical data," *Journal of Xi'an University of Electronic Science and Technology*, vol. 50, pp. 166–177, 2023.

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

[4] Z. Qu, Y. Tang, M. Ghulam, and P. Tiwari, "Privacy protection in intelligent vehicle networking: A novel federated learning algorithm based on information fusion," *Inf. Fusion*, vol. 98, p. 101824, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258434239

[5] R. A. S. A. A. Z. K. V. M. Shariatnia S, Ziaratban M, "Modeling the diagnosis of coronary artery disease by discriminant analysis and logistic regression: a cross-sectional study," *BMC Med Inform Decis Mak*, vol. 22, no. 1, p. 85, 2022.

[6] G. Xu, H. Li, Y. Dai, K. Yang, and X. Lin, "Enabling efficient and geometric range query with access control over encrypted spatial data," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 870–885, 2019.

[7] W. Canxia and L. Wenjuan, "Research on patient privacy protection and countermeasures in the application of health and medical big data," *Network Security Technology and Applications*, 2022.

[8] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *ArXiv*, vol. abs/1602.05629, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:16861557

[9] L. Zhu, Z. Liu, and S. Han, *Deep leakage from gradients*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[10] Z. Wang, M. Song, and Z. Zhang, "Beyond inferring class representatives: User-level privacy leakage from federated learning," p. 2512–2520, 2019.

[11] Z. Ghodsi, T. Gu, and S. Garg, "Safetynets: Verifiable execution of deep neural networks on an untrusted cloud," 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:25998539

[12] A. Fu, Z. Chen, Y. Mu, W. Susilo, Y. Sun, and J. Wu, "Cloud-based outsourcing for enabling privacy-preserving large-scale non-negative matrix factorization," *IEEE Transactions on Services Computing*, vol. 15, no. 1, pp. 266–278, 2022.

[13] X. Hua and T. Youliang, "Weighted social network privacy protection under differential privacy," *Journal of Xidian University*, no. 001, p. 049, 2022.

[14] A. Falcetta and M. Roveri, "Privacy-preserving deep learning with homomorphic encryption: An introduction," *IEEE Computational Intelligence Magazine*, vol. 17, no. 3, pp. 14–25, 2022.

[15] C. Dwork, "Differential privacy," in *International Colloquium on Automata, Languages and Programming*, 2006. [Online]. Available: https://api.semanticscholar.org/CorpusID:2565493

[16] C. Dwork and J. Lei, "Differential privacy and robust statistics," ser. STOC '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 371–380. [Online]. Available: https://doi.org/10.1145/1536414.1536466

[17] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *ArXiv*, vol. abs/1712.07557, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:3630366

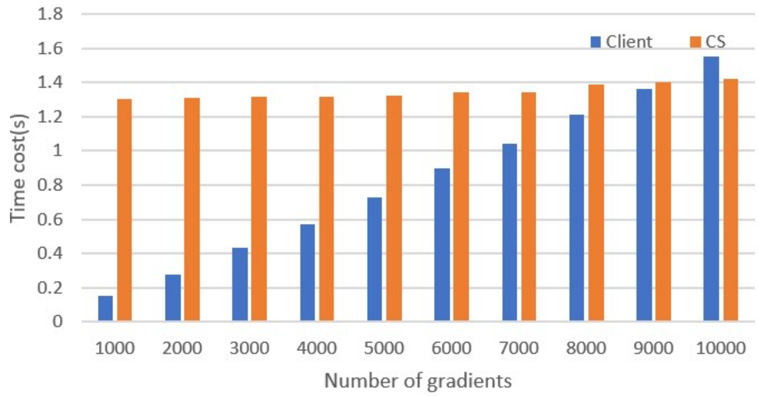

FIGURE 7. **Comparison of computation cost on client and CS.**

TABLE 4. **Computation cost of different parts of the training process**

Number of gradients for each user	Key Generation (TA)	Encryption (Users)	Decryption (CS)	Verification (Users + CS)	Total Cost
2000	1.5ms	2.7ms	1147ms	0.5ms+2ms	1153.7ms
4000	1.5ms	2.7ms	1314ms	0.5ms+1.9ms	1320.6ms
6000	1.6ms	3.3ms	1376ms	0.6ms+2.7ms	1384.2ms
8000	2.5ms	3.3ms	1430ms	0.7ms+2.2ms	1438.7ms
10000	3.2ms	3.4ms	1662ms	0.7ms+4.1ms	1673.4ms

