2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

Privacy-Preserving Defense Against Poisoning Attacks in Federated Learning

Youlin Huang[1], Bencan Gong[2] and Shuoxiang Wang[3]

[1,3] Hubei Key Laboratory of Intelligent Vision Monitoring for Hydropower Engineering, Three Gorges University, Yichang, China
[2] College of Computer and Information Technology, Three Gorges University, Yichang, China
gonbc@sina.com

Abstract. Federated Learning (FL), as a collaborative training paradigm that does not rely on raw data sharing, faces dual security threats of privacy leakage and data poisoning attacks. These threats not only compromise client data privacy but also degrade the performance of the global model. To address this challenge, we propose a Privacy-Preserving Defense against Poisoning Attacks (PPDPA), which integrates privacy preservation and poisoning detection through a lossless masking mechanism. In this framework, the gradient uploaded by each client is first masked using a removable mask to protect gradient privacy. Without revealing the original gradients, the masked gradients are then aggregated, and Singular Value Decomposition (SVD) is employed to extract features and perform dimensionality reduction. In the resulting low-dimensional space, a clustering-based approach is used to identify poisoned gradients. Additionally, a verification mechanism is designed to ensure the integrity of the masking process during aggregation, effectively preventing attackers from manipulating the mask for stealthy poisoning. Finally, poisoned gradients are either removed during aggregation to defend against data poisoning attacks. Extensive experiments demonstrate that PPDPA outperforms existing state of the art privacy-preserving detection methods in both detection accuracy and defense efficiency.

Keywords: Federated Learning (FL), Defense Mechanism, Privacy Preservation, Label Flipping Attacks, Singular Value Decomposition (SVD).

1 INTRODUCTION

Federated Learning (FL) [1], as an emerging distributed machine learning paradigm, has already shown its potential in privacy-sensitive domains. FL enables collaborative model training while ensuring that data remains on edge devices, avoiding the need for centralized data storage [2, 3]. However, with the expansion of its applications, researchers have discovered that attackers may be able to infer local data [4, 5] from gradient information, which undoubtedly poses new challenges to data privacy.

To address this issue, Privacy-Preserving Federated Learning (PPFL) has emerged [6, 7]. By employing various privacy-preserving techniques to conceal the original gradients, PPFL effectively enhances data privacy. However, existing privacy-preserving techniques face several challenges in practical applications. On one hand, while Secure

Multi-party computation (MPC) [8, 9] can be employed, it incurs substantial communication overhead due to frequent information exchanges among clients, severely limiting its practicality in scenarios with numerous clients or constrained communication resources. On the other hand, selecting an appropriate differential privacy (DP) [10, 11] budget remains highly challenging, as it requires a delicate trade-off between privacy protection and model performance. Consequently, when gradients are encrypted or masked, effectively defending against poisoning attacks becomes a critical yet unresolved challenge.

Due to its high computational efficiency and negligible impact on model performance, the Singular Value Decomposition (SVD)-based masking mechanism is considered an effective and practical approach for gradient protection in FL [12]. The core idea is as follows: Instead of directly uploading raw gradients, each client perturbs its locally computed gradient matrix by applying two locally generated orthogonal masking matrices and, producing an encrypted gradient. Upon receiving all masked gradients from clients, the server can reconstruct the original gradients using pre-agreed inverse transformations.

However, the SVD-based masking mechanism also has potential security risks. It provides untrusted clients with two potential poisoning attack vectors: 1) A malicious client may upload carefully crafted poisoned gradients under the cover of SVD masking, contaminating the global model in a manner similar to traditional poisoning attacks. 2) An attacker could tamper with the orthogonal masking matrices, introducing bias errors during gradient reconstruction or aggregation, even if the original gradients themselves are benign.

We observe two main limitations in existing research: 1) Most studies [13, 14] separately address privacy preservation (e.g., encryption, masking, differential privacy) and robust defense (e.g., anomaly detection, trusted aggregation), lacking a unified modeling framework. 2) Existing defense methods heavily rely on prior knowledge and ideal assumptions, such as the existence of trusted clients or known attack ratios. To overcome these limitations, we propose a novel removable noise mechanism that not only effectively preserves data privacy but also enables precise identification of poisoned data within masked matrices. The main contributions of this study are summarized as follows.

1. We propose a PPFL poisoning detection method named PPDPA, which effectively defends against poisoning attacks by analyzing the masked gradient features of the final-layer neurons. The method integrates SVD to extract key features and reduce dimensionality, significantly enhancing efficiency and compressing data volume. Furthermore, a removable masking mechanism is introduced to eliminate sensitive information without compromising model accuracy, thereby ensuring secure data protection.
2. We propose a dual-sharing and commitment mechanism to validate both masked and original gradients, enabling effective detection of poisoned gradients linked to anomalous masking. To further improve robustness against more covert adversarial strategies, we incorporate the Calinski-Harabasz (CH) index to assess clustering quality and enhance the reliability of the detection process.

3. We evaluated the performance of PPDPA in detecting malicious clients on both CIFAR-10 and MNIST datasets, covering scenarios with attacker ratios of up to 40%. Compared to various advanced defense methods, PPDPA demonstrates superior performance in test scenarios.

