Computers & Security 136 (2024) 103562

Contents lists available at ScienceDirect

Computers & Security

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

VPPFL: A verifiable privacy-preserving federated learning scheme against poisoning attacks

Yuxian Huang [a,*], Geng Yang [a], Hao Zhou [a], Hua Dai [a], Dong Yuan [b], Shui Yu [c]

[a] School of Computer Science and Software, Nanjing University of Posts and Telecommunication, Nanjing, 210003, Jiangsu, China
[b] School of Electrical and Information Engineering, University of Sydney, Sydney, 2006, NSW, Australia
[c] School of Computer Science, University of Technology Sydney, Sydney, 2007, NSW, Australia

ARTICLE INFO

Keywords:
Federated learning
Poisoning attacks
Differential privacy
Privacy-preserving
Defense strategy

ABSTRACT

Federated Learning (FL) allows users to train a global model without sharing original data, enabling data to be available and invisible. However, not all users are benign and malicious users can corrupt the global model by uploading poisonous parameters. Compared with other machine learning schemes, two reasons make it easier for poisoning attacks to succeed in FL: 1) Malicious users can directly poison the parameters, which is more efficient than data poisoning; 2) Privacy preserving techniques, such as homomorphic encryption (HE) or differential privacy (DP), give poisonous parameters a cover, which makes it difficult for the server to detect outliers. To solve such a dilemma, in this paper, we propose VPPFL, a verifiable privacy-preserving federated learning scheme (VPPFL) with DP as the underlying technology. The VPPFL can defend against poisoning attacks and protect users' privacy with small computation and communication cost. Specifically, we design a verification mechanism, which can verify parameters that are perturbed by DP Noise, thus finding out poisonous parameters. In addition, we provide comprehensive analysis from the perspectives of security, convergence and complexity. Extensive experiments show that our scheme maintains the detection capability compared to prior works, but it only needs 15%-30% computation cost and 7%-14% communication cost.

1. Introduction

Big data-driven machine learning has been applied in many real-world tasks, including face recognition, tumor prediction and shopping recommendation (Mohr et al., 2021), (LeCun et al., 2015). However, datasets, like isolated islands, are usually distributed among different users and contain private information (Alessandro et al., 2022). Especially for datasets such as medical diagnosis results, the privacy of patients may be leaked with the release of datasets. The increasingly serious problem of privacy leakage has spawned an emerging distributed learning framework — federated learning (FL) (Su et al., 2022), (Lim et al., 2020). In FL, a global model can be constructed by sharing parameters, rather than original data.

Unfortunately, FL has a serious challenge to be solved. FL is hard to defend against poisoning attacks (Tolpegin et al., 2020), (Fung et al., 2018), (Jagielski et al., 2018). In FL, malicious users can upload poisonous parameters to deviate the aggregated model from a correct trend, which makes it perform poorly. Compared with other machine learning schemes, the following two reasons make it easier for poisoning attacks to succeed in FL. First, malicious users can directly poison the parameters, which is more efficient than data poisoning (Liu et al., 2021). Second, with the aim of protecting privacy, users usually adopt homomorphic encryption (HE) (Aono et al., 2017), (Gentry et al., 2013) or differential privacy (DP) (Xu et al., 2017), (Zeng et al., 2022) to encrypt or perturb parameters before uploading them. These privacy protection techniques put a cloak over the parameters, which protects privacy but also hides poisonous parameters (Melis et al., 2019), (Bhagoji et al., 2019). For the server, it is difficult to detect poisonous parameters without lifting up the cloak (Li X. et al., 2023), (Zhang et al., 2020).

Therefore, it is necessary to design a scheme against poisoning attacks in FL. The scheme should detect poisonous parameters under the premise of protecting privacy. Recent research has provided us with several possible solutions. Some researchers utilize the property of HE and perform a series of operations on the encrypted parameters to realize the detection (Miao et al., 2022), (Z. Ma et al., 2022). Nevertheless, HE has brought a lot of computation and communication cost to FL. There is no doubt that the detection further increases the cost. Excessive cost

* Corresponding author.
E-mail address: 2021040402@njupt.edu.cnm (Y. Huang).

https://doi.org/10.1016/j.cose.2023.103562
Received 23 July 2023; Received in revised form 19 October 2023; Accepted 21 October 2023
Available online 27 October 2023

may make these FL schemes unable to scale well in practice. In addition, some researchers try to defend against poisoning attacks under DP protection (Zhou et al., 2022). They divide users into different groups and deploy edge servers to handle each group. Because all the parameters are perturbed by DP noise, edge servers verify them by comparing the models of whether each user participates in the aggregation or not, instead of directly verifying the perturbed parameters. However, the detection capability of their scheme is affected by the number of users in each group. When the number of users is small, the aggregated models are greatly influenced by DP noise, resulting in the edge server easily mistaking benign parameters for poisonous ones. When the number of users is large, poisonous parameters are averaged during the aggregation, making it hard to verify.

It can be seen that existing solutions cannot resist poisoning attacks while providing high security and efficiency. To address these problems, we propose a verifiable privacy-preserving federated learning scheme (VPPFL), which can efficiently defend against poisoning attacks during the federated training process while protecting privacy. Compared with previous works (Miao et al., 2022), (Z. Ma et al., 2022), (Zhou et al., 2022), our scheme has less communication and computation overhead, and the ability of detecting poisoning attacks is not affected by the number of users. The main contributions of this paper are summarized as follows.

- We propose a verifiable privacy-preserving federated learning scheme (VPPFL) with DP as the underlying technology. The VPPFL can defend against poisoning attacks, in addition to preventing the servers or attackers from infringing users' privacy. Moreover, compared with previous works, our scheme has less communication and computation overhead.
- We design a verification mechanism to detect poisonous parameters in FL. In this mechanism, the servers can verify parameters that are perturbed by DP Noise, thus finding out poisonous parameters and malicious users. What's more, the mechanism has better applicability, and its detection ability is not affected by the number of users.
- We provide comprehensive security analysis, convergence analysis and complexity analysis of the scheme. Extensive experiments conducted on MNIST and CIFAR-10 show that our scheme can detect poisoning attacks precisely with little cost compared with prior works.

The rest of paper is organized as follows. Section 2 introduces the related work. In Section 3, we present preliminary knowledge. System model, adversarial model and design goals are defined in Section 4. We describe the VPPFL in details in Section 5. Section 6 discuss the theoretical analysis. Then, we make the performance evaluation in Section 7. We conclude the paper in Section 8.

2. Related work

As our work focus on the privacy protection and poisoning attacks detection in FL, in this section, we briefly discuss the related work in the following two sub-topics.

2.1. Privacy-preserving federated learning (PPFL)

In FL, users do not share original data, but jointly train a global model by exchanging model parameters, thus protecting their privacy. Nevertheless, privacy can still be divulged to some extent by analyzing the model parameters. By utilize cosine similarity and adversarial attack optimization, Jonas et al. (2020) reconstruct high-resolution original images on the basis of gradients. Khosravy et al. (2022) propose a model inversion attack under the semi-white-box condition, which effectively reduces the search space of reconstructed data and improves the success rate of privacy theft. Liu et al. (2022) find that the predictive sensitivity of training data is lower than that of non-training data, and propose a novel membership inference attack. This attack only needs the black-box API of model to judge whether a particular example is used for FL.

To solve this problem, homomorphic encryption (HE) and differential privacy (DP) are introduced in FL. HE allows the server to perform basic operations on the encrypted parameters, and the result of operations on the cipher text is equal to that on the plain text after decryption. Therefore, in FL, users can homomorphically encrypt the parameters and upload them to the server, which can not only protect privacy, but also complete the model aggregation correctly. For example, Zhang et al. (2019) propose a privacy-enhanced federated learning (PEFL) scheme to deal with an untrusted server. They protect the model parameters by using Paillier homomorphic encryption algorithm to encrypt users' local gradients. Mandal and Gong (2019) present PrivFL, a logistic regression framework in a federated setting. The framework achieves privacy protection through an additive homomorphic encryption algorithm. Nevertheless, the computation and communication of cipher text bring huge cost to FL. To reduce the cost, some researchers use DP to protect privacy. Wei et al. (2020) calculate the global sensitivity of model parameters in the uploading and broadcasting phases of FL and add corresponding Gaussian noise to protect these parameters. Jiang et al. (2023) propose a federated edge learning scheme by using hybrid difference privacy and adaptive compression, which can better protect users' privacy from inference attacks. The disadvantage of DP is that the noise added to the parameters is irreversible, which affects the availability of them. However, DP noise follows a certain distribution and can be canceled out to some extent during the aggregation in FL. Especially when the number of users is larger, the distribution of noise is more uniform, and the impact of DP on the model is smaller.

2.2. Federated learning against poisoning attacks

Model poisoning attack is one of the most powerful attacks in federated learning. Malicious users can directly poison the parameters to destroy the aggregated model, which is more efficient than data poisoning attacks.

Model poisoning attacks can be classified into two categories based on the adversary's objectives, namely untargeted attacks and targeted attacks. Untargeted attacks (Fang et al., 2020), (Shejwalkar and Houmansadr, 2021), (Cao and Gong, 2022) aim to disrupt the availability of the aggregated model. For example, Fang et al. (2020) conduct the first comprehensive study on local model poisoning attacks in federated learning. They assume that an attacker had compromised some users and manipulated the local model parameters during the training process, resulting in a significantly testing error rate. Shejwalkar and Houmansadr (2021) present a generic framework for poisoning attacks, which can reduce the predictive accuracy of the aggregated model by 1.5 to 60 times. Cao and Gong (2022) propose a model poisoning attack called MPAF. They assume that an adversary injects malicious clients into the FL system and controls them to send poisonous model updates to the server, resulting in the predictive accuracy less than 0.4. Targeted attacks (Bagdasaryan et al., 2020), (Sun et al., 2022), (H. Li et al., 2023) aim to make incorrect predictions for specific samples while preserving correct predictions for other samples. Bagdasaryan et al. (2020) inject a backdoor into the aggregated model in FL, allowing them to control the model's performance on the selected backdoor task. Sun et al. (2022) propose the Attacking Distance-aware Attack (ADA) to enhance a poisoning attack by finding the optimized target class in the feature space. H. Li et al. (2023) propose 3DFed, an adaptive, extensible, and multi-layered framework to launch covert FL targeted attacks in a black-box setting.

Many solutions have been proposed against model poisoning attacks in FL. Blanchard et al. (2017) propose Krum, a Byzantine-resilient algorithm for distributed learning. In each iteration, the server selects several models that are most likely to be benign for training, based

Table 1
A comparative summary between our scheme and previous schemes.