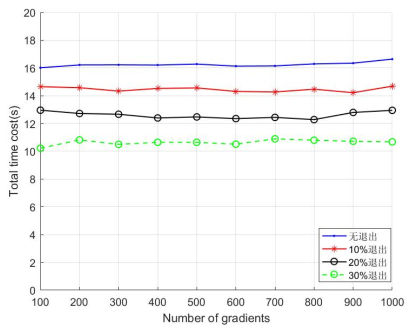

FIGURE 8. **Comparison of computation cost on CS when users dropout.**

[18] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, and H. Vincent Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.
[19] R. L. Rivest and M. L. Dertouzos, "On data banks and privacy homomorphisms," 1978. [Online]. Available: https://api.semanticscholar.org/CorpusID:6905087
[20] B. Wang, H. Li, Y. Guo, and J. Wang, "Ppflhe: A privacy-preserving federated learning scheme with homomorphic encryption for healthcare data," *Appl. Soft Comput.*, vol. 146, no. C, oct 2023. [Online]. Available: https://doi.org/10.1016/j.asoc.2023.110677
[21] S. S. Jing Ma, Si-Ahmed Naas and X. Lyu, "Privacy-preserving federated learning based on multi-key homomorphic encryption," vol. 37, no. 9, pp. 5880–5901, 2022.
[22] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifynet: Secure and verifiable federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 911–926, 2020.
[23] X. Guo, Z. Liu, J. Li, J. Gao, and T. Baker, "Verifl: Communication-efficient and fast verifiable aggregation for federated learning," *IEEE Transactions on Information Forensics and Security*, vol. PP, no. 99, pp. 1–1, 2020.
[24] L. Lin and X. Zhang, "Ppverifier: A privacy-preserving and verifiable federated learning method in cloud-edge collaborative computing environment," *IEEE Internet of Things Journal*, vol. 10, no. 10, pp. 8878–8892, 2023.
[25] Y. Ren, Y. Li, G. Feng, and X. Zhang, "Privacy-enhanced and verification-traceable aggregation for federated learning," *IEEE Internet of Things Journal*, vol. 9, no. 24, pp. 24 933–24 948, 2022.
[26] T. Nishide and K. Sakurai, "Distributed paillier cryptosystem without trusted dealer," in *International Conference on Information Security Applications*, 2010.
[27] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, p. 612–613, nov 1979. [Online]. Available: https://doi.org/10.1145/359168.359176
[28] K. Bonawitz, Ivanov, Vladimir, and Kreuter, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17, New York, NY, USA, 2017, p. 1175–1191. [Online]. Available: https://doi.org/10.1145/3133956.3133982

FIRST A. HUIYONG WANG received the Ph.D. degree in software theory and applications from the Chinese Academy of Sciences, China, in 2017. He is currently a Lecturer with the School of Mathematics and Computing Science, Guilin University of Electronic Technology, China. His research interests include privacy-preserving computation, information security, cyber security, multi-party computation, and homomorphic encryption.

FIRST B. TENGFEI YANG is currently a master's student in the field of network information security and privacy protection at Guilin University of Electronic Technology. Her current research interests include cybersecurity, privacy protection, and artificial intelligence. yangtengfei2022@163.com.

FIRST C. YONG DING received the PH.D.degree in cryptography from the School of Communication Engineering, XiDian University, China,in 2005. From 2008 to 2009, he was a Research Fellow of computer science with the City University of Hong Kong. He is currently a Professor with the School of Computer Science and Information Security, Guilin University of Electronic Technology,China. His research interests include cryptography ang information securit.

FIRST D. SHIJIE TANG is currently a lecturer and doctoral student in the field of industrial control security at Guilin University of Electronic Science and Technology, China. Her current main research interests include industrial control security, network information security and privacy protection.tangsj@guet.edu.cn.

FIRST E. YUJUE WANG received the joint Ph.D. degree from Wuhan University, Wuhan, China, and the City University of Hong Kong, Hong Kong, in 2015. He was a Research Fellow with the School of Information Systems, Singapore Management University. He is currently with the School of Computer Science and Information Security, Guilin University of Electronic Technology, China. His research interests include applied cryptography, database security, and cloud computing security.