Cluster Computing
https://doi.org/10.1007/s10586-024-04558-5

PPFLV: privacy-preserving federated learning with verifiability

Qun Zhou[1] · Wenting Shen[1]

Received: 21 January 2024 / Revised: 10 March 2024 / Accepted: 7 May 2024

Abstract
Federated learning, as an emerging framework for distributed machine learning, has received widespread attention. In federated learning, the cloud server and the users cooperatively train a model by sharing gradients rather than local private data. However, the users' private data may still be exposed by the shared gradients. Furthermore, the cloud server may perform incorrect aggregation operations on the gradients sent by users and send a forged or previous aggregated gradient to the users. In this paper, we propose PPFLV, a privacy-preserving federated learning scheme with verifiability. Specifically, to protect the users' privacy, we design an efficient double gradient blinding and encryption method to blind and encrypt the users' local gradients. Furthermore, we propose a novel double gradient verification method that can achieve secure verification while resisting replay attacks in the verification phase. With the proposed verification method, the users only require to perform lightweight operations to verify the correctness of the aggregated encrypted gradients and recover the aggregated gradient from the aggregated encrypted gradients. The experimental results show that PPFLV achieves comparable classification accuracy to the basic federated learning scheme while providing privacy protection and verifiability. Furthermore, PPFLV exhibits lower computation and communication overhead compared to related schemes.

Keywords Privacy-preserving · Verifiable · Federated learning · Cloud computing

1 Introduction

With the exponential growth of data, deep learning has been rapidly developed in many fields, such as natural language processing [1], autonomous driving [2], and medical diagnosis [3]. In deep learning, a large amount of data is collected, which is used to train a powerful and accurate model. However, these collected data might include some sensitive information [4–6]. For example, in e-healthcare systems, the users' health data include sensitive information. If these health data are directly uploaded to the cloud server for training and prediction, the users' sensitive data and health status will be ineluctably leaked to the cloud server.

Federated learning is an effective technology to solve the privacy problem of user data [7–9]. In federated learning, when training a neural network model, the users only require to upload the local gradients obtained through local training to the cloud server, instead of uploading and sharing their original data. The cloud server is able to obtain a global model by aggregating the gradients uploaded by the users. However, a lot of researches have shown that even if the users' training data is not required to be uploaded to the cloud server, the cloud server is still able to infer the private training data based on the shared local gradients [10–12]. To protect the users' privacy, plenty of privacy-preserving federated learning schemes have been proposed [13–17]. Bonawitz et al. [18] designed a privacy-preserving federated learning scheme by using secure multi-party computation technique. Phong et al. [19] constructed a federated learning scheme supporting privacy protection, in which the gradients are encrypted by using two homomorphic encryption technologies. Jia et al. [20] adopted differential privacy technology to achieve users' privacy protection in federated learning. In the above schemes, the cloud server can carry out the gradient aggregation without exposing the privacy of the users' local gradients.

Another important issue in federated learning is the verification of the correctness of the aggregated result

✉ Wenting Shen
shenwentingmath@163.com

[1] College of Computer Science and Technology, Qingdao University, Qingdao 266071, China

Published online: 16 June 2024

[21, 22]. In practice, to reduce computation overhead, a "lazy" cloud server may not aggregate all gradients sent by the users [23]. Even worse, a malicious cloud server may generate and return an incorrect aggregated gradient to the users for the sake of impacting the model updates [24]. To solve the above problems, Xu et al. [4] proposed a verifiable and privacy-preserving federated learning scheme. This scheme is able to guarantee the privacy of user data by masking the local gradients and verifying the correctness of the aggregated result by using the homomorphic hash function. However, if the malicious users collude with the cloud server, they can launch the brute force attack to recover the users' local gradients. Hahn et al. [11] proposed a federated learning scheme supporting verifiability and privacy preserving, which can resist brute force attacks. Nevertheless, this scheme is vulnerable to replay attacks in the verification phase, in which the cloud server may use the previous aggregated gradient to trick the users into passing the verification.

In this paper, we propose a privacy-preserving federated learning scheme with verifiability (PPFLV), which is able to resist replay attacks.

Contribution: Our contributions can be summarized as follows:

1. We design a novel double gradient verification method, which can resist replay attacks. With the two aggregated encrypted gradients returned by the cloud server, the users only need to perform the lightweight calculation to check whether the cloud server correctly aggregates the users' gradients. The users are able to recover the aggregated gradient from the aggregated encrypted gradients. Furthermore, by employing this verification method, the cloud server cannot successfully pass the verification with the previous or wrong aggregated gradients.
2. We design an efficient double gradient blinding and encryption method to blind and encrypt the users' local gradients, which is compatible with our designed verification method. We use the secret sharing technique to share the users' private keys and noise. The correctness of these secret shares can be verified by employing the verification technology based on discrete logarithms. In addition, when the users are offline or the network is delayed, the online users can recover the offline users' private keys and the online users' noises, while the correctness of the aggregated gradient can still be guaranteed.
3. We provide the security analysis and performance evaluation of the proposed PPFLV. The security analysis demonstrates that the proposed PPFLV satisfies correctness, gradient privacy, immunity from replay attack and unforgeability. The experiment results show that the proposed PPFLV has high accuracy and is efficient in terms of gradient encryption, gradient aggregation and verification.

Organization: The rest of this paper is organized as follows. Section 2 shows the related works. In Sect. 3, we introduce the system model and the threat model. In Sect. 4, We present the preliminaries for PPFLV. In Sect. 5, we present the construction of the proposed PPFLV. We give the security analysis and performance evaluation of the proposed PPFLV in Sects. 6 and 7, respectively. We conclude the paper in Sect. 8.

2 Related work

In recent years, federated learning has been used in a wide range of industries. Privacy preserving and verifiability of aggregated results in federated learning have received considerable attention. In federated learning, the cloud server can recover the users' private data based on the local gradients uploaded by the users. To protect user data privacy, plenty of privacy-preserving federated learning schemes [25–29] have been proposed. Privacy-preserving federated learning focuses on protecting users' data privacy and preventing users from exposing sensitive data. Phong et al. [19] used Paillier homomorphic encryption technology and LWE homomorphic encryption technology to encrypt gradients for achieving privacy protection. Tang et al. [30] proposed a robust privacy-preserving federated learning scheme that protects the privacy of local gradients and global models. Based on ElGamal multiplicative homomorphic encryption technology, Fang et al. [31] put forward a privacy-preserving federated learning scheme, in which the gradients of the participants can be protected. Using homomorphic encryption technology, the encrypted gradients still can be aggregated. Jia et al. [20] and Wei et al. [32] utilized differential privacy to protect users' data privacy in federated learning. In these two schemes [20, 32], the noise is added to the gradients to achieve differential privacy. By combining secure multi-party computing with differential privacy technology, Mugunthan et al. [33] designed a federated learning scheme supporting privacy-preserving. Zhou et al. [34] introduced a trusted blinding server to blind the gradient ciphertexts and constructed a privacy-preserving federated learning scheme. The problem of collusion between the cloud server and the users can be solved in this scheme. Bonawitz et al. [18] put forward a federated learning scheme with privacy protection by using secret sharing. In this scheme, secure multi-party computation technology is utilized to calculate the sums of model parameters. The above schemes can realize the users' privacy protection and gradient

aggregation, but they cannot guarantee the correctness of the aggregated results.

To verify the correctness of aggregated gradient generated by the cloud server, many verifiable federated learning schemes [16, 35–38] have been proposed. Xu et al. [4] utilized homomorphic hash function to check the correctness of aggregated gradients. Peng et al. [39] constructed a verifiable federated learning scheme based on blockchain, which is able to optimize the training process in federated learning by using the reward mechanism of blockchain. In this scheme, the verifiable proofs are recorded in the blockchain. Fu et al. [21] adopted Lagrange interpolation to achieve the verification of aggregated gradients and used Chinese remainder theorem to reduce communication overhead. Zhang et al. [24] utilized bilinear aggregate signature to construct a federated learning scheme supporting secure verification. In this scheme, the participants can check whether the aggregation server correctly aggregates the gradients uploaded by the participants. Guo et al. [40] also considered the problem of aggregated gradient verification, and used homomorphic hash to guarantee the correctness of the aggregated gradient. Xu et al. [41] put forward a non-interactive verifiable federated learning scheme, which introduces two servers to aggregate the gradients. In this scheme, the users are able to verify the correctness of aggregated gradient by cross-authentication. Hahn et al. [11] proposed a double aggregation approach to complete the verification of aggregated gradient. Only lightweight primitives are used in this scheme, which improves the computational efficiency of verification. However, this scheme cannot resist replay attacks. The cloud server can pass the verification of the participants by using the previous aggregated gradient. Thus, it is meaningful to explore a privacy-preserving and verifiable federated learning scheme with efficient encryption and verification while resisting replay attacks.

3 System model and threat model

3.1 System model

The system model of PPFLV is shown in Fig. 1. It consists of three kinds of entities: cloud server, user group and KGC (Key Generation Center).

- Cloud server: The cloud server is in charge of aggregating the encrypted gradients sent by users. The cloud server generates and sends the aggregated encrypted gradients to each user in the group. The aggregated encrypted gradients are used to verify whether the cloud server performs the aggregation operation correctly.
- User group: A user group consists of multiple users. The users train the model locally, encrypt the local gradients, and send the encrypted gradients to the cloud server. After receiving the aggregated encrypted gradients sent by the cloud server, the users can check the correctness of aggregated encrypted gradients. If the aggregated encrypted gradients are correct, the users can calculate the aggregated gradient and update the model.
- KGC: KGC is responsible for initializing the model parameters of the neural network, and generating system parameters and public-private key pairs for the users.

3.2 Threat model

In our threat model, KGC is a trustworthy entity and does not collude with the users and the cloud server. The users are regarded as an honest-but-curious entity. They honestly perform the specified procedures based on the agreed agreement, but are curious about other users' data privacy. The cloud server is considered to be a malicious entity. The cloud server may attempt to infer the data privacy of all users. The cloud server may also perform incomplete aggregation operation, and falsify the wrong aggregated encrypted gradients or use the previous aggregated encrypted gradient to deceive the users.

4 Preliminaries

In this section, we introduce the preliminaries such as Shamir's secret sharing, authenticated encryption and key agreement.

4.1 Shamir's secret sharing

In Shamir's t-out-of-n secret sharing technique [42], a secret value τ is split into n shares, where n indicates the total number of shares and t is the threshold. The secret value τ can be reconstructed by more than t shares. The Shamir's secret sharing protocol consists of the following algorithms:

1. $ShamirS.share(\tau, t, U) \rightarrow \tau_i$: The secret share algorithm takes as input the secret value τ, a user group U with logical identities $[0, 1, \ldots, n-1]$ and the threshold t $(t<n)$, and outputs the secret share τ_i for each member U_i $(i \in [0, n-1])$.
2. $ShamirS.recon(\{\tau_i\}_{i\in\mu}, t) \rightarrow \tau$: The secret reconstruction algorithm takes as input a set of secret shares $\{\tau_i | i \in \mu, \mu \subseteq [0, n-1]\}$ and the threshold t $(t<n)$, and outputs a secret value τ.