Schemes	P1	P2	P3	P4
Krum (Blanchard et al., 2017)	Yes	No	Yes	No
FLCert (Cao et al., 2022)	Yes	No	Yes	No
Romoa (Mao et al., 2021)	Yes	No	Yes	No
PBFL (Miao et al., 2022)	Yes	HE	No	No
ShieldFL (Z. Ma et al., 2022)	Yes	HE	No	No
DPBFL (X. Ma et al., 2022)	Yes	DP	No	No
DPFLPA (Zhou et al., 2022)	Yes	DP	Yes	Yes
VPPFL (ours)	Yes	DP	Yes	No

* P1: Whether resisting model poisoning attacks or not. P2: Whether providing privacy protection or not, what privacy protection is provided. P3: Whether achieving lightweight computation and communication cost or not. P4: Whether being affected by the number of users.

on the Euclidean distance between models. Cao et al. (2022) propose FLCert, a FL framework that can tolerate a finite number of malicious users. The key idea is to divide users into different groups, and each group will train a global model. The server takes into account the predictions of these models to produce a final result. Mao et al. (2021) propose a robust model aggregation solution named Romoa, which can deal with targeted and untargeted attacks with a unified approach. Moreover, Romoa achieves more precise attack detection and better fairness for users in FL by constructing a new similarity measurement.

Unfortunately, the above schemes cannot detect parameters that are encrypted or perturbed. Therefore, Miao et al. (2022) propose a Privacy-preserving Byzantine-robust Federated Learning (PBFL) scheme under the blockchain architecture. Specifically, they utilize cosine similarity to detect gradients uploaded by users. In addition, they employ fully homomorphic encryption to achieve secure aggregation. Finally, they use a blockchain system to guarantee the transparency of the process. Z. Ma et al. (2022) propose ShieldFL, a privacy-preserving defense strategy using two-trapdoor homomorphic encryption, which can resist model poisoning attacks without privacy leakage. However, both PBFL and ShieldFL involve a lot of computation and communication in cipher text. X. Ma et al. (2022) propose a differentially private Byzantine-robust federated learning scheme (DPBFL). DPBFL is effective in preventing adversarial attacks launched by the Byzantine participants and achieves differential privacy through a novel aggregation protocol in the shuffle model. Nevertheless, the shuffle model also introduces significant computational and communication overhead to FL, especially for large-scale datasets. Zhou et al. (2022) design a differentially private FL scheme against poisoning attacks based on edge computing. In the work, edge server performs anomaly detection on the parameters uploaded by users in the group. Unfortunately, their anomaly detection method is affected by the number of users in the group. Too many or too few users will reduce the detection ability.

Therefore, it is extremely challenging to defend against poisoning attacks while providing high security and efficiency in FL. To solve this issue, we propose VPPFL in this paper. Table 1 compares our VPPFL scheme with the above-mentioned solutions in terms of various properties. The results show that our scheme not only protect the privacy of users but also resists poisoning attacks with little cost.

3. Preliminaries

In this section, we introduce the knowledge of federated learning and differential privacy.

3.1. Federated learning

A general FL system (Zheng et al., 2022), (Yang et al., 2019) is consist of a central server and n users. Let D_i denote the data set hold by user i, where $i \in \{1,2,3,\cdots,N\}$. In each iteration, $t \in \{1,2,3,\cdots,T\}$, there are the following steps:

- Step 1: The server broadcasts a global model $w^{(t)}$ to the N users.
- Step 2: Each user trains the global model $w^{(t)}$ on its own data set D_i, and sends the updated model $w_i^{(t)}$ to the server.
- Step 3: The server performs secure aggregation over the uploaded parameters, and gets a new global model $w^{(t+1)}$ for the next iteration.

The training process in Step 2 can be formulated as

$$w_i^{(t)} = \arg\min_{w} F_i(w, w^{(t)}, D_i) \tag{1}$$

where $F_i(\cdot)$ is the loss function of user i. The model aggregation in Step 3 is as follows:

$$w^{(t+1)} = \sum_{i=1}^{N} p_i w_i^{(t)} \tag{2}$$

In FL, N users collaboratively train a global model with the help of servers. After several iterations, $w^{(t)}$ can converge to an optimal solution.

3.2. Differential privacy

Dwork et al. (2006) first proposed differential privacy (DP) in 2006, which provides a strong criterion for privacy preservation. The (ϵ,δ)-DP can be defined as follows.

Definition 1 *(Differential Privacy (Wei et al., 2020))*. A randomized mechanism M satisfies (ϵ,δ)-DP if for any two adjacent datasets D and D' and for all outputs O ($O \subseteq Range(M)$),

$$Pr[M(D) \in O] \leq e^{\epsilon} \times Pr[M(D') \in O] + \delta. \tag{3}$$

$\epsilon > 0$ is the privacy budget that controls the strength of privacy guarantee and δ represents the event that the ratio $\frac{Pr[M(D)\in O]}{Pr[M(D'))\in O]}$ cannot be bounded by e^{ϵ}.

For numerical data, we usually use the Gaussian mechanism to realize (ϵ,δ)-DP. First, we need to calculate the global sensitivity (Dwork, 2011).

Definition 2 *(Global Sensitivity)*. For a real-valued function f: $D \to R$ and any two adjacent datasets D and D', the global sensitivity of f is defined as:

$$\Delta f = \max_{D,D'} \| f(D) - f(D') \|. \tag{4}$$

In DP, Δf is understood as the similarity of two adjacent datasets for the function f. Then, we can perturb the data according to the Gaussian mechanism (Wang et al., 2020).

Definition 3 *(Gaussian Mechanism)*. Given a data set D, for a real-valued function f: $D \to R$ with sensitivity Δf, the mechanism M provides (ϵ,δ)-DP satisfying:

$$M(D) = f(D) + N(0,\sigma^2) \tag{5}$$

where $N(0,\sigma^2)$ satisfies a Gaussian distribution with mean 0 and standard deviation $\sigma > \Delta f \cdot \frac{\sqrt{2ln(1.25/\delta)}}{\epsilon}$.

Definition 4 *(Post-processing Properties of Differential Privacy)*. Let $M(D) : D \to R$ satisfy (ϵ,δ)-DP. If $g(R) : R \to R'$ is any random mapping, then $g(M(D)) : D \to R'$ satisfies (ϵ,δ)-DP.

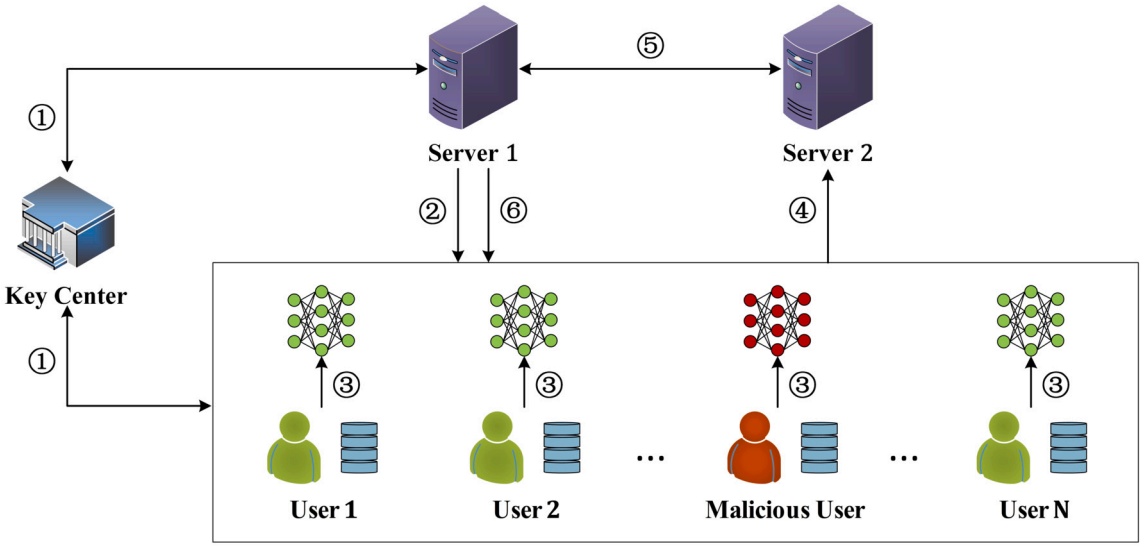


Fig. 1. The system model of VPPFL.

Table 2
Comparison of parameters after adding Gaussian noise.

Number	0	1	2	3	4	5	6	7	8	9	Average
Original Parameter	0.838	-0.964	-0.529	0.829	0.577	-0.551	0.279	0.959	-0.345	-0.202	0.089
Perturbed Parameter	2.211	-1.956	-0.159	0.948	1.65	-0.256	-0.167	-0.174	-1.44	-1.152	0.0495

4. Problem formulation

In this section, we describe the system model, adversarial model and design goals of VPPFL.

4.1. System model

As shown in Fig. 1, the system model consists of a key center, N users and two servers. The specific role of each entity in VPPFL is described below.

- *Key Center*: Key Center is a trusted institution that negotiates key pairs for users and Server 1.
- *Users*: Users are classified into benign users and malicious users. Benign users aim to benefit from FL. Malicious users upload poisonous parameters and attempt to destroy the global model.
- *Servers*: Server 1 is responsible for the publication of Gaussian noise, helping users protect their privacy. Besides, Server 1 and Server 2 cooperate to implement the verification mechanism.

These entities perform the following steps. First, Server 1 and User i negotiate a symmetric key through the Key Center (Step ①). Then, in each iteration, Server 1 generates the Gaussian noise and sends them to users after encryption (Step ②). Next, users perform the local training (Step ③) and upload parameters to Server 2 after adding the Gaussian noise (Step ④). After that, Server 1 and Server 2 cooperate to detect poisonous parameters and realize the secure aggregation (Step ⑤). During this process, the users' privacy is not violated. Finally, Server 1 sends the aggregated model to benign users after encryption (Step ⑥).

4.2. Adversarial model

Key Center is a trusted third party. Server 1 and Server 2 are curious-but-honest servers. They honestly execute the established protocols, but may curiously deduce users' privacy. Moreover, both of the servers are non-colluding, which can be realized in practice (Z. Ma et al., 2022). The potential threats caused by these entities are shown as follows.

1) Privacy Disclosure: In FL, although users' data does not leave the local, privacy leakage may still occur in the transmission of parameters. The attackers can perform model inversion attacks on parameters to reconstruct users' original data, and can also perform membership inference attacks on parameters to judge whether the training data contains a particular sample.

2) Poisoning Attacks: In FL, malicious users can upload random parameters to implement poisoning attacks. When FL is subjected to these attacks, the parameters aggregation can be present as follows:

$$w^{(t+1)} = \sum_{i \in B} p_i w_i^{(t)} + \sum_{j \notin B} p_j w_j^{(t)}, \tag{6}$$

where B is the set of benign users. The purpose of FL is to collaboratively train models, which is also the objective of the majority of users (benign users) in FL. Malicious users, on the other hand, aim to disrupt the global model, and their number should be less than 50%, which is a prerequisite for detection. In addition, the goal of malicious users is to destroy the global model without being detected. Therefore, crafty malicious users will control the size of poisonous parameters in a reasonable range. In addition, some privacy-preserving methods can further obscure these poisonous parameters. Taking DP as an example, in order to protect privacy, the perturbation of DP for a single feature is relatively large. Only because these noises conform to a certain distribution, they can cancel out when aggregated.