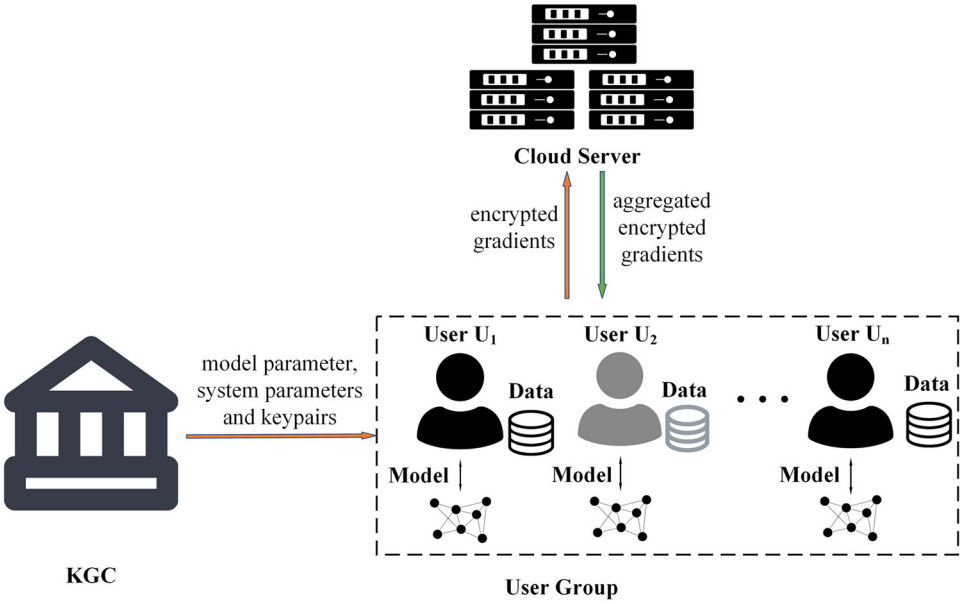


Fig. 1 System model

4.2 Authenticated encryption

Authenticated encryption (AE) is a symmetric encryption algorithm, in which the encryption key and the decryption key are the same. AE is used to ensure the confidentiality and integrity of user data [43]. The algorithms of AE are as follows:

1. $AE.gen(\lambda) \rightarrow \kappa$: Taking the key space λ as input, this algorithm samples a secret key κ randomly and uniformly from the key space λ.
2. $AE.enc(m, \kappa) \rightarrow c$: Taking the secret key κ and the plaintext m as input, this algorithm outputs the ciphertext c.
3. $AE.dec(c, \kappa) \rightarrow m$: Taking the secret key κ and a ciphertext c as input, this algorithm outputs the plaintext m.

4.3 Key agreement

Diffie-Hellman key agreement [44] is used in our PPFLV to generate the shared encryption key of AE for any two users. Let $\mathbb{G}$ be a group with prime order q and g be the generator of the group $\mathbb{G}$. The Diffie-Hellman key agreement consists of the following algorithms:

1. $KA.gen(g, q, \mathbb{G}) \rightarrow (SK_i, PK_i)$: This algorithm takes a generator g, the order q and the group $\mathbb{G}$ as input, and outputs the secret key SK_i and the public key $PK_i = g^{SK_i}$.
2. $KA.agree(SK_i, PK_j) \rightarrow \alpha_{i,j}$: This algorithm takes the secret key SK_i of the user U_i and the public key PK_j of the user U_j as input, and output the shared encryption key $\alpha_{i,j}$. In our PPFLV, the shared encryption key $\alpha_{i,j}$ is calculated as follows: $\alpha_{i,j} = H(PK_j^{SK_i})$, where $H : \mathbb{G} \rightarrow \mathbb{Z}_q^*$ is cryptographic hash function. Note that $\alpha_{i,j} = \alpha_{j,i}$.

5 The proposed scheme

5.1 Overview

The main idea of our proposed scheme is to achieve the verifiability of aggregated gradient on the basis of privacy protection. To achieve privacy protection, we design a novel double gradient blinding and encryption method to blind and encrypt the users' local gradients. The user U_i $(i \in [0, n-1])$ blinds the local gradient ω_i as $a_i = \omega_i + \pi_{s_1}(R)$ and $b_i = \omega_i \cdot \pi_{s_1}(R)$ under a pseudo-random function $\pi_{s_1}(\cdot)$, where R is the current iteration number. The user U_i $(i \in [0, n-1])$ uses his private key SK_i and the user U_j's $(j \in [0, n-1], j \equiv i(\text{mod}2), j \neq i)$ public key PK_j to compute a secret value $\alpha_{i,j} = KA.agree(SK_i, PK_j)$. Note that $\alpha_{i,j} = \alpha_{j,i}$. The user U_i $(i \in [0, n-1])$ utilizes $\alpha_{i,j}$ to calculate a random vector $F_{s_2}(\alpha_{i,j}||R)$ under a pseudo-random function $F_{s_2}(\cdot)$, then encrypts the blinded gradients a_i and b_i as follows:

$$\hat{a_i} = a_i + \sum_{j \equiv i(\text{mod}2), j > i} F_{s_2}(\alpha_{i,j}||R) - \sum_{j \equiv i(\text{mod}2), j < i} F_{s_2}(\alpha_{i,j}||R)$$

$$\hat{b_i} = b_i + \sum_{j \equiv i(\text{mod}2), j > i} F_{s_2}(\alpha_{i,j}||R) - \sum_{j \equiv i(\text{mod}2), j < i} F_{s_2}(\alpha_{i,j}||R)$$

In the encrypted gradients $\hat{a_i}$, $\hat{a_j}$ $(j \equiv i(\text{mod}2), j > i)$, the

random vectors $F_{s_2}(\alpha_{i,j}||R)$ and $F_{s_2}(\alpha_{j,i}||R)$ respectively generated by the users U_i and U_j can cancel each other out. In the encrypted gradients $\hat{b_i}$, $\hat{b_j}$ $(j \equiv i(\text{mod}2), j > i)$, the random vectors $F_{s_2}(\alpha_{i,j}||R)$ and $F_{s_2}(\alpha_{j,i}||R)$ respectively generated by the users U_i and U_j can also cancel each other out. If the cloud server is able to successfully receive the encrypted gradients from all users, the cloud sever aggregates the encrypted gradients as follows:

$$\sum_{i\in[0,n-1],i\equiv 0(\text{mod}2)} \hat{a_i} + \sum_{i\in[0,n-1],i\equiv 1(\text{mod}2)} \hat{a_i} = \sum_{i\in[0,n-1]} a_i,$$
$$\sum_{i\in[0,n-1],i\equiv 0(\text{mod}2)} \hat{b_i} + \sum_{i\in[0,n-1],i\equiv 1(\text{mod}2)} \hat{b_i} = \sum_{i\in[0,n-1]} b_i.$$

Based on the aggregated encrypted gradients $\sum_{i\in[0,n-1]} a_i$ and $\sum_{i\in[0,n-1]} b_i$, each user U_i $(i \in [0, n-1])$ is able to obtain the aggregated gradient through the equation:

$$\sum_{i\in[0,n-1]} a_i - |n| \cdot \pi_{s_1}(R) = \sum_{i\in[0,n-1]} b_i \cdot \frac{1}{\pi_{s_1}(R)} = \sum_{i\in[0,n-1]} \omega_i.$$

However, in practice, not all users send the gradients to the cloud server. There are some users who go offline due to external factors [18]. When some users are offline, the online users cannot obtain the correct aggregated gradient since the random vectors added in the encrypted gradients of offline users cannot be cancelled out. To solve the user offline problem, we use the secret sharing technique to share the private keys of all users. We set the identity set of all online users to I_{online} and the identity set of all offline users to $I_{offline} = \{0, 1, ..., n-1\} - I_{online}$. Each user U_i $(i \in [0, n-1])$ utilizes the secret sharing technique to share his private key SK_i to other users in the group. If the user U_j $(j \in I_{offline})$ is offline during the training process, the cloud server can recover the offline users U_j $(j \in I_{offline})$' private keys SK_j $(j \in I_{offline})$ based on the secret shares provided by more than t users. Then the cloud server can calculate the secret values $\alpha_{j,v} = KA.agree(SK_j, PK_v)$ $(j \in I_{offline}, v \in [0, n-1], v \equiv j(\text{mod}2), v \neq j)$ based on the recovered private keys SK_j $(j \in I_{offline})$ of the offline users U_j $(j \in I_{offline})$ and the public keys PK_v $(v \in [0, n-1], v \equiv j(\text{mod}2), v \neq j)$ of the users U_v $(v \in [0, n-1], v \equiv j(\text{mod}2), v \neq j)$. In this way, the cloud server is able to recover the random vectors $F_{s_2}(\alpha_{j,v}||R)$ $(j \in I_{offline}, v \in [0, n-1], v \equiv j(\text{mod}2), v \neq j)$ and removed these random vectors from $\sum_{i\in[0,n-1]} a_i$ and $\sum_{i\in[0,n-1]} b_i$ during the aggregation process.

Although the above method can solve the user offline problem, there is still network latency issue. Due to network latency, there are some users who did not upload the encrypted gradients to the cloud server on time [4]. This incurs that the cloud server identifies these users with network latency as offline users. The cloud server recovers the private keys of these users, calculates the corresponding secret values, and recovers the random vectors. Once these users with network latency successfully upload the encrypted gradients to the cloud server, the cloud server may be able to correctly recover these users' local gradients by canceling out the random vectors.

To solve the problem of network latency, we add a noise β_i in the encrypted gradients $\hat{a_i}$, $\hat{b_i}$ $(i \in [0, n-1])$. Each user U_i $(i \in [0, n-1])$ utilizes the secret sharing technique to share the noise β_i to other users in the group. The encrypted gradients a'_i, b'_i $(i \in [0, n-1])$ are computed as follows: $a'_i = \hat{a_i} + F_{s_2}(\beta_i)$ and $b'_i = \hat{b_i} + F_{s_2}(\beta_i)$. Using the above method, the users' local gradients will not be leaked to the cloud server. Meanwhile, the users still can obtain the correct aggregated gradient. After receiving the encrypted gradients from the online users, the cloud server recovers the noises β_i $(i \in I_{online})$ and the private keys SK_j $(j \in I_{offline})$ based on the secret shares provided by more than t users. With the recovered private keys SK_j $(j \in I_{offline})$ of the offline users U_j $(j \in I_{offline})$ and the public keys PK_v $(v \in [0, n-1], v \equiv j(\text{mod}2), v \neq j)$ of the user U_v $(v \in [0, n-1], v \equiv j(\text{mod}2), v \neq j)$, the cloud server can compute the secret values $\alpha_{j,v} = KA.agree(SK_j, PK_v)$ $(j \in I_{offline}, v \in [0, n-1], v \equiv j(\text{mod}2), v \neq j)$ and recovers the random vectors $F_{s_2}(\alpha_{j,v}||R)$ $(j \in I_{offline}, v \in [0, n-1], v \equiv j(\text{mod}2), v \neq j)$. The cloud server performs the aggregation operation as below:

$$\begin{aligned} A = &\sum_{i\in I_{online}} a'_i - \sum_{i\in I_{online}} F_{s_2}(\beta_i) \\ &- \sum_{j\in I_{offline}, v\in[0,n-1], v\equiv j(\text{mod}2), v>j} F_{s_2}(\alpha_{j,v}||R) \\ &+ \sum_{j\in I_{offline}, v\in[0,n-1], v\equiv j(\text{mod}2), v<j} F_{s_2}(\alpha_{j,v}||R) \\ B = &\sum_{i\in I_{online}} b'_i - \sum_{i\in I_{online}} F_{s_2}(\beta_i) \\ &- \sum_{j\in I_{offline}, v\in[0,n-1], v\equiv j(\text{mod}2), v>j,} F_{s_2}(\alpha_{j,v}||R) \\ &+ \sum_{j\in I_{offline}, v\in[0,n-1], v\equiv j(\text{mod}2), v<j} F_{s_2}(\alpha_{j,v}||R) \end{aligned}$$

As a result, the cloud server can remove the random vectors $F_{s_2}(\beta_i)$ $(i \in I_{online})$ of the online users and the random vectors $F_{s_2}(\alpha_{j,v}||R)$ $(j \in I_{offline}, v \in [0, n-1], v \equiv j(\text{mod}2))$ of the offline users during the aggregation phase. The users can obtain the correct aggregated encrypted gradients. Based on the aggregated encrypted gradients, the users can compute the aggregated gradient.