Intuitively, Table 2 shows the comparison between the original parameters and the perturbed parameters. We generate 10 random numbers between $[-1, 1]$ as the original parameters of 10 users, and add Gaussian noise G ($G \sim N(0,1), \epsilon = 0.1, \delta = 0.01$) to make it satisfy the DP protection. The results show that the perturbation for individual user is very large, but their aggregated results are approximate. This phenomenon is more obviously when there are more users.

As a result, when users adopt DP to protect their privacy, the uploaded parameters can be described as follows.

$$\overline{w}_i^{(t)} = \begin{cases} w_i^{(t)} + G_i^{(t)}, i \in B \\ \tilde{w}_i^{(t)}, i \notin B \end{cases} \tag{7}$$

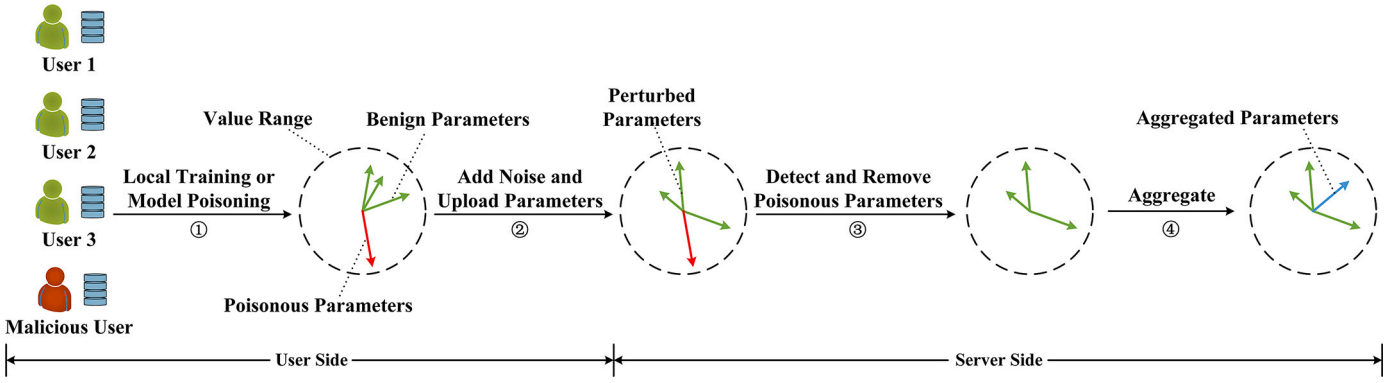


Fig. 2. The technical overview.

Table 3
Table of notations.

Notation	Description
B	Set of benign users
N	Number of users
T	Number of iterations
D_i	Data set of User i
$G_i^{(t)}$	Gaussian noise of User i in t-th iteration
(pk_i, sk_i)	Asymmetric key pair of User i
(pk, sk)	Asymmetric key pair of Server 1
k_i	Symmetric key of User i
ϵ	Privacy budget
δ	Privacy parameter
Δf	Global sensitivity of parameters
μ	Mean of Gaussian noise
σ	Variance of Gaussian noise
$w_i^{(t)}$	Local parameters trained by benign User i in t-th iteration
$\tilde{w}_i^{(t)}$	Local parameters generated by malicious user i in t-th iteration
$\overline{w}_i^{(t)}$	Local parameters uploaded by User i in t-th iteration
$W^{(t)}$	Matrix of parameters uploaded by users in t-th iteration
$V^{(t)}$	Difference matrix of noise in t-th iteration
$d^{(t)}$	Distance matrix in t-th iteration
$w_i^{*(t)}$	The parameters of User i after being processed by Server 2 in t-th iteration.
$w^{(t)}$	Global parameters in t-th iteration
C	Clipping threshold for bounding $w_i^{(t)}$

where $w_i^{(t)}$ and $G_i^{(t)}$ are the parameters and noise of benign user i, $\tilde{w}_i^{(t)}$ is the poisonous parameters of malicious user i. Because of the existing of DP noise, server cannot judge whether $\overline{w}_i^{(t)}$ $(i = 1, 2, \cdots, N)$ are poisoned or not by directly comparing their size or direction, which makes the poisoning attacks more covert.

4.3. Design goals

Under the adversarial model, we aim to propose a FL scheme that can protect privacy, verify poisonous parameters and reduce cost. Specifically, we are committed to realizing the following goals.

1) Privacy Preservation. We need to protect the users' privacy from being compromised. The transmission of all parameters in VPPFL should satisfy differential privacy protection. Neither servers nor attackers can infer users' original data from analyzing parameters.

2) Verifiability. We need to resist poisoning attacks while protecting privacy. The servers can verify the parameters uploaded by the users, and detect poisonous ones.

3) Efficiency. Compared with the previous works, we need to reduce the computation and communication cost. Besides, the detection capability is not affected by the number of users in our scheme.

5. Design of VPPFL

In this section, we introduce the technical overview, and then describe the VPPFL in detail. The notations are shown in Table 3.

5.1. Technical overview

In FL, parameters can be represented as multi-dimensional vectors. The existing works usually detect poisonous parameters by calculating the difference between these vectors. Because the outliers that are quite different from other vectors have a high possibility of poisoning. For example, the work (Zhou et al., 2022) measures the difference by computing the L_2 norm of distance between different parameters, which is shown in Equation (8).

$$d_{ij}^{(t)} = \| w_i^{(t)} - w_j^{(t)} \|_2 \tag{8}$$

However, the above method can not be applied to privacy-preserving federated learning directly. When we use DP to perturb users' parameters, the equation of distance becomes as follows.

$$\begin{aligned} d_{ij}^{(t)} &= \| (w_i^{(t)} + G_i^{(t)}) - (w_j^{(t)} + G_j^{(t)}) \|_2 \\ &= \| (w_i^{(t)} - w_j^{(t)}) + (G_i^{(t)} - G_j^{(t)}) \|_2 \end{aligned} \tag{9}$$

According to Table 2, we know that the perturbation to a single user's parameters is very large under the protection of DP. Therefore, $(G_i^{(t)} - G_j^{(t)})$ may have a great effect on the distance. To solve this issue, we eliminate the effect of noise in our verification mechanism.

Fig. 2 shows the overview of VPPFL. First, benign users perform the local training, while malicious users generate poisonous parameters (Step ①). Then, benign users add noise to parameters for DP protection and malicious users directly upload poisonous parameters (Step ②). The size of both perturbed and poisonous parameters are limited in a value range, which makes the attack more covert. After servers receive these parameters, they detect and remove the poisonous parameters based on a verification mechanism (Step ③). Moreover, servers cannot get or infer users' original parameters during the entire process. Finally, servers aggregate benign parameters and send the result to users (Step ④). These steps will iterate until the global model converges.

5.2. Construction of VPPFL

VPPFL consists of four parts, namely privacy-preserving mechanism, local training, verification mechanism, and model aggregation.

1) Privacy-Preserving Mechanism: In this mechanism, Server 1 and users first negotiate a set of symmetric keys through the Key Center. After that, in each iteration, Server 1 generates N noise vectors according to Gaussian mechanism and sends them to each user after encryption. The details are shown in Algorithm 1.

a) Negotiation of Symmetric Key: Users and Server 1 first send their public keys to the Key Center, while the Key Center generates N symmetric keys. Then, the Key Center uses public keys to encrypt symmetric keys and sends them back. Finally, Server 1 and each user use their own secret keys to decrypt and obtain the symmetric key k_i.

After this negotiation, each user and Server 1 get a set of symmetric keys. They can be used to encrypt the Gaussian noise and aggregated

Algorithm 1: Privacy-Preserving Mechanism

Input: Users' key pair (pk_i, sk_i), Server 1's key pair (pk, sk), Privacy budget ϵ, Privacy parameter δ, Global sensitivity Δf, The mean of Gaussian noise μ.
Output: Symmetric key k_i, Gaussian noise $G_i^{(t)}$.

```
/* Negotiation of symmetric keys */
Users and Server 1 send pk_i, pk to the Key Center;
The Key Center generates symmetric key k_i;
for i = 1 to N do
    The Key Center encrypts k_i with pk_i, and sends it to the user i;
    User i decrypts the ciphertext and gets k_i;
end
The Key Center encrypts k_i with pk, and sends them to the Server 1;
/* Publication of Gaussian Noise */
for t = 1 to T do
    Based on parameter μ, ϵ, δ, Δf, Server 1 generates a set of Gaussian noise G_i^(t);
    for i = 1 to N do
        Server 1 encrypts G_i^(t) with k_i, and sends it to User i;
        User i decrypts the ciphertext and gets G_i^(t);
    end
end
```

Algorithm 2: Local Training

Input: Encrypted global parameter $[w^{(t)}]$, Loacl training data D_i, Symmetric key k_i, Gaussian noise $G_i^{(t)}$.
Output: Local parameter $\overline{w}_i^{(t)}$.

```
for i = 1 to N do
    /*Benign Users*/
    if user i is benign then
        User i uses k_i to encrypts [w^(t)], and gets the global parameter w^(t);
        User i trains w^(t) on local training data D_i, and obtains updated local parameter w_i^(t);
        User i perturbs w_i^(t) with Gaussian noise G_i^(t), and uploads perturbed local parameter w̄_i^(t) to Server 2;
    end
    /*Malicious Users*/
    else
        User i launches model poisoning by uploading poisonous parameters w̄_i^(t) to Server 2.
    end
end
```

model sent by Server 1. Besides, compared with asymmetric keys, symmetric keys are more efficient, which can further reduce the cost.

b) Publication of Gaussian noise: Server 1 generates a set of Gaussian noise. The variance is σ, which is calculated based on the parameters ϵ, δ and Δf. The process of calculation is shown in Section 3.2. The mean of noise is μ instead of 0, which is to protect users' privacy when Server 1 and Server 2 cooperate to detect poisonous parameters. Then, Server 1 will use symmetric key k_i to encrypt noise $G_i^{(t)}$ and send it to user i. Finally, each user can get the noise $G_i^{(t)}$ after decryption.

2) Local Training: In each iteration, benign users locally train the model using SGD, and upload perturbed parameters to Server 2. Malicious users launch model poisoning by uploading poisonous parameters to Server 2. We show the details in Algorithm 2.

3) Verification Mechanism: In this mechanism, Server 1 and 2 will work together to verify parameters. At first, Server 1 calculates the difference between any Gaussian noise $G_i^{(t)}$ and the rest of Gaussian noise. For convenience of notation, we set i equal to 1 here. The calculation can be formulated as follows.

$$V_j^{(t)} = G_1^{(t)} - G_j^{(t)}, j = 1, 2, \cdots, N \tag{10}$$

Then, the difference matrix $V^{(t)}$ can be represented as Equation (11).

$$\begin{aligned} V^{(t)} &= [V_1^{(t)}, V_2^{(t)}, \cdots, V_N^{(t)}]^T \\ &= \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 1 & -1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \cdots & -1 \end{bmatrix} \cdot \begin{bmatrix} G_1^{(t)} \\ G_2^{(t)} \\ \vdots \\ G_N^{(t)} \end{bmatrix} \\ &= A^{(t)} \cdot G^{(t)} \end{aligned} \tag{11}$$

where $A^{(t)}$ is an $N \times N$ coefficient matrix and $G^{(t)}$ is a matrix composed of Gaussian noise sent to users. Obviously, $A^{(t)}$ is not full rank, so Server 2 cannot infer the Gaussian noise $G^{(t)}$ from the above equation. Moreover, since Server 2 does not know noise's mean μ, it cannot obtain additional information by calculating the sum of Gaussian noise. More details will be explained in Section 6.1.

After receiving the matrix V, Server 2 is able to eliminate the influence of Gaussian noise on benign parameters, which is shown in the following equation.

$$\begin{aligned} W^{(t)} + V^{(t)} &= [\overline{w}_1^{(t)}, \overline{w}_2^{(t)}, \cdots, \overline{w}_N^{(t)}]^T + [V_1^{(t)}, V_2^{(t)}, \cdots, V_N^{(t)}]^T \\ &= [\overline{w}_{i\in B}^{(t)} + V_{i\in B}^{(t)}, \overline{w}_{j\notin B}^{(t)} + V_{j\notin B}^{(t)}]^T \\ &= \begin{bmatrix} w_{i\in B}^{(t)} + G_{i\in B}^{(t)} + G_1^{(t)} - G_{i\in B}^{(t)} \\ \tilde{w}_{j\notin B}^{(t)} + G_1^{(t)} - G_{j\notin B}^{(t)} \end{bmatrix} \\ &= \begin{bmatrix} w_{i\in B}^{(t)} + G_1^{(t)} \\ \tilde{w}_{j\notin B}^{(t)} + G_1^{(t)} - G_{j\notin B}^{(t)} \end{bmatrix} \\ &= [w_1^{*(t)}, w_2^{*(t)}, \cdots, w_N^{*(t)}]^T \end{aligned} \tag{12}$$

According to Equation (12), we can obtain the expression for $w_i^{*(t)}$.

$$w_i^{*(t)} = \begin{cases} w_i^{(t)} + G_1^{(t)}, i \in B \\ \tilde{w}_i^{(t)} + G_1^{(t)} - G_i^{(t)}, i \notin B \end{cases} \tag{13}$$

Then, Server 2 can calculate the distance between User i's parameters and those of other users. If user i is benign ($i \in B$), the process of calculation is shown as follows.

$$\begin{aligned} d_{i\in B}^{(t)} &= \sum_{j=1}^{N} \| w_i^{*(t)} - w_j^{*(t)} \| \\ &= \sum_{j\in B} \| (w_i^{(t)} + G_1^{(t)}) - (w_j^{(t)} + G_1^{(t)}) \| \\ &+ \sum_{j\notin B} \| (w_i^{(t)} + G_1^{(t)}) - (\tilde{w}_j^{(t)} + G_1^{(t)} - G_j^{(t)}) \| \\ &= \sum_{j\in B} \| w_i^{(t)} - w_j^{(t)} \| + \sum_{j\notin B} \| w_i^{(t)} - \tilde{w}_j^{(t)} + G_j^{(t)} \| \end{aligned} \tag{14}$$

If User i is malicious ($i \notin B$), the process of calculation is shown as follows.

$$\begin{aligned} d_{i\notin B}^{(t)} &= \sum_{j=1}^{N} \| w_i^{*(t)} - w_j^{*(t)} \| \\ &= \sum_{j\in B} \| (\tilde{w}_i^{(t)} + G_1^{(t)} - G_i^{(t)}) - (w_j^{(t)} + G_1^{(t)}) \| \\ &+ \sum_{j\notin B} \| (\tilde{w}_i^{(t)} + G_1^{(t)} - G_i^{(t)}) - (\tilde{w}_j^{(t)} + G_1^{(t)} - G_j^{(t)}) \| \\ &= \sum_{j\in B} \| \tilde{w}_i^{(t)} - w_j^{(t)} - G_i^{(t)} \| \\ &+ \sum_{j\notin B} \| \tilde{w}_i^{(t)} - \tilde{w}_j^{(t)} - G_i^{(t)} + G_j^{(t)} \| \end{aligned} \tag{15}$$

Then, Server 2 gets the matrix $d^{(t)} = [d_1^{(t)}, d_2^{(t)}, \cdots, d_N^{(t)}]$. For two benign users, the model parameters uploaded by them are similar. Furthermore, in Section 4.2, we assume that the number of malicious users is smaller than that of benign users. Therefore, we can get the following formula.

Algorithm 3: Verification Mechanism

Input: Local parameter $\overline{w}_i^{(t)}$, Gaussian noise $G_i^{(t)}$.
Output: Identity of the malicious users.

1 Server 1 computes the difference matrix $V^{(t)}$ and sends it to Server 2;
2 Server 2 adds $V^{(t)}$ to $\overline{w}_i^{(t)}$, thus eliminating the effect of Gaussian noise;
3 Server 2 computes the distance matrix $d^{(t)}$;
4 Server 2 uses the clustering algorithm to divide the elements in $d^{(t)}$ into two categories, and the smaller category corresponds to malicious users.

$$\begin{cases} \| w_i^{(t)} - w_j^{(t)} \| \approx 0 \\ \sum_{j \notin B} \| w_i^{(t)} - \tilde{w}_j^{(t)} + G_j^{(t)} \| < \sum_{j \in B} \| \tilde{w}_i^{(t)} - w_j^{(t)} - G_i^{(t)} \| \end{cases} \tag{16}$$

We subtract $d_{i \in B}^{(t)}$ from $d_{i \notin B}^{(t)}$, and get the difference between them

$$\begin{aligned} d_{i \in B}^{(t)} - d_{i \notin B}^{(t)} &\approx 0 - \sum_{j \notin B} \| \tilde{w}_i^{(t)} - \tilde{w}_j^{(t)} - G_i^{(t)} + G_j^{(t)} \| \\ &\quad + (\sum_{j \notin B} \| w_i^{(t)} - \tilde{w}_j^{(t)} + G_j^{(t)} \| \\ &\quad - \sum_{j \in B} \| \tilde{w}_i^{(t)} - w_j^{(t)} - G_i^{(t)} \|) \\ &< - \sum_{j \notin B} \| \tilde{w}_i^{(t)} - \tilde{w}_j^{(t)} - G_i^{(t)} + G_j^{(t)} \| \\ &\ll 0 \end{aligned} \tag{17}$$

Therefore, $d_{i \in B}^{(t)}$ is far less than $d_{i \notin B}^{(t)}$. As a result, Server 2 can use the clustering algorithm to divide the elements in $d^{(t)}$ into two categories, and the smaller category corresponds to malicious users. In this way, Server 2 completes the detection. Algorithm 3 summarizes the verification mechanism in VPPFL.

4) Model Aggregation: In this process, Server 1 and 2 will aggregate these benign parameters and send back to benign users after encryption. The details are shown in Algorithm 4. Firstly, Server 2 computes the sum of parameters uploaded by benign users. The calculation is shown as follows.

$$\begin{aligned} \overline{w}^{(t+1)} &= \frac{1}{|B|} \sum_{i \in B} \overline{w}_i^{(t)} \\ &= \frac{1}{|B|} \sum_{i \in B} (w_i^{(t)} + G_i^{(t)}) \\ &\approx \frac{1}{|B|} \sum_{i \in B} w_i^{(t)} + \mu \end{aligned} \tag{18}$$

Because the mean of Gaussian noise $G_i^{(t)}$ is μ. $\sum_{i \in B} G_i^{(t)}$ tends to μ when enough benign users participate. Server 2 is unaware of μ, but Server 1 is aware of it. Therefore, Server 2 sends $\overline{w}^{(t+1)}$ to Server 1, and Server 1 can get the aggregated parameters after subtracting μ.

$$\begin{aligned} w^{(t+1)} &= \overline{w}^{(t+1)} - \mu \\ &\approx \frac{1}{|B|} \sum_{i \in B} w_i^{(t)} \end{aligned} \tag{19}$$

Finally, in order to prevent $w^{(t+1)}$ from leaking privacy, Server 1 encrypts it with symmetric key k_i and then sends it to each user.

Algorithm 4: Model Aggregation

Input: Local parameter $\overline{w}_i^{(t)}$, The mean of Gaussian noise μ.
Output: Aggregated parameter $w^{(t+1)}$.

1 Server 2 computes the sum $\overline{w}^{(t+1)}$ of parameters uploaded by benign users and send it to Server 1;
2 Server 1 subtracts μ from $\overline{w}^{(t+1)}$ to obtain the aggregated parameter $w^{(t+1)}$;
3 Server 1 uses k_i to encrypt $w^{(t+1)}$ and sends it to each user.

6. System analysis

In this section, we analyze VPPFL from the perspectives of security, convergence and complexity.

6.1. Security analysis

In FL, attackers can infer users' local data by analyzing parameters. To prevent privacy leakage, we must protect them. According to Section 5, we can see that there are four parameter transfers in VPPFL. Among them, the Gaussian noise and aggregated parameters sent by Server 1 to users are encrypted by symmetric keys, so there is no privacy leakage in the two processes. Then, we need to analyze whether there are privacy leakage in the remaining two processes (the uploading of local parameters and the interactions between servers).

1) The Uploading of Local Parameters: In the process of local training, users use Gaussian noise to perturb the model parameters before uploading them, so that they can meet the requirements of DP.

Theorem 1. *In VPPFL, the parameter $\overline{w}_i^{(t)}$ uploaded by user i satisfies (ϵ, δ)-DP.*

Proof. According to Definition 2, VPPFL requires to compute the global sensitivity Δf before determining the scale of Gaussian noise. Δf can be expressed as follows.

$$\begin{aligned} \Delta f &= \max_{D,D'} \| f(D_i) - f(D_i') \| \\ &= \max_{D,D'} \| w_i^{(t)} - w_i^{(t)\prime} \| \\ &= \max_{D,D'} \| \frac{1}{|D_i|} \sum_{j=1}^{|D_i|} \arg\min_w F_i(w, D_{i,j}) \\ &\quad - \frac{1}{|D_i'|} \sum_{j=1}^{|D_i'|} \arg\min_w F_i(w, D_{i,j}') \| \\ &= \frac{2C}{|D_i|}. \end{aligned} \tag{20}$$

D_i' is an adjacent data set to D_i. They have the same size but only differ by one sample. $D_{i,j}$ is the j-th sample in D_i. C is a clipping threshold for bounding w_i. We assume that all users have the same size of data for local training and define the size by m. Then, we obtain $\Delta f = \frac{2C}{m}$.