To verify the correctness of aggregated encrypted gradients, we design a novel double gradient verification method. Inspired of VerSA [11], we use the current iteration round number to verify the correctness of aggregated encrypted gradients to resist replay attacks. In VerSA [11], the user u computes $\alpha = \sum_{v\in\mu} s_{u,v}$, where μ is the set of current online users and $s_{u,v}$ $(v \in \mu)$ are the secret values negotiated by the user u with other users. Then the user u calculates two vectors $a = PRG(\alpha||0)$ and $b = PRG(\alpha||1)$. The user u uses a and b to encrypt the local gradient x_u and obtain a model verification code $F(x_u) = a \circ x_u + b$. Each user u sends the masked gradient y_u and the model verification code $F(x_u)$ to the cloud server. The cloud server respectively aggregates y_u and $F(x_u)$ to obtain the aggregated gradient $z = \sum_{u\in\mu} y_u$ and the aggregated model verification code $z' = \sum_{u\in\mu} F(x_u)$. The cloud server returns z and z' to all users. Each user can verify the correctness of the aggregated gradient z by checking whether the equation $z' = a \circ z + |\mu| \cdot b$ holds. However, when the online users involved in two iterations are the same, α remains consistent. Consequently, the verification parameters a and b also remain unchanged. This implies that the cloud server could exploit the previous aggregated gradient z and its corresponding aggregated model verification code z' to deceive the users into passing the verification.

To resist replay attack, we use the current iteration round number R to blind and encrypt the local gradients. Consequently, the aggregated encrypted gradients A and B contain the iteration round number R. In the verification phase, the cloud server utilizes the current iteration round number R to verify the correctness of aggregated encrypted gradients A and B. Thus, the cloud server cannot use the previous aggregated encrypted gradients to pass the verification.

5.2 Description of the proposed scheme

Our proposed scheme contains the following four phase: initialization phase, training phase, aggregation phase, and verification and update phase. In the initialization phase, KGC initializes the model parameters of the neural network, then generates security parameters and public/private key pairs for users. KGC sends the corresponding public/private key pair to each user. Each user utilizes the secret sharing technique to share his private key and the noise to other users in the group. In the training phase, each user trains the neural network by using his local dataset. Then each user blinds the local gradient, encrypts the blinded gradients, and sends the encrypted gradients to the cloud server. In the aggregation phase, the cloud server aggregates the encrypted gradients sent by the users and sends the aggregated encrypted gradients to all online users. In the verification and update phase, after receiving the aggregated encrypted gradients from the cloud server, the online users verify the correctness of the aggregated encrypted gradients. If the aggregated encrypted gradients can pass the verification, the users recover the aggregated gradient from the aggregated encrypted gradients and locally updates the trainable parameters. The specific flow is shown in Fig. 2.

1. **Initialization phase**

   (a) KGC initializes the model parameter W of the neural network and setups the learning rate η based on the neural network architecture negotiated by the users. KGC generates two public-private key pairs (PK_i, SK_i), (spk_i, ssk_i) and a noise β_i for each user U_i $(i \in [0, n-1])$. The first public-private key pair (PK_i, SK_i) $(i \in [0, n-1])$ is utilized to generate the secret value, and the second public-private key pair (spk_i, ssk_i) $(i \in [0, n-1])$ is used to encrypt the secret shares of the private key. KGC also generates two secret seeds s_1, $s_2 \in \mathbb{Z}_q^*$ and two pseudo-random functions $\pi_{s_1}(\cdot)$ and $F_{s_2}(\cdot)$. The secret seed s_1 is used to blind the local gradients and verify the correctness of aggregated encrypted gradients. The secret seed s_2 is used to encrypt the blinded gradients.

   (b) KGC sends the secret seeds s_1, s_2 and the public-private key pairs (PK_i, SK_i), (spk_i, ssk_i) to each user U_i $(i \in [0, n-1])$ and publishes the pseudo-random functions $\pi_{s_1}(\cdot)$ and $F_{s_2}(\cdot)$. KGC sends the secret seed s_2 to the cloud server.

   (c) Each user U_i $(i \in [0, n-1])$ sends his public key (PK_i, spk_i) and the identity i to the cloud server and other users in the group U.

   (d) Each user U_i $(i \in [0, n-1])$ chooses a random polynomial

$$h_i(x) = SK_i + \sum_{d=1}^{t-1} \rho_{i,d} x^d \bmod q \ (\rho_{i,d} \in \mathbb{Z}_q^*), \tag{1}$$

   where $\rho_{i,d}$ $(d \in [1, t-1])$ are the random values used to generate the random polynomial $h_i(x)$ and the commitment values. The user U_i $(i \in [0, n-1])$ caculates his private key SK_i's secret share $SK_{i,j} = h_i(j)$ $(j \in [0, n-1], j \neq i)$, then generates and publishes the commitment values g^{SK_i} and $g^{\rho_{i,d}}$ $(d \in [1, t-1])$. The user U_i $(i \in [0, n-1])$ selects the other random polynomial

$$f_i(x) = \beta_i + \sum_{d=1}^{t-1} \delta_{i,d} x^d \bmod q \ (\delta_{i,d} \in \mathbb{Z}_q^*), \tag{2}$$

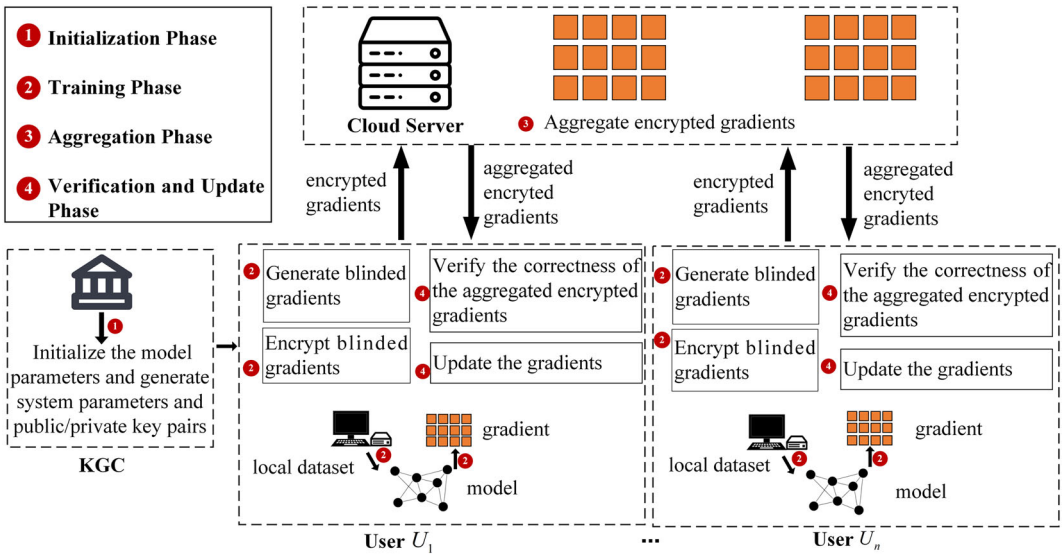


Fig. 2 Architecture of the PPFLV

where $\delta_{i,d}$ $(d \in [1, t-1])$ are the random values used to generate the random polynomial $f_i(x)$ and the commitment values. The user U_i $(i \in [0, n-1])$ computes the noise β_i's secret share $\beta_{i,j} = f_i(j)$ $(j \in [0, n-1])$, then generates and publishes the commitment values g^{β_i} and $g^{\delta_{i,d}}$ $(d \in [1, t-1])$.

(e) The user U_i $(i \in [0, n-1])$ encrypts his secret shares $SK_{i,j}$ $(j \in [0, n-1], j \neq i)$ and $\beta_{i,j}$ $(j \in [0, n-1], j \neq i)$ in the authenticated encryption manner:

$$c_{i,j} \leftarrow AE.enc(KA.agree(spk_j, ssk_i), SK_{i,j} \| i \| j)$$
$$c'_{i,j} \leftarrow AE.enc(KA.agree(spk_j, ssk_i), \beta_{i,j} \| i \| j)$$

The user U_i $(i \in [0, n-1])$ sends the ciphertexts $c_{i,j}$ $(j \in [0, n-1], j \neq i)$ and $c'_{i,j}$ $(j \in [0, n-1], j \neq i)$ to the cloud server. The cloud server broadcasts $\{c_{i,j}, c'_{i,j} | i \in [0, n-1], i \neq j\}$ to each user U_j in the group.

(f) After receiving the ciphertexts $c_{i,j}$, $c'_{i,j}$ $(i \in [0, n-1], i \neq j)$ from the cloud server, each user U_j $(j \in [0, n-1])$ respectively decrypts the ciphertexts $c_{i,j}$ and $c'_{i,j}$ as follows:

$$SK_{i,j} \leftarrow AE.dec(KA.agree(spk_i, ssk_j), c_{i,j} \| i \| j)$$
$$\beta_{i,j} \leftarrow AE.dec(KA.agree(spk_i, ssk_j), c'_{i,j} \| i \| j)$$

The user U_j $(j \in [0, n-1])$ verifies whether the secret shares $SK_{i,j}$ $(i \in [0, n-1], i \neq j)$ and $\beta_{i,j}$ $(i \in [0, n-1], i \neq j)$ are correct by following the equations:

$$g^{SK_{i,j}} = g^{SK_i} \prod_{d=1}^{t-1} (g^{\rho_{i,d}})^{j^d} \quad (3)$$

and

$$g^{\beta_{i,j}} = g^{\beta_i} \prod_{d=1}^{t-1} (g^{\delta_{i,d}})^{j^d} \quad (4)$$

If the equations (3), (4) hold, the user U_j believes $SK_{i,j}$ $(i \in [0, n-1], i \neq j)$ and $\beta_{i,j}$ $(i \in [0, n-1], i \neq j)$ are correct. The user U_j $(j \in [0, n-1])$ stores the secret shares $SK_{i,j}$ and $\beta_{i,j}$ from the users U_i $(i \in [0, n-1], i \neq j)$.