To ensure (ϵ, δ)-DP, we set the noise scale as $\sigma = \frac{c \Delta f}{\epsilon}$ $(c > \sqrt{2ln(1.25/\delta)})$, which is represented by the standard deviation of the additive Gaussian noise. Based on Definition 3, we can prove that $M(D)$ satisfies (ϵ, δ)-DP, where $M(D)$ is shown below.

$$\begin{aligned} M(D_i) &= f(D_i) + N(0, \sigma^2) \\ &= w_i^{(t)} + N(0, \sigma^2) \end{aligned} \tag{21}$$

However, in VPPFL, the distribution of Gaussian noise added by user i is $N(\mu, \sigma^2)$, where the mean of noise is μ. Therefore, $\overline{w}_i^{(t)}$ can be expressed as follows.

$$\begin{aligned} \overline{w}_i^{(t)} &= w_i^{(t)} + N(\mu, \sigma^2) \\ &= w_i^{(t)} + N(0, \sigma^2) + \mu \\ &= M(D_i) + \mu \end{aligned} \tag{22}$$

According to Definition 4, VPPFL satisfies (ϵ, δ)-DP for local parameter $\overline{w}_i^{(t)}$ of user i in t-th iteration. Theorem 1 is proved. □

2) The Interactions between Servers: In the process of verification mechanism and model aggregation, Server 1 and Server 2 exchange information to detect poisoned parameters and complete model aggregation. We need to prove that these interactions do not help servers to violate users' privacy.

Theorem 2. *In VPPFL, both Server 1 and Server 2 are unable to infer user i's original parameter $w_i^{(t)}$ by interacting information.*

Proof. According to Section 5.2, Server 1 only has the aggregated parameter $\overline{w}^{(t+1)}$, so it cannot infer the users' original parameters. Server 2 has the perturbed parameters $\overline{w}_i^{(t)}$ uploaded by user i and difference matrix $V^{(t)}$ sent by Server 1. We use Equation (23) to represent the information owned by Server 2.

$$\begin{cases} A^{(t)} \cdot G^{(t)} = V^{(t)} \\ w_i^{(t)} + G_i^{(t)} = \overline{w}_i^{(t)}, i \in B \end{cases} \tag{23}$$

Based on Equation (11), we know that the rank of matrix A_i is $r = N-1$, but there are N variables in the vector G_i. As a result, $A^{(t)} \cdot G^{(t)} = V^{(t)}$ has infinitely many solutions. Moreover, because Server 1 sets the mean of noise to μ instead of 0, it is not useful for Server 2 to add these noise without knowing μ. Therefore, Server 2 cannot figure out the noise $G_i^{(t)}$, let alone users' original parameter $w_i^{(t)}$.

To sum up, neither Server 1 nor Server 2 can infer users' original parameters from the interaction. Theorem 2 is proved. □

6.2. Convergence analysis

Unlike traditional machine learning, FL trains the global model by aggregating parameters from users, which makes the convergence of the model difficult to guarantee. Therefore, it is necessary to analyze the convergence of FL schemes. The work (X. Li et al., 2020) has proved that a common FL scheme is convergent. In VPPFL, because we add Gaussian noise to protect parameters, the convergence range of global model is different from that of a common FL scheme. Thus, we compute $E\{F(w^{(t)}) - F(\hat{w})\}$ to establish the convergence range of VPPFL. At first, we make the following assumptions on the loss function.

Assumption 1. *We assume that the global loss function $F(\cdot)$ satisfies the following conditions.*
1) $F(w) = \frac{1}{|B|}\sum_{i\in B} F_i(w)$, where $F_i(\cdot)$ is the local loss function of user i.
2) $F(\cdot)$ and $F_i(\cdot)$ are convex.
3) $\hat{w}$ is the optimal result.
4) $F_i(\cdot)$ is β-Lipschitz, i.e., $\| F_i(w) - F_i(w') \| < \beta \| w - w' \|$, for any w and w'.

Based on Assumption 1, we achieve the convergence upper bound and lower bound of the VPPFL in Theorem 3 and 4 respectively.

Theorem 3. *The convergence upper bound of VPPFL can be formulated as*

$$E\{F(w^{(t)}) - F(\hat{w})\} \le 2\beta C + \beta\mu[2\phi(\frac{\mu}{\sigma}) - 1] + \frac{2\beta\sigma}{\sqrt{2\pi}} e^{-\mu^2/2\sigma^2},$$

where $\phi(\cdot)$ is the distribution function of the standard normal distribution.

Proof. In VPPFL, Servers perform aggregation on the perturbed parameters of benign users. According to Assumption 1, we have

$$\begin{aligned} E\{F(w^{(t)}) - F(\hat{w})\} &= E\{\frac{1}{|B|}\sum_{i\in B}[F_i(w^{(t)}) - F_i(\hat{w})]\} \\ &\le E\{\beta \| w^{(t)} - \hat{w} \|\} \\ &\le \beta E\{\| w^{(t)} \| + \| \hat{w} \|\} \\ &= \beta E\{\| \frac{1}{|B|}\sum_{i\in B}(w_i^{(t)} + G_i^{(t)}) \| + \| \hat{w} \|\} \\ &\le \beta E\{\frac{1}{|B|}\sum_{i\in B} \| w_i^{(t)} \| + \frac{1}{|B|}\sum_{i\in B} \| G_i^{(t)} \| \\ &\quad + \| \hat{w} \|\}. \end{aligned} \tag{24}$$

Table 4
Comparison of communication cost.

	UC	SC
PBFL (Miao et al., 2022)	$O(M\hat{X})$	$O((6N+2)M\hat{X})$
ShieldFL (Z. Ma et al., 2022)	$O(M\hat{X})$	$O((8N+1)M\hat{X})$
DPFLPA (Zhou et al., 2022)	$O(MX)$	$O(MX)$
VPPFL (ours)	$O(MX)$	$O(NM\hat{X} + (N+1)MX)$

* UC and SC denote the communication cost of user and server in each iteration, respectively.

Since $G_i^{(t)} \sim N(\mu, \sigma^2)$, $E\{\| G_i^{(t)} \|\}$ can be expressed as

$$\begin{aligned} E\{\| G_i^{(t)} \|\} &= \int_{-\infty}^{+\infty} |x| \cdot \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2} dx \\ &= \int_{-\infty}^{+\infty} |\mu + \sigma x| \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx \\ &= \mu\Big(\int_{-\frac{\mu}{\sigma}}^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx - \int_{-\infty}^{-\frac{\mu}{\sigma}} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx\Big) \\ &\quad + \frac{\sigma}{\sqrt{2\pi}}\Big(\int_{-\frac{\mu}{\sigma}}^{+\infty} x e^{-x^2/2} dx - \int_{-\infty}^{-\frac{\mu}{\sigma}} x e^{-x^2/2} dx\Big) \\ &= \mu[2\phi(\frac{\mu}{\sigma}) - 1] + \frac{2\sigma}{\sqrt{2\pi}} e^{-\mu^2/2\sigma^2}. \end{aligned} \tag{25}$$

Because $\| w_i^{(t)} \|, \| \hat{w} \| \le C$, the Equation (20) can be reformulated as

$$\begin{aligned} E\{F(w^{(t)}) - F(\hat{w})\} &\le \beta(E\{\| w_i^{(t)} \|\} + E\{\| G_i^{(t)} \|\} \\ &\quad + E\{\| \hat{w} \|\}) \\ &\le 2\beta C + \beta\mu[2\phi(\frac{\mu}{\sigma}) - 1] \\ &\quad + \frac{2\beta\sigma}{\sqrt{2\pi}} e^{-\mu^2/2\sigma^2}. \end{aligned} \tag{26}$$

Theorem 3 is proved. □

Theorem 4. *The convergence lower bound of VPPFL can be formulated as*

$$E\{F(w^{(t)}) - F(\hat{w})\} \ge 0.$$

Proof. Because $\hat{w}$ is the optimal result, for $\forall F_i(.)$, $F_i(w_i^{(t)}) \ge F_i(\hat{w})$. Therefore, we have

$$\begin{aligned} E\{F(w^{(t)}) - F(\hat{w})\} &= E\{\frac{1}{|B|}\sum_{i\in B}[F_i(w^{(t)}) - F_i(\hat{w})]\} \\ &\ge 0. \end{aligned} \tag{27}$$

□

6.3. Complexity analysis

To evaluate the efficiency of VPPFL, we compare its computation and communication cost with that of other recent algorithms.

1) Communication Cost: We assume that N is the number of users, and M is the dimensions of local parameters. Then, let X be the communication complexity of a number, and $\hat{X}$ be the communication complexity of an encrypted number, where $X \ll \hat{X}$. The comparison results of communication cost are shown in Table 4.

The work (Miao et al., 2022) and (Z. Ma et al., 2022) adopt the homomorphic encryption to protect the parameters, and the transmission of cipher text greatly increases the communication cost. The cost required for each user to send the encrypted parameters is $O(M\hat{X})$. In the verification phase, servers in work (Miao et al., 2022) and (Z. Ma et al.,

Table 5
Comparison of computation cost.

	UC	SC
PBFL (Miao et al., 2022)	$O(T_1 + MT_2)$	$O(9NMT_2)$
ShieldFL (Z. Ma et al., 2022)	$O(T_1 + MT_2)$	$O(6NMT_2)$
DPFLPA (Zhou et al., 2022)	$O(T_1 + MT_3)$	$O(N(N-1)MT_3)$
VPPFL (ours)	$O(T_1 + MT_3)$	$O(\frac{1}{2}N(N+1)MT_3)$

* UC and SC denote the computation cost of user and server in each iteration, respectively.

Table 6
Descriptions of datasets.

Dataset	Type	Training Data	Test Data	Pixel	Class
MNIST	Grey	60000	10000	28*28	10
CIFAR-10	RGB	50000	10000	32*32	10

2022) need to do normalization and direction judgment, which sum up to $O((6N+2)M\hat{X})$ and $O((8N+1)M\hat{X})$, respectively.

The work (Zhou et al., 2022) and our scheme VPPFL use differential privacy to protect parameters, which does not introduce additional communication cost. Therefore, the cost required for each user is $O(MX)$. In VPPFL, it incurs an additional $O(NM\hat{X} + NMX)$ cost for servers since the noise is sent by users after encryption and there are interaction between servers. However, our scheme can be applied to a large number of users, and work (Zhou et al., 2022) can only accomplish detection when the number of users is small. The details will be show in Section 7.2.

2) Computation Cost: Let T_1 be the computation complexity of local training, T_2 be the complexity of computing an encrypted number, and T_3 be the complexity of computing an plain number, where $T_2 \gg T_3$. The comparison results of computation cost are shown in Table 5. For users, the computation cost is composed of two parts, which are T_1 for local training and MT_2 or MT_3 for encryption or perturbation. For servers, the work (Miao et al., 2022) spends $2NMT_2$ for normalization judgment and $7NMT_2$ for direction judgment. The work (Z. Ma et al., 2022) spends $3NMT_2$ for normalization judgment and $3NMT_2$ for direction judgment. In work (Zhou et al., 2022), the server needs to compute the aggregated parameters with and without each local parameter. Therefore, the computation cost of server in work (Zhou et al., 2022) is $O(N(N-1)MT_3)$. In our scheme, the server requires NMT_3 computation cost to eliminate the impact of differential privacy and $\frac{1}{2}N(N-1)MT_3$ computation cost to compute the distances.