2. **Training phase**

(a) Each user U_i $(i \in [0, n-1])$ trains the neural network with his local dataset $D_i = \{< x_l, y_l >\}_{l \in [1,K]}$ and computes the local gradient ω_i. The user U_i computes the loss function in the R-th iteration as follows:

$$L_\varphi(D'_i, W) = \frac{1}{|D'_i|} \sum_{(x_l, y_l) \in D'_i} C(\varphi(x_l, W), y_l)$$

where $\varphi()$ represents neural network, W represents trainable parameters in $\varphi()$, D'_i is the subset of D_i, and $C()$ represents the criterion to compute the discrepancy between the network's output $\varphi(x_l, W)$ and the label data y_l. The user U_i computes the local gradient $\omega_i = \nabla L_\varphi(D'_i, W)$, where ∇ denotes the vector differential operator.

(b) Algorithm 1 describes the processes of gradient blinding and encryption. To protect privacy, the user U_i $(i \in [0, n-1])$ uses the secret seed s_1 to blind the local gradient ω_i as $a_i = \omega_i + \pi_{s_1}(R)$ and $b_i = \omega_i \cdot \pi_{s_1}(R)$ under a pseudo-random function $\pi_{s_1}(\cdot)$, where R is the current iteration number. The user U_i $(i \in [0, n-1])$ computes the secret value $\alpha_{i,j} = KA.agree(SK_i, PK_j)$ with his private key SK_i and the public key PK_j of the user U_j $(j \in [0, n-1], j \equiv i(\text{mod}2), j \neq i)$. Note that $\alpha_{i,j} = \alpha_{j,i}$. Based on the noise β_i, the secret value $\alpha_{i,j}$ $(j \in [0, n-1], j \equiv i(\text{mod}2), j \neq i)$, the current iteration number R and pseudo-random function $F_{s_2}(\cdot)$, the user U_i encrypts the blinded gradients a_i and b_i as follows:

$$a_i' = a_i + F_{s_2}(\beta_i) + \sum_{j \equiv i(\text{mod}2), j > i} F_{s_2}(\alpha_{i,j} \| R) - \sum_{j \equiv i(\text{mod}2), j < i} F_{s_2}(\alpha_{i,j} \| R) \tag{5}$$

$$b_i' = b_i + F_{s_2}(\beta_i) + \sum_{j \equiv i(\text{mod}2), j > i} F_{s_2}(\alpha_{i,j} \| R) - \sum_{j \equiv i(\text{mod}2), j < i} F_{s_2}(\alpha_{i,j} \| R) \tag{6}$$

(c) The user U_i $(i \in [0, n-1])$ sends the encrypted gradients a_i', b_i' to the cloud server.

3. **Aggregation phase**

(a) After receiving the encrypted gradients a_i', b_i' from the user U_i $(i \in [0, n-1])$, the cloud server sets the identity set of all online users to I_{online}, and sets the identity set of all offline users to $I_{offline} = \{0, 1, ..., n-1\} - I_{online}$. The cloud server verifies whether $I_{online} \geq t$, where t is the threshold value of the secret sharing. If $I_{online} \geq t$, the cloud server broadcasts the identity sets I_{online} and $I_{offline}$ to all online users.

(b) The online user U_i $(i \in I_{online}, v \in I_{offline})$ sends the secret shares $SK_{v,i}$ $(i \in I_{online})$ of the private keys SK_v $(v \in I_{offline})$ of all offline users U_v $(v \in I_{offline})$ and the secret shares $\beta_{i,j}$ $(i, j \in I_{online}, j \neq i)$ of the noise β_i to the cloud server.

(c) The cloud server recovers the private keys SK_v $(v \in I_{offline})$ of all offline users U_v $(v \in I_{offline})$ and the noises β_i $(i \in I_{online})$ of all online users U_i $(i \in I_{online})$ based on the received secret shares $SK_{v,i}$ $(v \in I_{offline}, i \in I_{online})$ and the secret shares $\beta_{i,j}$ $(i, j \in I_{online}, j \neq i)$, respectively. Then, the cloud server computes the secret values $\alpha_{v,j} = KA.agree(SK_v, PK_j)$ $(v \in I_{offline}, j \in [0, n-1], j \equiv v(\text{mod}2), j \neq v)$ by using the private keys SK_v $(v \in I_{offline})$ of offline users U_v $(v \in I_{offline})$ and the public keys PK_j $(j \in [0, n-1], j \equiv v(\text{mod}2), j \neq v)$ of the users U_j $(j \in [0, n-1], j \equiv v(\text{mod}2), j \neq v)$.

(d) The cloud server performs the double gradient aggregation operation as follows:

Algorithm 1 Gradient blinding and encryption algorithm

Input: private gradient ω_i, iteration number R, noise β_i, secret seeds s_1 and s_2, pseudo-random functions $\pi_{s_1}(\cdot)$ and $F_{s_2}(\cdot)$.
Output: encrypted gradients a_i', b_i'.
1: Blind the gradient ω_i:

$a_i = \omega_i + \pi_{s_1}(R)$, $b_i = \omega_i \cdot \pi_{s_1}(R)$

2: Compute the secret value $\alpha_{i,j}$:

$\alpha_{i,j} = KA.agree(SK_i, PK_j)$ $(j \in [0, n-1], j \equiv i(\text{mod})2, j \neq i)$

3: Encrypt the blinded gradients a_i and b_i:

$$a_i' = a_i + F_{s_2}(\beta_i) + \sum_{j \equiv i(\text{ mod }2), j > i} F_{s_2}(\alpha_{i,j} \| R) - \sum_{j \equiv i(\text{ mod }2), j < i} F_{s_2}(\alpha_{i,j} \| R)$$

$$b_i' = b_i + F_{s_2}(\beta_i) + \sum_{j \equiv i(\text{ mod }2), j > i} F_{s_2}(\alpha_{i,j} \| R) - \sum_{j \equiv i(\text{ mod }2), j < i} F_{s_2}(\alpha_{i,j} \| R)$$

4: Return a_i' and b_i'.

$$A = \sum_{i \in I_{online}} a_i' - \sum_{i \in I_{online}} F_{s_2}(\beta_i) - \sum_{v \in I_{offline}, j \in [0,n-1], j \equiv v (\mathrm{mod} 2), j > v} F_{s_2}(\alpha_{v,j} \| R) + \sum_{v \in I_{offline}, j \in [0,n-1], j \equiv v (\mathrm{mod} 2), j < v} F_{s_2}(\alpha_{v,j} \| R) \quad (7)$$

$$B = \sum_{i \in I_{online}} b_i' - \sum_{i \in I_{online}} F_{s_2}(\beta_i) - \sum_{v \in I_{offline}, j \in [0,n-1], j \equiv v (\mathrm{mod} 2), j > v} F_{s_2}(\alpha_{v,j} \| R) + \sum_{v \in I_{offline}, j \in [0,n-1], j \equiv v (\mathrm{mod} 2), j < v} F_{s_2}(\alpha_{v,j} \| R) \quad (8)$$

The cloud sever sends the aggregated encrypted gradients A and B to all online users U_i $(i \in I_{online})$.

4. **Verification and update phase** Algorithm 2 describes the processes of verification and update. After receiving the aggregated encrypted gradients A and B from the cloud server, the online users U_i $(i \in I_{online})$ verify whether the cloud server correctly performs the aggregation operation through the following equation:

$$A - |I_{online}| \cdot \pi_{s_1}(R) = B \cdot \frac{1}{\pi_{s_1}(R)} \quad (9)$$

If the equation (9) holds, it means that the aggregated encrypted gradients A and B are correct. Then, the user U_i $(i \in I_{online})$ calculates the aggregated gradient $\omega = A - |I_{online}| \cdot \pi_{s_1}(R)$ and locally updates the trainable parameter $W = W - \eta \cdot \frac{\omega}{|I_{online}|}$. The next round of federated learning will be executed until the termination condition is satisfied.

6 Security analysis

Theorem 1 (Correctness) *When the cloud server correctly aggregates all online users' encrypted gradients, the aggregated encrypted gradients can pass the users' verification.*

Proof To simplify the following derivation, we suppose that the cloud server is able to receive encrypted gradients (a_i', b_i') sent by the online user U_i $(i \in I_{online})$. Here, we consider the cases of offline users and network latency. The cloud server identifies the users with network latency as offline users because these users cannot upload the encrypted gradients to the cloud server on time.

The cloud server aggregates all online users' a_i' $(i \in I_{online})$ as follows:

$$\begin{aligned}\sum_{i \in I_{online}} a_i' &= \sum_{i \in I_{online}} a_i + \sum_{i \in I_{online}} F_{s_2}(\beta_i) + \sum_{i \in I_{online}, j \in [0,n-1], j \equiv i (\mathrm{mod} 2), j > i} F_{s_2}(\alpha_{i,j} \| R) - \sum_{i \in I_{online}, j \in [0,n-1], j \equiv i (\mathrm{mod} 2), j < i} F_{s_2}(\alpha_{i,j} \| R) \\ &= \sum_{i \in I_{online}} a_i + \sum_{i \in I_{online}} F_{s_2}(\beta_i) + \sum_{i \in I_{online}, j \in I_{online}, j \equiv i (\mathrm{mod} 2), j > i} F_{s_2}(\alpha_{i,j} \| R) + \sum_{v \in I_{offline}} \sum_{j \in [0,n-1], j \equiv v (\mathrm{mod} 2), j > v} F_{s_2}(\alpha_{v,j} \| R) - \\ &\quad \sum_{i \in I_{online}, j \in I_{online}, j \equiv i (\mathrm{mod} 2), j < i} F_{s_2}(\alpha_{i,j} \| R) - \sum_{v \in I_{offline}} \sum_{j \in [0,n-1], j \equiv v (\mathrm{mod} 2), j < v} F_{s_2}(\alpha_{v,j} \| R)\end{aligned}$$

$F_{s_2}(\alpha_{i,j} \| R)$ and $F_{s_2}(\alpha_{j,i} \| R)$ generated by the users U_i $(i \in I_{online})$ and U_j $(j \in I_{online})$ can cancel each other out. $F_{s_2}(\alpha_{v,j} \| R)$ between the offline users U_v $(v \in I_{offline})$ and

Algorithm 2 Verification and update algorithm

Input: aggregated encrypted gradients A and B, iteration number R, secret seeds s_1, pseudo-random functions $\pi_{s_1}(\cdot)$, set I_{online} of online users and learning rate η.

Output: the trainable parameter W.

1: Verify whether the cloud server correctly performs the aggregation operation: $A - |I_{online}| \cdot \pi_{s_1}(R) = B \cdot \frac{1}{\pi_{s_1}(R)}$.

2: Calculate aggregated gradient: $\omega = A - |I_{online}| \cdot \pi_{s_1}(R)$.

3: Update the trainable parameter: $W = W - \eta \cdot \frac{\omega}{|I_{online}|}$

4: Return the trainable parameter W.

the users U_j $(j \in [0, n-1], j \equiv v(\text{mod})2, j > v)$, $F_{s_2}(\alpha_{v,j}\|R)$ between the offline users U_v $(v \in I_{offline})$ and the users U_j $(j \in [0, n-1],\ j \equiv v(\text{mod})2, j < v)$, and the $\sum_{i \in I_{online}} F_{s_2}(\beta_i)$ cannot be cancelled out. Thus,

$$\sum_{i \in I_{online}} a_i' = \sum_{i \in I_{online}} a_i + \sum_{i \in I_{online}} F_{s_2}(\beta_i) + \sum_{v \in I_{offline}} \sum_{j \in [0,n-1], j \equiv v(\text{mod}2), j > v} F_{s_2}(\alpha_{v,j}\|R) - \sum_{v \in I_{offline}} \sum_{j \in [0,n-1], j \equiv v(\text{mod}2), j < v} F_{s_2}(\alpha_{v,j}\|R)$$

The cloud server can recover the noise β_i $(i \in I_{online})$ and the private key SK_v $(v \in I_{offline})$ based on the secret shares provided by more than t users. With the recovered private key SK_v of the offline user U_v $(v \in I_{offline})$ and the public key PK_j of the user U_j $(j \in [0, n-1], j \equiv v(\text{mod})2, j > v)$, the cloud server can compute the secret value $\alpha_{v,j} = KA.agree(SK_v, PK_j)$ and recovers the random vectors $F_{s_2}(\alpha_{v,j}\|R)$ $(v \in I_{offline}, j \in [0, n-1], j \equiv v(\text{mod})2, j > v)$. Similarly, with the recovered private key SK_j of the offline user U_v $(v \in I_{offline})$ and the public key PK_j of the user U_j $(j \in [0, n-1], j \equiv v(\text{mod})2, j < v)$, the cloud server can compute the secret value $\alpha_{v,j} = KA.agree(SK_v, PK_j)$ and recovers the random vectors $F_{s_2}(\alpha_{v,j}\|R)$ $(v \in I_{offline}, j \in [0, n-1], j \equiv v(\text{mod})2, j < v)$. Thus, the cloud server can remove the random vectors $F_{s_2}(\alpha_{v,j}\|R)$ generated by the offline users U_v $(v \in I_{offline})$ and the users U_j $(j \in [0, n-1], j \equiv v(\text{mod})2)$ and the noises $F_{s_2}(\beta_i)$ as follows.

$$\begin{aligned} A &= \sum_{i \in I_{online}} a_i' - \sum_{i \in I_{online}} F_{s_2}(\beta_i) \\ &\quad - \sum_{v \in I_{offline}} \sum_{j \in [0,n-1], j \equiv v(\text{mod}2), j > v} F_{s_2}(\alpha_{v,j}\|R) \\ &\quad + \sum_{v \in I_{offline}} \sum_{j \in [0,n-1], j \equiv v(\text{mod}2), j < v} F_{s_2}(\alpha_{v,j}\|R) \\ &= \sum_{i \in I_{online}} a_i + \sum_{i \in I_{online}} F_{s_2}(\beta_i) \\ &\quad + \sum_{v \in I_{offline}} \sum_{j \in [0,n-1], j \equiv v(\text{mod}2), j > v} F_{s_2}(\alpha_{v,j}\|R) - \\ &\quad \sum_{v \in I_{offline}} \sum_{j \in [0,n-1], j \equiv v(\text{mod}2), j < v} F_{s_2}(\alpha_{v,j}\|R) \\ &\quad - \sum_{i \in I_{online}} F_{s_2}(\beta_i) \\ &\quad - \sum_{v \in I_{offline}} \sum_{j \in [0,n-1], j \equiv v(\text{mod}2), j > v} F_{s_2}(\alpha_{v,j}\|R) \\ &\quad + \sum_{v \in I_{offline}} \sum_{j \in [0,n-1], j \equiv v(\text{mod}2), j < v} F_{s_2}(\alpha_{v,j}\|R) \\ &= \sum_{i \in I_{online}} a_i = \sum_{i \in I_{online}} (\omega_i + \pi_{s_1}(R)). \end{aligned}$$

Similarly, the cloud server aggregates b_i' $(i \in I_{online})$ as follows:

$$\begin{aligned} \sum_{i \in I_{online}} b_i' &= \sum_{i \in I_{online}} b_i + \sum_{i \in I_{online}} F_{s_2}(\beta_i) + \\ &\quad \sum_{i \in I_{online}, j \in [0,n-1], j \equiv i(\text{mod}2), j > i} F_{s_2}(\alpha_{i,j}\|R) - \\ &\quad \sum_{i \in I_{online}, j \in [0,n-1], j \equiv i(\text{mod}2), j < i} F_{s_2}(\alpha_{i,j}\|R) \\ &= \sum_{i \in I_{online}} b_i + \sum_{i \in I_{online}} F_{s_2}(\beta_i) + \\ &\quad \sum_{i \in I_{online}, j \in I_{online}, j \equiv i(\text{mod}2), j > i} F_{s_2}(\alpha_{i,j}\|R) + \\ &\quad \sum_{v \in I_{offline}} \sum_{j \in [0,n-1], j \equiv v(\text{mod}2), j > v} F_{s_2}(\alpha_{v,j}\|R) - \\ &\quad \sum_{i \in I_{online}, j \in I_{online}, j \equiv i(\text{mod}2), j < i} F_{s_2}(\alpha_{i,j}\|R) - \\ &\quad \sum_{v \in I_{offline}} \sum_{j \in [0,n-1], j \equiv v(\text{mod}2), j < v} F_{s_2}(\alpha_{v,j}\|R) \end{aligned}$$

$F_{s_2}(\alpha_{i,j}\|R)$ and $F_{s_2}(\alpha_{j,i}\|R)$ generated by the users U_i $(i \in I_{online})$ and U_j $(j \in I_{online})$ can cancel each other out. $F_{s_2}(\alpha_{v,j}\|R)$ between the offline users U_v $(v \in I_{offline})$ and the users U_j $(j \in [0, n-1], j \equiv v(\text{mod})2, j > v)$, $F_{s_2}(\alpha_{v,j}\|R)$ between the offline users U_v $(v \in I_{offline})$ and the users U_j $(j \in [0, n-1],\ j \equiv v(\text{mod})2, j < v)$, and the $\sum_{i \in I_{online}} F_{s_2}(\beta_i)$ cannot be cancelled out. Therefore,

$$\sum_{i \in I_{online}} b_i' = \sum_{i \in I_{online}} b_i + \sum_{i \in I_{online}} F_{s_2}(\beta_i) + \sum_{v \in I_{offline}} \sum_{j \in [0,n-1], j \equiv v(\text{mod}2), j > v} F_{s_2}(\alpha_{v,j}\|R) - \sum_{v \in I_{offline}} \sum_{j \in [0,n-1], j \equiv v(\text{mod}2), j < v} F_{s_2}(\alpha_{v,j}\|R)$$

The cloud server can recover the noise β_i $(i \in I_{online})$ and the private key SK_v $(v \in I_{offline})$ based on the secret shares provided by more than t users. With the recovered private key SK_v of the offline user U_v $(v \in I_{offline})$ and the public key PK_j of the user U_j $(j \in [0, n-1], j \equiv v(\text{mod})2, j > v)$, the cloud server can compute the secret value $\alpha_{v,j} = KA.agree(SK_v, PK_j)$ and recovers the random vectors $F_{s_2}(\alpha_{v,j}\|R)$ $(v \in I_{offline}, j \in [0, n-1], j \equiv v(\text{mod})2, j > v)$. Similarly, with the recovered private key SK_j of the offline user U_v $(v \in I_{offline})$ and the public key PK_j of the user U_j $(j \in [0, n-1], j \equiv v(\text{mod})2, j < v)$, the cloud server can compute the secret value $\alpha_{v,j} = KA.agree(SK_v, PK_j)$ and recovers the random vectors $F_{s_2}(\alpha_{v,j}\|R)$ $(v \in I_{offline}, j \in [0, n-1], j \equiv v(\text{mod})2, j < v)$. Thus, the cloud server can remove the random vectors $F_{s_2}(\alpha_{v,j}\|R)$ generated by the offline users U_v $(v \in I_{offline})$ and the users

U_j $(j \in [0, n-1], j \equiv v(\text{mod}2)$ and the noises $F_{s_2}(\beta_i)$ as follows.

$$\begin{aligned} B = & \sum_{i \in I_{online}} b_i' - \sum_{i \in I_{online}} F_{s_2}(\beta_i) \\ & - \sum_{v \in I_{offline}} \sum_{j \in [0,n-1], j \equiv v(\text{mod}2), j > v} F_{s_2}(\alpha_{v,j} \| R) \\ & + \sum_{v \in I_{offline}} \sum_{j \in [0,n-1], j \equiv v(\text{mod}2), j < v} F_{s_2}(\alpha_{v,j} \| R) \\ = & \sum_{i \in I_{online}} b_i + \sum_{i \in I_{online}} F_{s_2}(\beta_i) \\ & + \sum_{v \in I_{offline}} \sum_{j \in [0,n-1], j \equiv v(\text{mod}2), j > v} F_{s_2}(\alpha_{v,j} \| R) \\ & - \sum_{v \in I_{offline}} \sum_{j \in [0,n-1], j \equiv v(\text{mod}2), j < v} F_{s_2}(\alpha_{v,j} \| R) \\ & - \sum_{i \in I_{online}} F_{s_2}(\beta_i) - \\ & \sum_{v \in I_{offline}} \sum_{j \in [0,n-1], j \equiv v(\text{mod}2), j > v} F_{s_2}(\alpha_{v,j} \| R) + \\ & \sum_{v \in I_{offline}} \sum_{j \in [0,n-1], j \equiv v(\text{mod}2), j < v} F_{s_2}(\alpha_{v,j} \| R) \\ = & \sum_{i \in I_{online}} b_i = \sum_{i \in I_{online}} (\omega_i \cdot \pi_{s_1}(R)). \end{aligned}$$

If the equation $A - |I_{online}| \cdot \pi_{s_1}(R) = B \cdot \frac{1}{\pi_{s_1}(R)} = \sum_{i \in I_{online}} \omega_i$ holds, all online users are convinced that the cloud server correctly aggregates all users' encrypted gradients. □

Theorem 2 (Gradient privacy) *The cloud server cannot obtain the users' local gradients from the encrypted gradients sent by the users.*

Proof In our scheme, we adopt double gradient blinding and encryption method to blind and encrypt the local gradient ω_i as follows:

$$\begin{aligned} a_i' = & a_i + F_{s_2}(\beta_i) + \sum_{j \equiv i(\text{mod}2), j > i} F_{s_2}(\alpha_{i,j} \| R) \\ & - \sum_{j \equiv i(\text{mod}2), j < i} F_{s_2}(\alpha_{i,j} \| R) \\ b_i' = & b_i + F_{s_2}(\beta_i) + \sum_{j \equiv i(\text{mod}2), j > i} F_{s_2}(\alpha_{i,j} \| R) \\ & - \sum_{j \equiv i(\text{mod}2), j < i} F_{s_2}(\alpha_{i,j} \| R), \end{aligned}$$

where $a_i = \omega_i + \pi_{s_1}(R)$ and $b_i = \omega_i \cdot \pi_{s_1}(R)$, $\pi_{s_1}(R)$ is a random vector produced by the user based on the current iteration number R. The local gradient ω_i is blinded by the random vector $\pi_{s_1}(R)$. Then, the blinded gradients a_i and b_i are encrypted by the random vectors $F_{s_2}(\alpha_{i,j} \| R)$ and $F_{s_2}(\beta_i)$, where the secret value $\alpha_{i,j} = KA.agree(SK_i, PK_j)$ can be calculated based on the user U_i $(i \in [0, n-1])$'s private key SK_i and the user U_j $(j \in [0, n-1], j \equiv i(\text{mod}2), j \neq i)$'s public key PK_j and the noise β_i is randomly generated by the *GM*.