Because $T_2 \gg T_3$, the computation cost of work (Zhou et al., 2022) and our scheme is much less than work (Miao et al., 2022) and (Z. Ma et al., 2022). Moreover, the computation cost of our scheme is the least.

7. Performance evaluation

In this section, we evaluate the performance of VPPFL by comparing it with state of the art algorithms on different datasets.

7.1. Experimental setup

1) Datasets: We adopt two widely used datasets, MNIST and CIFAR-10. The details about datasets are shown in Table 6. MNIST contains 10 classes of handwritten digits, providing 60,000 training samples and 10,000 test samples. Each sample is a gray-scale image of 28*28 pixels. CIFAR-10 is a universal object recognition dataset containing 10 classes of 3-channel color RGB images. The size of the images is 32*32, and there are 50,000 training images and 10,000 test images in the dataset.

Furthermore, we consider both independent and non-independent identically distributed settings (IID and non-IID). In the IID settings, the training data is uniformly distributed, and each user has IID dataset. In the non-IID settings, the training data are divided by class and each user stores samples belonging to a single class.

2) FL Settings: For local model architecture, we consider ResNet18, a popular neural network architecture. For local model training, we adopt SGD algorithm to update parameters. We set the learning rate lr as 0.001, momentum m as 0.0001, and weight decay λ as 0.1. In addition, we set iteration t as 200, local epochs as 1, and batch size as 32.

3) Model Poisoning Attacks: To evaluate VPPFL against model poisoning, we construct the following two standard model poisoning attacks, namely, untargeted attacks and targeted attacks.

- *Untargeted Attacks*: The untargeted attacks aim to disrupt the availability of the aggregated model. In our experiment, malicious users will upload random vectors as local parameters to server 2, poisoning the aggregated model. In order to make the attack difficult to detect, the absolute values of features in the random parameters are all smaller than C, which is the clipping threshold of parameters.
- *Targeted Attacks*: The targeted attacks aim to make incorrect predictions for specific samples while preserving correct predictions for other samples. In our experiments, we changed the labels from '1' to '9' in the training datasets of malicious users to train poisonous parameters. Then, malicious users will send these poisonous parameters to server 2, poisoning the aggregated model.

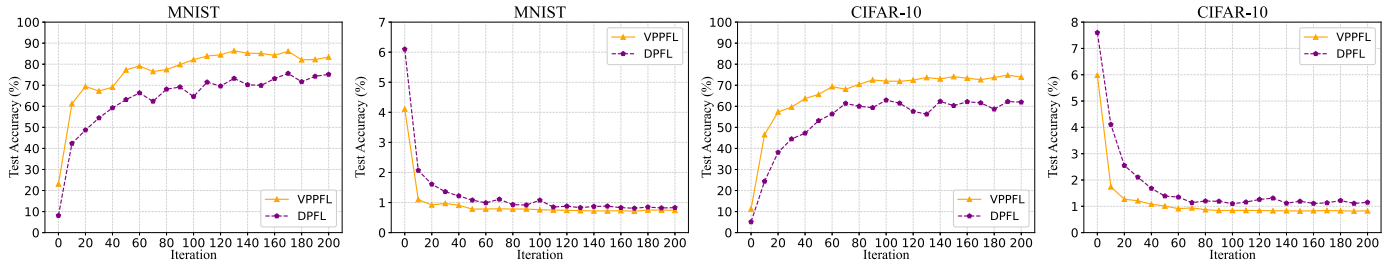
For the proportion of malicious users, we consider different scenarios ranging from 10% to 30%.

4) Experimental Environment: All the experiments were performed on a machine with an Intel(R) Core(TM) i9-10850K CPU running at 3.60GHz processor and a NVIDIA 3070 GPU.

7.2. The performance of VPPFL

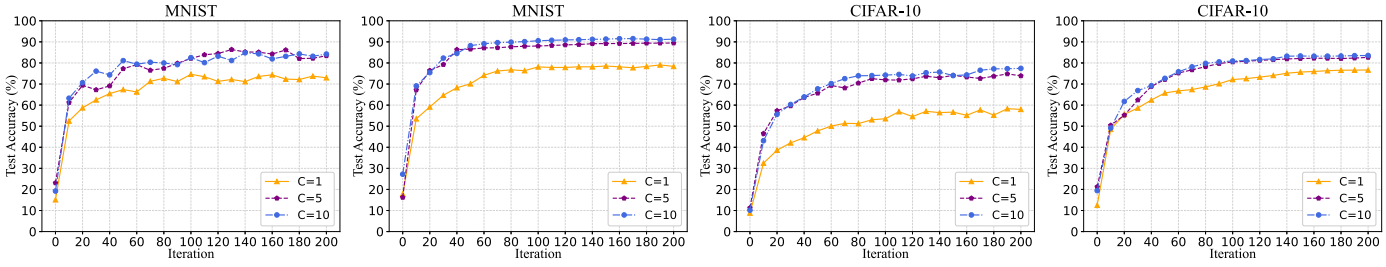
First, we compare VPPFL with DPFL to evaluate the performance of VPPFL. DPFL (Wei et al., 2020) is a common privacy-preserving federated learning algorithm, where both uplink and downlink channels are protected by DP. We assume that there are 10 benign users participating in federated learning and each user has 3000 samples. For simulation parameters, we set $\epsilon = 1$, $C = 5$, $\delta = 0.01$ and $\mu = 2$. The experiments are performed on both MNIST and CIFAR-10 datasets, and the results are shown in Fig. 3. As the number of iterations increase, the model will eventually converge. However, there exist subtle fluctuations due to the influence of noise. In addition, VPPFL performs better than DPFL both in terms of test accuracy and loss function. This is because in VPPFL we only add noise to the uplink channel and the downlink channel is protected by encryption. The noise added by different users in the uplink channel can be offset to some extent, while the noise added by the server in the downlink channel cannot be offset. Therefore, the noise in the downlink channel is bound to affect the performance of the DPFL.

Then, we will analyze the influence of privacy budget ϵ and clipping threshold C on scheme VPPFL. The privacy budget ϵ determines the scale of noise in DP. When ϵ is larger, less noise is added and the model performs better. The introduction of clipping threshold C provides an important basis for the definition of global sensitivity, which also has a certain impact on the performance of model. Fig. 4 show the accuracy of the global model when the privacy budget $epsilon$ is equal to 1 and 10, and the clipping threshold C is equal to 1, 5 and 10 for both MNIST and CIFAR-10. According to the results, it can be seen that the accuracy of the global model is affected when $C = 1$. This is because the value of C is too small, which causes parameters to be sharply clipped, changing the original gradient descent direction. When C is equal to 5 or 10, the clipping threshold has little effect, so there is little difference in the performance of models under these two conditions. Besides, we can find that the model is more accurate and stable when $\epsilon = 10$. Because the noise added to parameters is less than that when $\epsilon = 1$. The experiments on MNIST and CIFAR-10 show the same trend. To get a better tradeoff



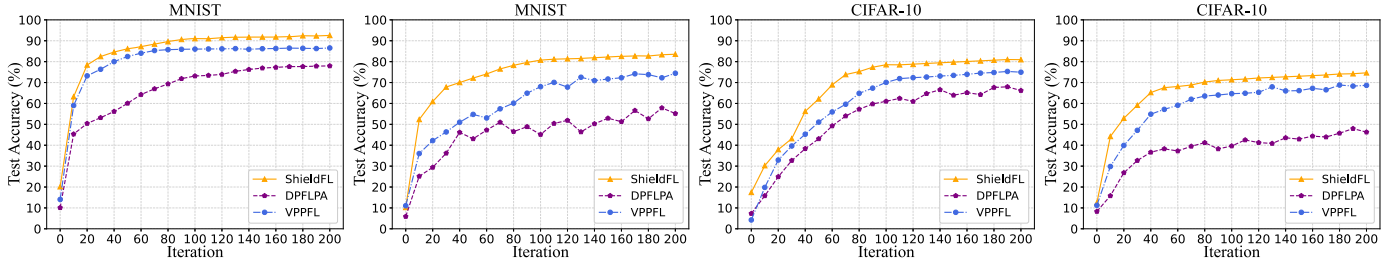
(a) $\epsilon = 1$, $C = 5$ (b) $\epsilon = 1$, $C = 5$

Fig. 3. The performance of VPPFL and DPFL. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)



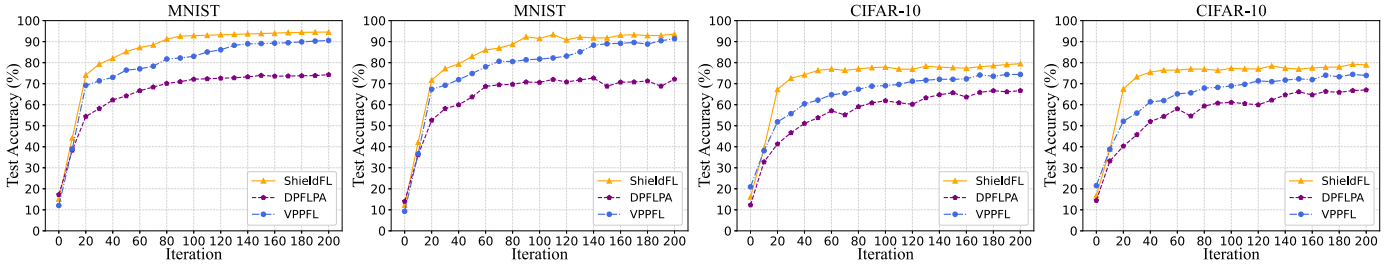
(a) $\epsilon = 1$ (b) $\epsilon = 10$ (c) $\epsilon = 1$ (d) $\epsilon = 10$

Fig. 4. The influence of the privacy budget ϵ and the clipping threshold C.



(a) $m = 10\%$ (b) $m = 30\%$ (c) $m = 10\%$ (d) $m = 30\%$

Fig. 5. The detection effect against untargeted attack with IID setting.



(a) $m = 10\%$ (b) $m = 30\%$ (c) $m = 10\%$ (d) $m = 30\%$

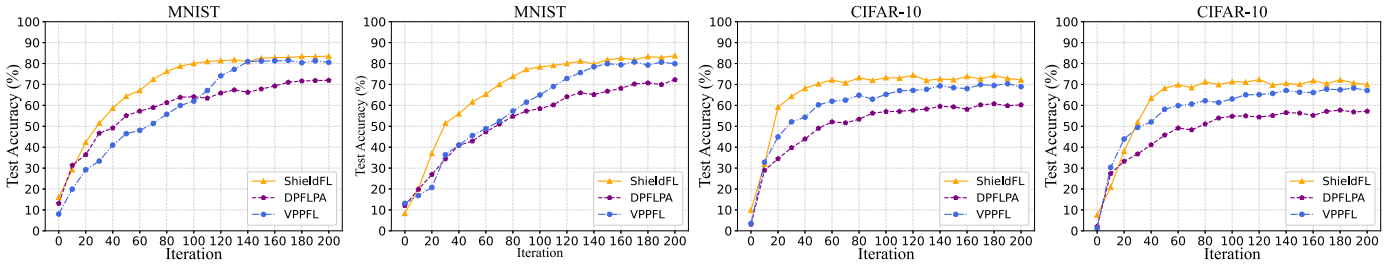
Fig. 6. The detection effect against targeted attack with IID setting.

between the privacy and performance, we set $\epsilon = 10$ and $C = 5$ in the following experiments.