The cloud server receives the encrypted gradients a_i' and b_i' from the online users U_i $(i \in I_{online})$ and aslo receives the encrypted gradients a_j' and b_j' from the the users U_j $(j \in I_{latency})$ with network latency. In the aggregation phase, the cloud server is able to recover the private keys SK_j of the users U_j $(j \in I_{latency})$ with network latency and the noise β_i of all online users U_i $(i \in I_{online})$ based on the received secret shares. However, the cloud server cannot recover the private keys SK_i of online users U_i $(i \in I_{online})$ and the noise β_j of the users U_j $(j \in I_{latency})$ with network latency since the cloud server cannot collude with more than t users to recover SK_i $(i \in I_{online})$ and β_j $(j \in I_{latency})$. As a result, the cloud server cannot caculate the random vectors $F_{s_2}(\beta_j)$ of the user U_j $(j \in I_{latency})$ with network latency without the noises β_j $(j \in I_{latency})$ and aslo cannot compute the online user U_i $(i \in I_{online})$'s secret value $\alpha_{i,j} = KA.agree(SK_i, PK_j)$ $(j \in [0, n-1], j \equiv i(\text{mod}2), j \neq i)$ without the private key SK_i $(i \in I_{online})$. Further, the cloud server cannot caculate the random vectors $F_{s_2}(\alpha_{i,j} \| R)$ $(i \in I_{online}, j \in [0, n-1], j \equiv i(\text{mod}2), j \neq i)$. Thus, even if the cloud server can obtain the random vectors $F_{s_2}(\beta_i)$ of all online users U_i $(i \in I_{online})$, without the random vectors $F_{s_2}(\alpha_{i,j} \| R)$ $(i \in I_{online}, j \in [0, n-1], j \equiv i(\text{mod}2), j \neq i)$, it still cannot obtain the blinded gradients a_i and b_i from the encrypted gradients a_i' and b_i' sent by the online users U_i $(i \in I_{online})$, let alone the local gradients ω_i $(i \in I_{online})$. In addition, even if the cloud server can calculate the random vectors $F_{s_2}(\alpha_{j,i} \| R)$ $(j \in I_{latency}, i \in [0, n-1], i \equiv j(\text{mod}2), i \neq j)$ of the user U_j $(j \in I_{latency})$ with network latency based on the recovered private key SK_j $(j \in I_{latency})$, without the noises β_j $(j \in I_{latency})$, it still cannot obtain the blinded gradients a_j and b_j from the encrypted gradients a_j' and b_j' from the users U_j $(j \in I_{latency})$, let alone the local gradients ω_j $(j \in I_{latency})$. □

Theorem 3 (Immunity from replay attacks) *In PPFLV, the cloud server is not able to pass the verification of the users by utilizing the previous aggregated gradients.*

Proof We prove that PPFLV can resist replay attacks through the following game. Specially, in the R_2-th iteration, the user U_i $(i \in [0, n-1])$ sends the new encrypted gradients $\{a_i^*, b_i^*\}$ to the cloud server, which are different from the previous encrypted gradients $\{a_i', b_i'\}$ in the previous R_1-th iteration. Then, the cloud server transmits the previous aggregated encrypted gradients $\{A, B\}$ to the user U_i $(i \in [0, n-1])$, where $\{A, B\}$ are the aggregated

encrypted gradients corresponding to the previous encrypted gradients $\{a_i', b_i'\}$. During the verification phase, if $\{A, B\}$ are able to pass the verification of the users, the cloud server wins this game; otherwise, it fails.

For the previous aggregated encrypted gradients $\{A, B\}$ in the R_1-th iteration, the following verification equation holds based on the iteration number R_1:

$$A - |I_{online}| \cdot \pi_{s_1}(R_1) = B \cdot \frac{1}{\pi_{s_1}(R_1)}. \tag{10}$$

In the new R_2-th iteration, if the cloud server can use $\{A, B\}$ to pass the verification, then the following equation also holds:

$$A - |I_{online}| \cdot \pi_{s_1}(R_2) = B \cdot \frac{1}{\pi_{s_1}(R_2)}. \tag{11}$$

□

From the above two verification equations (10), (11), we have

$$\begin{aligned} A &= B \cdot \frac{1}{\pi_{s_1}(R_1)} + |I_{online}| \cdot \pi_{s_1}(R_1) \\ &= B \cdot \frac{1}{\pi_{s_1}(R_2)} + |I_{online}| \cdot \pi_{s_1}(R_2) \end{aligned}$$

Further, we have $R_1 = R_2$, which is in contradiction with the assumption that $R_1 \neq R_2$. Therefore, the cloud server cannot win this game. In VerSA [11], if the online users in any two rounds are the same, then the parameters of the verification equation will become the same. In this case, the cloud server has the opportunity to launch a replay attack. The difference from VerSA [11] is that the equation parameters used to verify the correctness of the aggregate gradient in any round of PPFLV are different. This ensures that the cloud server does not have any chance to launch a replay attack in PPFLV.The PPFLV is able to resist the replay attacks.

Theorem 4 (Unforgeability) *In the PPFLV scheme, if the cloud server does not correctly aggregate the users' encrypted gradients, it cannot pass the verification of the users.*

Proof If A, B are the correct aggregated encrypted gradients sent by the cloud server, then the following verification equation holds: $A - |I_{online}| \cdot \pi_{s_1}(R) = B\frac{1}{\pi_{s_1}(R)}$. Assume that the cloud server forges aggregated encrypted gradients A', B' which are different from A, B. If the cloud server is able to pass the verification of the users, we have: $A' - |I_{online}| \cdot \pi_{s_1}(R) = B' \cdot \frac{1}{\pi_{s_1}(R)}$.

Further, we can deduce that A' must be satisfy $A' = num_{fake} + |I_{online}| \cdot \pi_{s_1}(R)$ and B' must satisfy $B' = num_{fake} \cdot \pi_{s_1}(R)$, where num_{fake} is a random vector chosen by the cloud server. It means that if the cloud server can forge successfully, it needs to know the secret seed s_1 of the pseudo-random function $\pi_{s_1}(\cdot)$. Only knowing s_1, the cloud server can compute $A' = num_{fake} + |I_{online}| \cdot \pi_{s_1}(R)$ and $B' = num_{fake} \cdot \pi_{s_1}(R)$, which can make the equation $A' - |I_{online}| \cdot \pi_{s_1}(R) = B' \cdot \frac{1}{\pi_{s_1}(R)}$ hold. However, the secret seed s_1 is randomly selected by the *GM*, which is kept secret from the cloud server. In the finite field $\mathbb{Z}_q^*$, the probability of the cloud server obtaining the correct secret seed is $\frac{1}{q}$, which is negligible since q is large prime. Therefore, it is computationally infeasible for the cloud server to forge the aggregated encrypted gradients to pass the verificaiton. □

7 Performance evaluation

In this section, we evaluate the performance of the PPFLV in terms of classification accuracy, computation overhead and communication overhead. The simulation experiment is conducted on a Linux server with Intel(R) Core(TM) i9-10980XE, 3.0GHZ, and 32GB memory. All experiments are implemented using Pytorch.

7.1 Classification accuracy

We use two datasets (CIFAR10 and MNIST) to evaluate the classification accuracy of two models :

- CIFAR10 [45] is a dataset consisting of three-channel RGB color images categorized into 10 classes. The images in this dataset have a size of 32 × 32 pixels. It contains a total of 60,000 images. We pick 50,000 images for training, while the remaining images are reserved for test.
- MNIST [46] is a handwritten digital dataset. It contains a total of 70,000 images. Each image in this dataset has a size of 28 × 28 pixels and features handwritten numbers ranging from 0 to 9 in white color against a black background. We select 60,000 images for training and the rest for test.

We select FedAvg [47], which is a plaintext federated learning scheme without privacy protection, as a benchmark for comparison with the proposed PPFLV in terms of classification accuracy. To evaluate the performance of the proposed PPFLV, we choose two well-known classical networks: Multilayer Perceptron (MLP) [48] and Convolutional Neural Networks (CNN) [49]. In our experiment, MLP consists of an input layer, two hundred hidden layers and an output layer. CNN trained with the CIFAR dataset is composed of two convolutional layers, one pooling layer

and three fully connected layers. The CNN trained with the MNIST dataset is composed of three convolutional layers and two fully connected layers. We split the MNIST dataset into IID (independent and identically distributed) dataset and non-IID (non-Independent Identically Distribution) dataset.

As depicted in Fig. 3a and c, when utilizing the MNIST dataset under IID, the classification accuracies obtained by PPFLV using both CNN and MLP neural networks for training exceed 90%. Furthermore, from Fig. 3a and c, we can find that, the classification accuracy of PPFLV is similar to that of FedAvg [47]. Figure 3b and d show that two model accuracy in MNIST non-IID database, the proposed PPFLV achieves comparable accuracy to the plaintext FedAvg [47]. Hence, data heterogeneity is connected to federated learning itself, but it does not impact the security design of our scheme. Thus, compared to plaintext federated learning scheme, our PPFLV hardly losses the classification accuracy while maintaining data privacy protection.

7.2 Computation overhead

To evaluate the performance of the proposed PPFLV on the user side, VerSA [11] and VCD-FL[50], which are the state-of-the-art verifiable federated learning schemes with privacy preserving, are selected as the benchmarks. We give the comparison of PPFLV, VerSA[11] and VCD-FL [50] in terms of users' computation overhead for gradient encryption and verification.

We set the number of gradients per user to vary from 1000 to 10000, and the number of users to vary from 100 to 1000. The size of each gradient entry is 64bit. Figure 4 shows the computation overhead comparison of encrypting gradients among the proposed PPFLV, VerSA [11] and VCD-FL [50] for different number of gradients per user and different number of users. Most of the computation overhead of the user for encrypting gradients is generated by the pseudo-random function and the key agreement. The pseudo-random function is used to generate the random vectors and the key agreement is utilized to generate the secret values between the users. Figure 4a and b show that the computation overhead of gradient encryption increases linearly with the number of users and the number of gradients per user in the proposed PPFLV, VerSA [11], and VCD-FL [50]. The computation overhead of gradient encryption in the proposed PPFLV is much lower than that in VCD-FL [50].

We also evaluate the computation overhead of the user during the verification phase. Both the proposed PPFLV and VerSA [11] only require to perform the lightweight calculation to verify the correctness of aggregated gradient. Figure 5a and b show that the computation overhead of verification in the proposed PPFLV, VerSA [11], and VCD-FL [50] is proportional to number of users and the number of gradients per user. In the phase of verification, the computation overhead of the proposed PPFLV is much lower than that in VCD-FL [50], and approximately the same as that of VerSA [11]. However, VerSA [11] cannot resist the replay attack in the verification phase. Thus, the

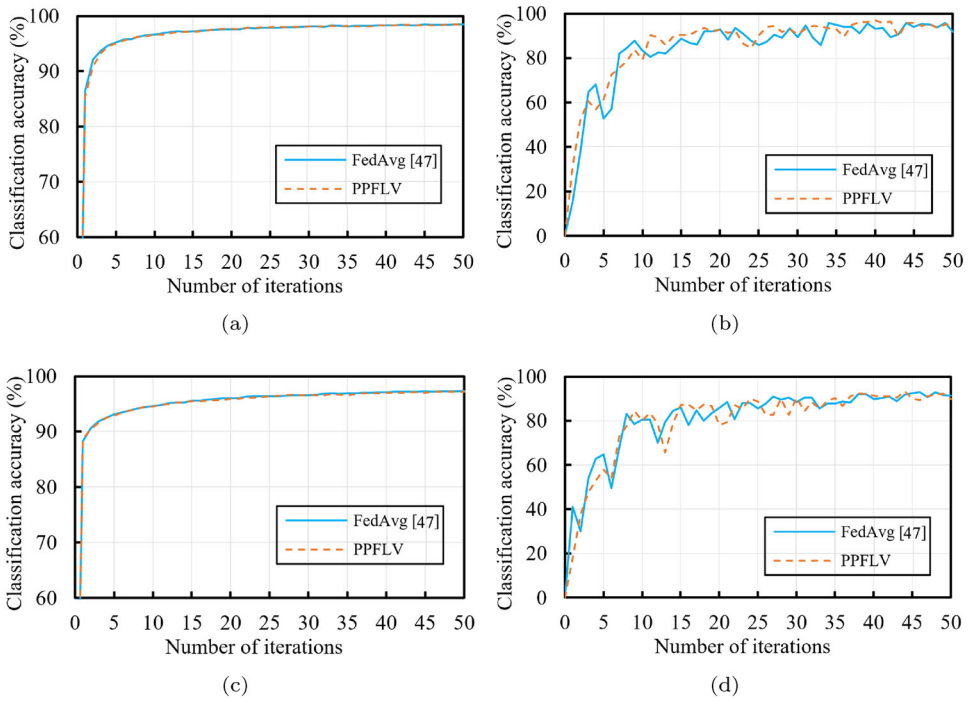


Fig. 3 Classification accuracy comparison of PPFLV and FedAvg [47] **a** CNN using MNIST dataset under IID, **b** CNN using MNIST dataset under non-IID, **c** MLP using MNIST dataset under IID, **d** MLP using MNIST dataset under non-IID

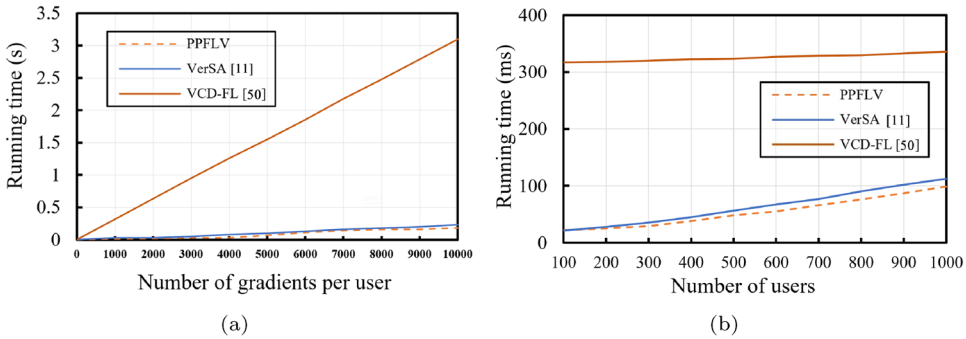


Fig. 4 Comparison of gradient encryption computation overhead **a** with different number of gradients per user, **b** with different number of users

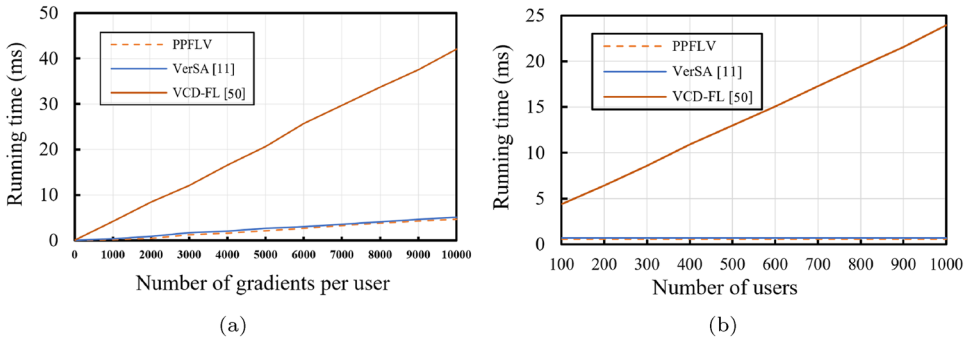


Fig. 5 Comparison of verification computation overhead **a** with different number of gradients per user, **b** with different number of users

proposed PPFLV is able to achieve efficient verification while resisting replay attack.

To evaluate the aggregation performance of the cloud server, we set the number of users to 500 and the gradient entries of each user to 5K and 10K, respectively. The size of each entry is 64bit.

Figure 6 shows the computation overhead of aggregating gradients on the cloud server side when all users are online. In Fig. 6, we have the observation that as the number of iterations increases, the computation overhead of aggregating gradients grows linearly. Furthermore, the computation overhead of aggregating gradients is also related to the number of gradient entries of each user, which increases with the number of gradient entries of each user.

We set the user offline rate from 10% to 50%. As discussed in Sect. 5.2, the cloud server needs to recover the private keys of the offline users when the users are offline. Figure 7a and b show that in PPFLV and VerSA [11], the computation overhead of aggregating gradients is affected by the number of gradient entries of each user when the user offline rate is fixed. The computation overhead of aggregating gradients in PPFLV and VerSA [11] grow linearly as the user offline rate increases. Furthermore, the computation overhead of aggregating gradients in PPFLV is significantly lower than VerSA [11]. This advantage becomes increasingly evident as the user offline rate increases. When the user offline rate reaches 50%, the computation overhead of aggregating gradients in PPFLV

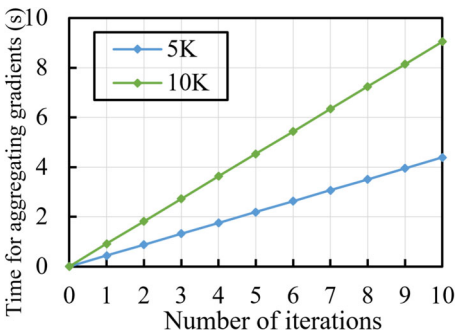


Fig. 6 Computation overhead of aggregating gradients for all users online

is just under half that of VerSA [11]. Thus, we can conclude that the performance of the proposed PPFLV surpasses that of VerSA [11] on the cloud server side.

7.3 Communication overhead

We compare the total communication overhead of our scheme with VerSA [11] and VerifyNet [4]. VerSA [11] and VerifyNet [4] are the state-of-the-art privacy-preserving and verifiable federated learning schemes. The communication overhead mainly comes from the phases of initialization, aggregation and verification. The keys and seeds are distributed in the initialization phase. The encrypted gradients are uploaded in the aggregation phase and the aggregated encrypted gradients are downloaded in the verification phase.

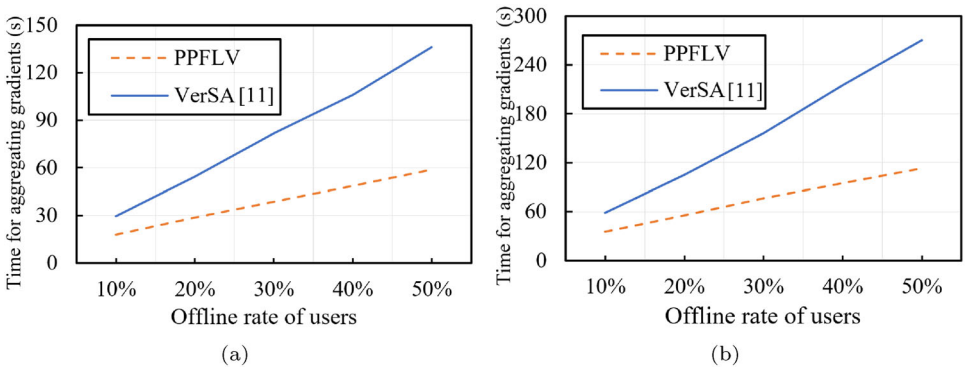


Fig. 7 Computation overhead of aggregating gradients for different user offline rates **a** 5K-entry gradients per user, **b** 10K-entry gradients per user

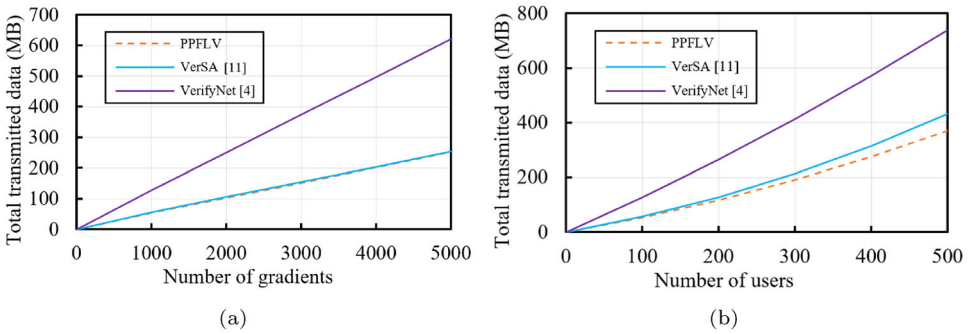


Fig. 8 Communication overhead comparison of PPFLV, VerSA [11] and VerifyNet [4] **a** with different number of gradients per user, **b** with different number of users

Figure 8a shows the communication overhead comparison of PPFLV, VerifyNet [4] and VerSA [11] by gradually increasing the number of gradients per user with a fixed number of 100 users. Figure 8b illustrates the communication overhead comparison of PPFLV, VerifyNet [4] and VerSA [11] when the number of users increases and the number of gradients per user is fixed at 1000 entries. From Fig.8a and b, we can find that the communication overhead of PPFLV, VerifyNet [4] and VerSA [11] increase linearly as the number of gradients per user and the number of users increases. Furthermore, with an increasing number of gradients per user, the communication overhead of PPFLV is significantly lower than that of VerifyNet [4], and is essentially comparable to VerSA [11]. As the number of users grows, the communication overhead of PPFLV is lower than that of VerifyNet [4] and VerSA [11].

8 Conclusion

To solve the problems of the users' private data leakage and the aggregated gradients verification in federated learning, in this paper, we propose PPFLV, a privacy-preserving federated learning scheme with verifiability. In PPFLV, the double gradient blinding and encryption method is used to blind and encrypt the users' local gradients and guarantee the users' privacy. The double gradient verification method is utilized to verify the correctness of the aggregated encrypted gradients calculated by the cloud server. Each user can independently verify the correctness of the aggregated encrypted gradients and recovers the aggregated gradient. The experiments show that PPFLV is efficient in terms of privacy protection and secure verification.

Acknowledgements This work is supported by National Natural Science Foundation of China (62102211), Shandong Provincial Natural Science Foundation (ZR2021QF018), and Shandong Province Higher Education Institutions Youth Innovation and Technology Support Program (2023KJ365).

Author contributions Qun Zhou wrote the main manuscript text and Wenting Shen modified and reviewed the manuscript.