7.3. The detection effect of VPPFL

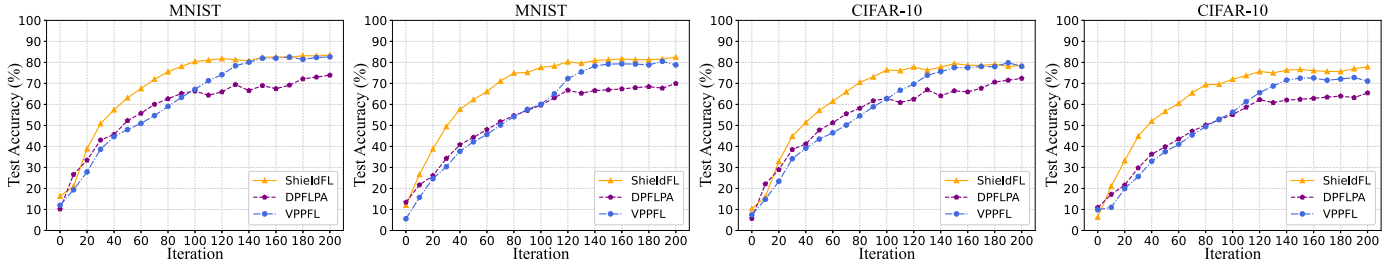
In this section, we conduct experiments to evaluate the detection effect of VPPFL. We compare VPPFL with two advanced algorithms, ShieldFL (Z. Ma et al., 2022) and DPFLPA (Zhou et al., 2022). ShieldFL is a privacy-preserving federated learning scheme using two-trapdoor homomorphic encryption, which can resist encrypted model poisoning without compromising privacy. DPFLPA is a differentially private federated learning scheme against poisoning attacks, which can perform anomaly detection on the perturbed parameters.

We first consider the setting of independent and identically distributed datasets. We assume that there are 10 users participating in FL and each user has 3000 IID samples. The proportion m of malicious users is 10% and 30%. For simulation parameters, we set $\epsilon = 10$, $C = 5$, $\delta = 0.01$ and $\mu = 2$. We conduct experiments on both MNIST and CIFAR-10 datasets, and the results are shown in Fig. 5 and Fig. 6. We can find that whether malicious users adopt untargeted attack or targeted



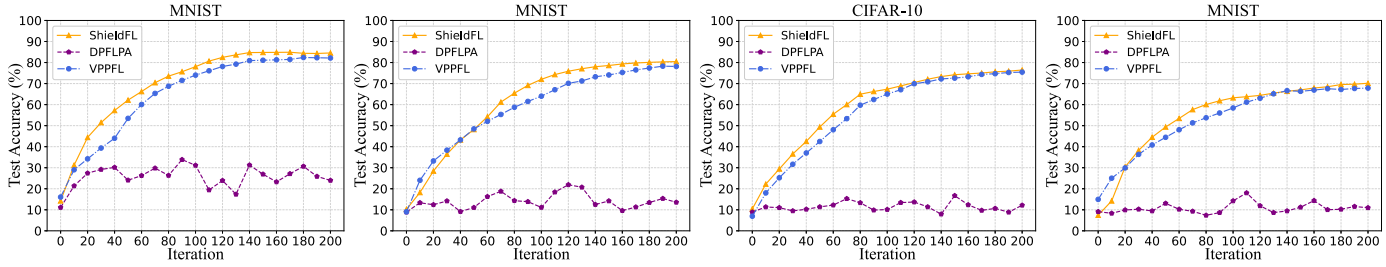
(a) $m = 10\%$ (b) $m = 30\%$ (c) $m = 10\%$ (d) $m = 30\%$

Fig. 7. The detection effect against untargeted attack with non-IID setting.



(a) $m = 10\%$ (b) $m = 30\%$ (c) $m = 10\%$ (d) $m = 30\%$

Fig. 8. The detection effect against targeted attack with non-IID setting.



(a) $m = 10\%$ (b) $m = 30\%$ (c) $m = 10\%$ (d) $m = 30\%$

Fig. 9. The detection effect against untargeted attacks when $N = 30$.

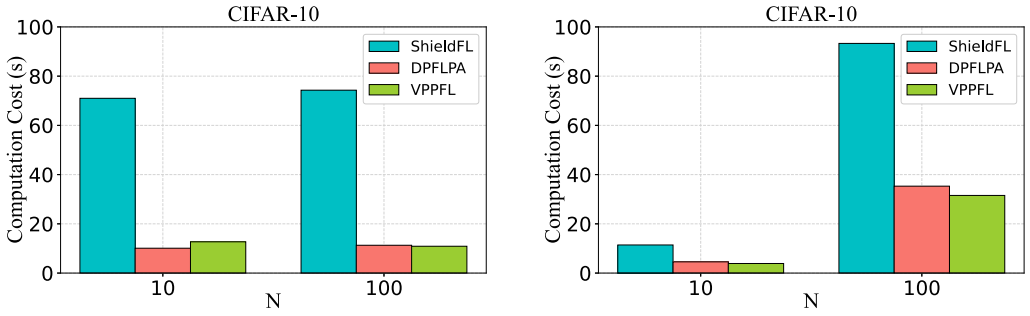
attack, the test accuracy of three schemes can converge. It means all of them can effectively detect malicious users under this condition. In addition, since differential privacy noise needs to be added in VPPFL and DPFLPA, the accuracy is lower than ShieldFL. Moreover, VPPFL performs better than DPFLPA, because it only needs to perturb in the uplink channel.

Then, we consider the setting of non independent and identically distributed datasets. Each user holds samples belonging to a single class, while the rest of settings remain unchanged. The results are shown in Fig. 7 and Fig. 8. We can observe that for the non-IID setting, all three frameworks are also effective in resisting model poisoning attacks. However, the convergence speed is slightly slower than that in the IID setting. Besides, as the number of epochs increases, the performance of VPPFL approaches that of ShieldFL, significantly surpassing DPFLPA.

After that, we want to analyze the impact of the number of users on the three frameworks. We increase the number of users to 30 and each user has 1000 samples. We assume that the samples trained by users are IID, and malicious users will engage in untargeted attacks. The rest of the settings remain unchanged. The results are shown in Fig. 9. It can be found that the test accuracy of ShieldFL and VPPFL is high, while that of DPFLPA extremely decreases. This phenomenon indicates that when more users participate in FL, ShieldFL and VPPFL can still detect the model poisoning attacks effectively, while DPFLPA is unable to detect. This is because in the design of scheme, the detection capability of ShieldFL and VPPFL does not change with the number of users, but DPFLPA is only applicable when the number of users is small. Besides, with the increase of epochs, we find that the accuracy of VPPFL is gradually close to ShieldFL. This is because as the number of users increases, the differential privacy noise added to the uplink channel can be better offset, thus reducing the impact of it on the VPPFL.

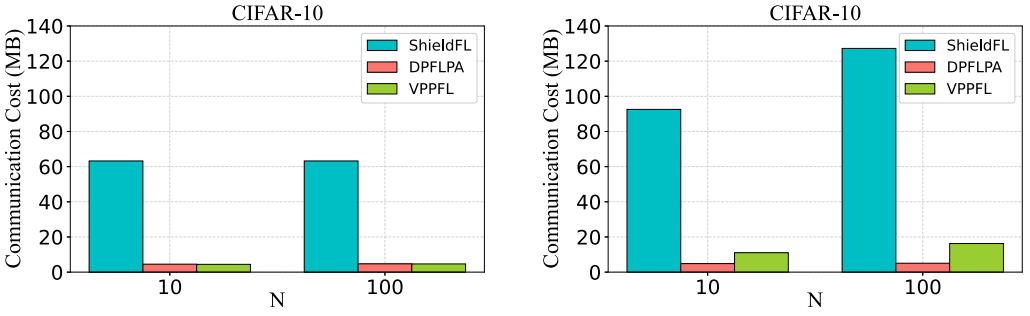
7.4. The efficiency of VPPFL

In this section, we conduct experiments to evaluate the efficiency of VPPFL. We set the number of users N to be 10 or 100, and each user has 100 samples. First, we compare the computation cost of ShieldFL, DPFLPA and VPPFL on CIFAR-10. We calculate the running time in user and server side in 10 iterations and take the average value as the computation cost. The experimental results are shown in Fig. 10. Obviously, the computation cost of DPFLPA and VPPFL is only 15%-30% of that of Shield. This is because we need to calculate on ciphertext in ShieldFL, which will increase the computation cost greatly. In addition, we compare the communication cost of these schemes on CIFAR-10, which is measured by the size of the parameters transmitted between users and servers. The communication cost of DPFLPA and VPPFL is only 7%-14% of that of Shield. In Fig. 11(a), we present the communication cost in user side. In ShieldFL, each user transmits the encrypted parameters, which occupy more space than the perturbed parameters in DPFLPA



(a) The computation cost in user side (b) The computation cost in server side

Fig. 10. The comparison result of computation cost in each iteration.



(a) The communication cost in user side (b) The communication cost in server side

Fig. 11. The comparison result of communication cost in each iteration.

and VPPFL. Fig. 11(b) shows the communication cost in server side. This cost consists of two parts, the cost for detecting the poisoning parameters and the cost for sending the aggregated parameters to user i. Since there is only one server in DPFLPA, and there is no need to communicate when detecting the poisoning parameters, so its communication cost is minimal.

The cost of VPPFL is slightly higher than that of DPFLPA, but these small additional cost is insignificant compared to its huge advantage in detection capability. Although the detection ability of ShieldFL is not weaker than VPPFL, its cost is the largest and much larger than the other two schemes.

8. Conclusion

In this paper, we propose VPPFL, a verifiable privacy-preserving federated learning scheme against poisoning attacks. Particularly, we design two technical components in VPPFL: 1) a privacy-preserving mechanism based on symmetric encryption and differential privacy, 2) a verification mechanism to resist poisoning attacks in privacy-preserving federated learning. Furthermore, we theoretically analyze our scheme from the perspectives of security, convergence and complexity, respectively. The experimental results shown that our scheme still performs well with a large number of users. Moreover, it is able to detect poisoning attacks precisely with only 15%-30% computation cost and 7%-14% communication cost compared with prior works using homomorphic encryption.

A number of avenues for further work are attractive. We evaluate our scheme on a balanced distribution of users' data. For the cases which users' data is non-IID, we leave them for future exploration.

CRediT authorship contribution statement

Yuxian Huang is responsible for the design of scheme, the experiment and the writing of paper.

Hao Zhou is responsible for the improvement of scheme.