Funding Funding were provided by National Natural Science Foundation of China (Grant No. 62102211) and Shandong Provincial Natural Science Foundation (Grant No. ZR2021QF018).

Data availability No datasets were generated or analysed during the current study.

Declarations

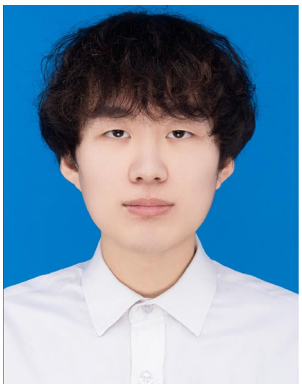
Competing interest The authors declare no competing interests.

References

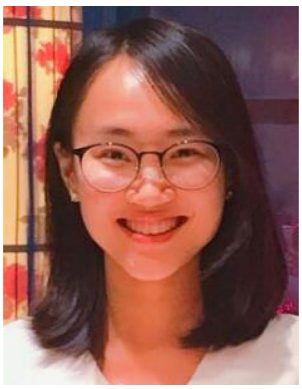
1. Feng, Q., He, D., Liu, Z., Wang, H., Choo, K.-K.R.: Securenlp: a system for multi-party privacy-preserving natural language processing. IEEE Trans. Inf. Forensics Secur. **15**, 3709–3721 (2020)

2. Xiong, Z., Li, W., Han, Q., Cai, Z.: Privacy-preserving auto-driving: a gan-based approach to protect vehicular camera data. In: 2019 IEEE International Conference on Data Mining (ICDM), pp. 668–677. IEEE (2019)
3. Bakator, M., Radosav, D.: Deep learning and medical diagnosis: a review of literature. Multimodal Technol. Interact. **2**(3), 47 (2018)
4. Xu, G., Li, H., Liu, S., Yang, K., Lin, X.: Verifynet: secure and verifiable federated learning. IEEE Trans. Inf. Forensics Secur. **15**, 911–926 (2019)
5. Chen, Y., Zhao, Q., Duan, P., Zhang, B., Hong, Z., Wang, B.: Verifiable privacy-preserving association rule mining using distributed decryption mechanism on the cloud. Expert Syst. Appl. **201**, 117086 (2022)
6. Wang, B., Chen, Y., Li, F., Song, J., Lu, R., Duan, P., Tian, Z.: Privacy-preserving convolutional neural network classification scheme with multiple keys. IEEE Trans. Serv. Comput. (2024)
7. Konečnỳ, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492 (2016)
8. Lu, S., Li, R., Liu, W., Guan, C., Yang, X.: Top-k sparsification with secure aggregation for privacy-preserving federated learning. Comput. Secur. **124**, 102993 (2023)
9. Dasu, V.A., Sarkar, S., Mandal, K.: PROV-FL: Privacy-Preserving Round Optimal Verifiable Federated Learning. In: Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security, pp. 33–44 (2022)
10. Zhou, H., Yang, G., Dai, H., Liu, G.: PFLF: privacy-preserving federated learning framework for edge computing. IEEE Trans. Inf. Forensics Secur. **17**, 1905–1918 (2022). https://doi.org/10.1109/TIFS.2022.3174394
11. Hahn, C., Kim, H., Kim, M., Hur, J.: VerSA: verifiable secure aggregation for cross-device federated learning. IEEE Trans. Dependable Secure Comput. (2021)
12. Wang, Z., Song, M., Zhang, Z., Song, Y., Wang, Q., Qi, H.: Beyond inferring class representatives: User-level privacy leakage from federated learning. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, pp. 2512–2520. IEEE (2019)
13. Mo, F., Haddadi, H., Katevas, K., Marin, E., Perino, D., Kourtellis, N.: PPFL: privacy-preserving federated learning with trusted execution environments. In: Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services, pp. 94–108 (2021)
14. Fang, H., Qian, Q.: Privacy preserving machine learning with homomorphic encryption and federated learning. Future Internet **13**(4), 94 (2021)
15. Ma, J., Naas, S.-A., Sigg, S., Lyu, X.: Privacy-preserving federated learning based on multi-key homomorphic encryption. Int. J. Intell. Syst. **37**(9), 5880–5901 (2022)
16. Zhang, X., Fu, A., Wang, H., Zhou, C., Chen, Z.: A privacy-preserving and verifiable federated learning scheme. In: ICC 2020-2020 IEEE International Conference on Communications (ICC), pp. 1–6. IEEE (2020)
17. Wang, W., Li, X., Qiu, X., Zhang, X., Zhao, J., Brusic, V.: A privacy preserving framework for federated learning in smart healthcare systems. Inf. Process. Manag. **60**(1), 103167 (2023)
18. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1175–1191 (2017)
19. Phong, L.T., Aono, Y., Hayashi, T., Wang, L., Moriai, S.: Privacy-preserving deep learning via additively homomorphic encryption. IEEE Trans. Inf. Forensics Secur. **13**(5), 1333–1345 (2018). https://doi.org/10.1109/TIFS.2017.2787987
20. Jia, B., Zhang, X., Liu, J., Zhang, Y., Huang, K., Liang, Y.: Blockchain-enabled federated learning data protection aggregation scheme with differential privacy and homomorphic encryption in IIoT. IEEE Trans. Ind. Inform. **18**(6), 4049–4058 (2021)
21. Fu, A., Zhang, X., Xiong, N., Gao, Y., Wang, H., Zhang, J.: VFL: a verifiable federated learning with privacy-preserving for big data in industrial IoT. IEEE Trans. Ind. Inform. **18**(5), 3316–3326 (2020)
22. Zhang, Y., Yu, H.: Towards verifiable federated learning. arXiv preprint arXiv:2202.08310 (2022)
23. Gao, H., He, N., Gao, T.: SVeriFL: successive verifiable federated learning with privacy-preserving. Inf. Sci. **622**, 98–114 (2023)
24. Zhang, C., Li, S., Xia, J., Wang, W., Yan, F., Liu, Y.: Batchcrypt: efficient homomorphic encryption for cross-silo federated learning. In: Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC 2020) (2020)
25. Shokri, R., Shmatikov, V.: Privacy-preserving deep learning. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 1310–1321 (2015)
26. Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., Zhou, Y.: A hybrid approach to privacy-preserving federated learning. In: Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, pp. 1–11 (2019)
27. Zhou, Z., Tian, Y., Xiong, J., Ma, J., Peng, C.: Blockchain-enabled secure and trusted federated data sharing in IIoT. IEEE Trans. Ind. Inform. (2022)
28. Hu, R., Guo, Y., Li, H., Pei, Q., Gong, Y.: Personalized federated learning with differential privacy. IEEE Internet Things J. **7**(10), 9530–9539 (2020)
29. Chen, J., Xue, J., Wang, Y., Huang, L., Baker, T., Zhou, Z.: Privacy-preserving and traceable federated learning for data sharing in industrial IoT applications. Expert Syst. Appl. **213**, 119036 (2023)
30. Tang, X., Shen, M., Li, Q., Zhu, L., Xue, T., Qu, Q.: Pile: robust privacy-preserving federated learning via verifiable perturbations. IEEE Trans. Depend. Secure Comput. (2023)
31. Fang, C., Guo, Y., Wang, N., Ju, A.: Highly efficient federated learning with strong privacy preservation in cloud computing. Comput. Secur. **96**, 101889 (2020)
32. Wei, K., Li, J., Ding, M., Ma, C., Yang, H.H., Farokhi, F., Jin, S., Quek, T.Q., Poor, H.V.: Federated learning with differential privacy: algorithms and performance analysis. IEEE Trans. Inf. Forensics Secur. **15**, 3454–3469 (2020)
33. Mugunthan, V., Polychroniadou, A., Byrd, D., Balch, T.H.: SMPAI: secure multi-party computation for federated learning. In: Proceedings of the NeurIPS 2019 Workshop on Robust AI in Financial Services (2019)
34. Zhou, C., Fu, A., Yu, S., Yang, W., Wang, H., Zhang, Y.: Privacy-preserving federated learning in fog computing. IEEE Internet Things J. **7**(11), 10782–10793 (2020)
35. Lin, L., Zhang, X.: PPVerifier: a privacy-preserving and verifiable federated learning method in cloud-edge collaborative computing environment. IEEE Internet Things J. (2022)
36. Zhao, J., Zhu, H., Wang, F., Lu, R., Liu, Z., Li, H.: PVD-FL: a privacy-preserving and verifiable decentralized federated learning framework. IEEE Trans. Inf. Forensics Secur. **17**, 2059–2073 (2022)
37. Wang, Y., Zhang, A., Wu, S., Yu, S.: Vosa: verifiable and oblivious secure aggregation for privacy-preserving federated learning. IEEE Trans. Depend. Secure Comput. (2022)
38. Ren, Y., Li, Y., Feng, G., Zhang, X.: Privacy-enhanced and verification-traceable aggregation for federated learning. IEEE Internet Things J. **9**(24), 24933–24948 (2022)

39. Peng, Z., Xu, J., Chu, X., Gao, S., Yao, Y., Gu, R., Tang, Y.: VFChain: enabling verifiable and auditable federated learning via blockchain systems. IEEE Trans. Netw. Sci. Eng. **9**(1), 173–186 (2021)
40. Guo, X., Liu, Z., Li, J., Gao, J., Hou, B., Dong, C., Baker, T.: VeriFL: communication-efficient and fast verifiable aggregation for federated learning. IEEE Trans. Inf. Forensics Secur. **16**, 1736–1751 (2020)
41. Xu, Y., Peng, C., Tan, W., Tian, Y., Ma, M., Niu, K.: Non-interactive verifiable privacy-preserving federated learning. Future Gener. Comput. Syst. **128**, 365–380 (2022)
42. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
43. Qin, B., Chen, Y., Huang, Q., Liu, X., Zheng, D.: Public-key authenticated encryption with keyword search revisited: security model and constructions. Inf. Sci. **516**, 515–528 (2020)
44. Blake-Wilson, S., Johnson, D., Menezes, A.: Key agreement protocols and their security analysis. Lect. Notes Comput. Sci. **1355**, 30–45 (1997)
45. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
46. Deng, L.: The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Process. Mag. **29**(6), 141–142 (2012). https://doi.org/10.1109/MSP.2012.2211477
47. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282. PMLR (2017)
48. Gardner, M.W., Dorling, S.: Artificial neural networks (the multilayer perceptron)-a review of applications in the atmospheric sciences. Atmos. Environ. **32**(14–15), 2627–2636 (1998)
49. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. **25** (2012)
50. Menegatti, D., Giuseppi, A., Manfredi, S., Pietrabissa, A.: A discrete-time multi-hop consensus protocol for decentralized federated learning. IEEE Access (2023)



Qun Zhou received the B.S. degrees from the college of Computer Science and Technology, Qingdao University, China, in 2021. He is currently pursuing the M.S. degree in Computer Technology of the College of Computer Science and Technology at Qingdao University. His research interests include privacy protection and federated learning.



Wenting Shen received Ph. D. degree in School of Mathematics from Shandong University, in 2020. She is currently an associate professor of the College of Computer Science and Technology at Qingdao University. She has published several research papers in refereed international journals including IEEE Transactions on Information Forensics and Security and IEEE Transactions on Dependable and Secure Computing. Her research interests include cloud computing security, privacy computing and big data security.