Geng Yang, Hua Dai, Dong Yuan and Shui Yu are responsible for are responsible for guiding the research direction and providing suggestions for revising the paper.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (62372244, 61972209), the Postgraduate Research & Practice Innovation Program of Jiangsu Province (KYCX210770).

References

Alessandro, A., Laura, B., Jeff, H., 2022. How privacy's past may shape its future. Science, 270–272.

Aono, Y., Hayashi, T., Wang, L., et al., 2017. Privacy-preserving deep learning via additively homomorphic encryption. IEEE Trans. Inf. Forensics Secur. 13 (5), 1333–1345.

Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., Shmatikov, V., 2020. How to backdoor federated learning. In: Proc. Int. Conf. Artif. Intell. Statist. (AISTATS), pp. 2938–2948.

Bhagoji, A., Chakraborty, S., Mittal, P., Calo, S., 2019. Analyzing federated learning through an adversarial lens. In: Proc. Int. Conf. Mach. Learn. (ICML), pp. 634–643.

Blanchard, P., Mhamdi, E., Guerraoui, R., et al., 2017. Machine learning with adversaries: byzantine tolerant gradient descent. In: Neural Information Processing Systems (NIPS).

Cao, X., Gong, N.Z., 2022. MPAF: model poisoning attacks to federated learning based on fake clients. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 3395–3403.

Cao, X., Zhang, Z., Jia, J., Gong, N.Z., 2022. FLCert: provably secure federated learning against poisoning attacks. IEEE Trans. Inf. Forensics Secur. 17, 3691–3705.
Dwork, C., 2011. A firm foundation for private data analysis. Commun. ACM 54 (1), 86–95.
Dwork, C., McSherry, F., Nissim, K., et al., 2006. Calibrating noise to sensitivity in private data analysis. In: Theory of Cryptography Conference (TCC), pp. 363–385.
Fang, M., Cao, X., Jia, J., Gong, N., 2020. Local model poisoning attacks to Byzantine-robust federated learning. In: Proc. USENIX Secur. Symp., pp. 1605–1622.
Fung, C., Yoon, C.J., Beschastnikh, I., 2018. Mitigating sybils in federated learning poisoning. arXiv preprint. arXiv:1808.04866.
Gentry, C., Sahai, A., Waters, B., 2013. Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Proc. Annu. Cryptol. Conf., pp. 75–92.
Jagielski, M., Oprea, A., Biggio, B., et al., 2018. Manipulating machine learning: poisoning attacks and countermeasures for regression learning. In: Proc. IEEE Symp. Secur. Privacy (SP), pp. 19–35.
Jiang, B., Li, J., Wang, H., et al., 2023. Privacy-preserving federated learning for industrial edge computing via hybrid differential privacy and adaptive compression. IEEE Trans. Ind. Inform. 19 (2), 1136–1144.
Jonas, G., Hartmut, B., Hannah, D., Michael, M., 2020. Inverting gradients-how easy is it to break privacy in federated learning? Adv. Neural Inf. Process. Syst. 33, 16937–16947.
Khosravy, M., Nakamura, K., Hirose, Y., et al., 2022. Model inversion attack by integration of deep generative models: privacy-sensitive face generation from a face recognition system. IEEE Trans. Inf. Forensics Secur. 17, 357–372.
LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444.
Li, H., et al., 2023. 3DFed: adaptive and extensible framework for covert backdoor attack in federated learning. In: IEEE Symposium on Security and Privacy (SP), pp. 1893–1907.
Li, X., Huang, K., Yang, W., et al., 2020. On the convergence of FedAvg on non-IID data. In: International Conference on Learning Representations (ICLR).
Li, X., Qu, Z., Zhao, S., et al., 2023. LoMar: a local defense against poisoning attack on federated learning. IEEE Trans. Dependable Secure Comput. 20 (1), 437–450.
Lim, W., Luong, N., Hoang, D., et al., 2020. Federated learning in mobile edge networks: a comprehensive survey. IEEE Commun. Surv. Tutor. 22 (3), 2031–2063.
Liu, L., Wang, Y., Liu, G., et al., 2022. Membership inference attacks against machine learning models via prediction sensitivity. IEEE Trans. Dependable Secure Comput. https://doi.org/10.1109/TDSC.2022.3180828.
Liu, X., Li, H., Xu, G., Chen, Z., Huang, X., Lu, R., 2021. Privacy-enhanced federated learning against poisoning adversaries. IEEE Trans. Inf. Forensics Secur. 16, 4574–4588.
Ma, X., Sun, X., Wu, Y., et al., 2022. Differentially private byzantine-robust federated learning. IEEE Trans. Parallel Distrib. Syst. 33 (12), 3690–3701.
Ma, Z., Ma, J., Miao, Y., et al., 2022. ShieldFL: mitigating model poisoning attacks in privacy-preserving federated learning. IEEE Trans. Inf. Forensics Secur. 17, 1639–1654.
Mandal, K., Gong, G., 2019. PrivFL: practical privacy-preserving federated regressions on high-dimensional data over mobile networks. In: ACM SIGSAC Conference.
Mao, Y., Yuan, X., Zhao, X., Zhong, S., 2021. Romoa: robust model aggregation for the resistance of federated learning to model poisoning attacks. In: 26th European Symposium on Research in Computer Security, pp. 476–496.
Melis, L., Song, C., Cristofaro, E.D., Shmatikov, V., 2019. Exploiting unintended feature leakage in collaborative learning. In: Proc. IEEE Symp. Secur. Privacy (SP), pp. 691–706.
Miao, Y., Liu, Z., Li, H., et al., 2022. Privacy-preserving byzantine-robust federated learning via blockchain systems. IEEE Trans. Inf. Forensics Secur. 17, 2848–2861.
Mohr, F., Wever, M., Tornede, A., Hullermeier, E., 2021. Predicting machine learning pipeline runtimes in the context of automated machine learning. IEEE Trans. Pattern Anal. Mach. Intell. 43 (9), 3055–3066.
Shejwalkar, V., Houmansadr, A., 2021. Manipulating the byzantine: optimizing model poisoning attacks and defenses for federated learning. In: Network and Distributed Systems Security (NDSS) Symposium.
Su, Z., Wang, Y., Luan, T., et al., 2022. Secure and efficient federated learning for smart grid with edge-cloud collaboration. IEEE Trans. Ind. Inform. 18 (2), 1333–1344.
Sun, Y., Ochiai, H., Sakuma, J., 2022. Semi-targeted model poisoning attack on federated learning via backward error analysis. In: International Joint Conference on Neural Networks (IJCNN), pp. 1–8.
Tolpegin, V., Truex, S., Gursoy, M.E., Liu, L., 2020. Data poisoning attacks against federated learning systems. In: Proc. Eur. Symp. Res. Comput. Secur. (ESORICS), pp. 480–501.
Wang, Q., Li, Z., Zou, Q., Zhao, L., et al., 2020. Deep domain adaptation with differential privacy. IEEE Trans. Inf. Forensics Secur. 15, 3093–3106.
Wei, K., Li, J., Ding, M., et al., 2020. Federated learning with differential privacy: algorithms and performance analysis. IEEE Trans. Inf. Forensics Secur. 15, 3454–3469.
Xu, C., Ren, J., Zhang, Y., et al., 2017. DPPro: differentially private high dimensional data release via random projection. IEEE Trans. Inf. Forensics Secur. 12 (12), 3081–3093.
Yang, Q., Liu, Y., Chen, T., et al., 2019. Federated machine learning: concept and applications. ACM Trans. Intell. Syst. Technol. 10 (2), 1–19.
Zeng, Y., Lin, Y., Yang, Y., et al., 2022. Differentially private federated temporal difference learning. IEEE Trans. Parallel Distrib. Syst. 33 (11), 2714–2726.
Zhang, J., Chen, B., Yu, S., Deng, H., 2019. PEFL: a privacy-enhanced federated learning scheme for big data analytics. In: IEEE Global Communications Conference (GLOBECOM), pp. 1–6.
Zhang, J., Chen, B., Cheng, X., et al., 2020. PoisonGAN: generative poisoning attacks against federated learning in edge computing systems. IEEE Int. Things J. 8 (5), 3310–3322.
Zheng, Y., Lai, S., Liu, Y., Yuan, X., Yi, X., Wang, C., 2022. Aggregation service for federated learning: an efficient, secure, and more resilient realization. IEEE Trans. Dependable Secure Comput. https://doi.org/10.1109/TDSC.2022.3146448.
Zhou, J., Wu, N., Wang, Y., 2022. A differentially private federated learning model against poisoning attacks in edge computing. IEEE Trans. Dependable Secure Comput. https://doi.org/10.1109/TDSC.2022.3168556.

Yuxian Huang received the B.S. degree from the Department of Computer Science, Nanjing University of Posts and Telecommunications, Jiangsu, China, in 2020, where he is currently pursuing Ph.D. degree with the Department of Cyberspace Security. His research interests include differential privacy, information security and federated learning.

Shui Yu (Fellow, IEEE) received the Ph.D. degree from Deakin University, Burwood, Australia, in 2004. He is currently a Professor with the School of Computer Science, University of Technology Sydney, Australia. In 2013, he initiated the research field of networking for Big Data, and his research outputs have been widely adopted by industrial systems, such as Amazon cloud security. He has authored or coauthored four monographs, edited two books, more than 400 technical papers, including top journals and top conferences, such as IEEE Transactions on Parallel and Distributed System, IEEE Transactions on Information Forensics and Security, IEEE Transactions on Computers and the INFOCOM. His research interests include Big Data, security and privacy, networking, and mathematical modeling.

Geng Yang (Member, IEEE) received the Ph.D degree in Computational Mathematics from the Laval University, Canada, in 1994. From 1994 to 1996, he worked as a postdoctoral researcher at the University of Montreal, Canada. He is currently a Professor and Ph.D. supervisor with the School of Computer Science, Nanjing University of Posts and Telecommunications, China. His research interests include computer communication and networks, parallel and distributed computing, cloud computing and information security.

Hao Zhou received the B.S. degree from the Department of Computer Science, Nanjing University of Posts and Telecommunications, Jiangsu, China, in 2015, where he is currently pursuing Ph.D. degree with the Department of Cyberspace Security. His current research interests include information security and deep learning.

Hua Dai (Member, IEEE) received the Ph.D degree in Computer Application Technology from the Nanjing University of Aeronautics and Astronautics, China, in 2011. He is currently a Professor and Ph.D. supervisor with the School of Computer Science, Nanjing University of Posts and Telecommunications, China. His research interests include database security, behavior identification and deep learning.

Dong Yuan (Member, IEEE) received the B.Eng. and M.Eng. degrees from Shandong University, Jinan, China, in 2005 and 2008, respectively, and the Ph.D. degree from the Swinburne University of Technology, Melbourne, Australia, in 2012, all in computer science. He is currently a Senior Lecturer with the School of Electrical and Information Engineering, University of Sydney, Sydney, Australia. His research interests include cloud computing, scheduling and resource management, cyberspace security, internet of things, and workflow systems.