2 RELATED WORK

2.1 Defense Algorithms Against Data Poisoning Attacks

In FL, data poisoning attacks disrupt model training by manipulating local training data, seriously impacting the performance and security of the global model. To address this issue, researchers have proposed various plaintext aggregation defense methods that identify and isolate abnormal updates by analyzing the distributional characteristics of client gradients, thereby enhancing the model's robustness in adversarial environments.

FL-Defender [17] identifies suspicious updates by extracting client update features and applying anomaly detection techniques (e.g., Local Outlier Factor). While demonstrating strong adaptability and interpretability, it suffers from high computational costs in high-dimensional spaces and sensitivity to parameter settings. Fung et al. [15] proposed a detection algorithm called FoolsGold that dynamically adjusts aggregation weights based on historical similarity of client updates, effectively defending against colluding attacks. However, it tends to misclassify benign participants in non-IID (non-Independent and Identically Distributed) data environments.

In Bulyan [18], it combines Krum and Trimmed Mean advantages, achieving stronger theoretical robustness at the cost of higher computational complexity and requiring predefined attacker ratios. The authors in [16] introduced the use of Principal Component Analysis (PCA) to address this issue. Building upon this, [21] proposed an enhanced defense method by incorporating Kernel Principal Component Analysis (KPCA) combined with K-means clustering. In a related approach, [19] applied K-means clustering to group the gradients of the output layer, enabling the identification of neurons with the largest gradient magnitudes as likely source neurons and target classes. FLAME [20] uses parameter pruning and recovery to remove backdoors, along with Mahalanobis distance to detect and filter suspicious updates. While effective against various attacks, it converges slowly and struggles when too many clients are malicious.

2.2 Privacy-Preserving Algorithms

In FL, privacy algorithms focus on preventing sensitive information leakage during gradient transmission. Although local training avoids centralized data storage, client model updates can still expose user data. Therefore, various protection methods have been proposed to reduce this risk and ensure security in sensitive environments.

In the study of poisoning detection under encrypted or masked data settings, Liu et al. proposed the Privacy-Enhanced Federated Learning (PEFL) framework [6], the first

solution capable of identifying anomalous gradients in ciphertext. PEFL employs homomorphic encryption (HE) [5] to compute Pearson correlation coefficients between encrypted gradients, distinguishing malicious from benign updates while preserving data privacy and improving defense efficacy. However, the computational overhead of homomorphic encryption limits its scalability in large-scale deployments. In addition to HE, Zhao et al. [23] presented a scheme that performs aggregation within a Trusted Execution Environment (TEE) under constrained memory conditions.

Differential privacy (DP) techniques are also widely applied in FL poisoning defense. Ziteng et al. [24] mitigated malicious gradient impacts through gradient clipping and Gaussian noise injection. While computationally efficient, DP inevitably trades off model convergence speed and precision, with limited effectiveness against sophisticated attacks. For MPC, Chen et al. [22] performs encrypted gradient distance calculations via MPC protocols. Despite rigorous privacy guarantees, complex computations and multi-round interactions cause significant efficiency degradation with growing client numbers or model sizes. Overall, these methods strike varying balances between privacy preservation and poisoning detection, yet further optimizations in efficiency, accuracy, and practical deploy ability remain imperative.

In conclusion, the dual trade-off between robustness and privacy protection remains a key challenge. How to achieve efficient and accurate anomaly detection and robust aggregation without exposing the original gradients continues to be a core difficulty.

3 PRELIMINARIES

3.1 Federated Learning

FL is a privacy-preserving distributed machine learning framework. Unlike traditional centralized training approaches, FL enables multiple clients (e.g., mobile devices, edge nodes) to participate in model training while keeping raw data locally stored, eliminating the need to upload data to a central server. This approach not only preserves data privacy but also significantly reduces communication overhead. More precisely, the FL procedure can be formulated as follows:

1. Model Initialization and Distribution: The server first initializes the global model parameters $W^{(0)}$ and distributes them to a selected set of clients $C \subseteq \{1,2,\dots,n\}$.
2. Local Client Training: Each client $i \in C$ performs local training on its dataset $D_i = \{< x_j, y_i >, (j = 1,2,\dots,m)\}$ using the current global model parameters $W^{(t)}$, where m denotes the number of samples in local dataset D_i for client i.

   The local objective function can be defined as:

$$L_i(W) = \frac{1}{|D_i|}\sum_{j=1}^{|D_i|}\mathcal{L}\left(W; x_j, y_j\right) \tag{1}$$

   where $\mathcal{L}(\cdot)$ is the loss function for a single sample. The client updates its local model through several steps of gradient descent (e.g., SGD):

$$W_i^{(t+1)} = W^{(t)} - \eta \nabla L_i(w^{(t)}) \quad (2)$$

where η is the learning rate and ∇L_i is the gradient of the local loss function with respect to the model parameters.

3. Server Model Aggregation: All participating clients upload their updated model parameters $W_i^{(t+1)}$ to the server, which performs weighted averaging to generate the new global model as shown in (3):

$$W^{(t+1)} = \sum_{i \epsilon C_n} \frac{|D_i|}{\sum_{j \epsilon C_n} |D_j|} W_i^{(t+1)} \quad (3)$$

This weighting strategy is known as the Federated Averaging (FedAvg) [25] algorithm, which is one of the most classic and widely used methods in FL. Steps 2 and 3 are repeated until the model on the server meets the desired accuracy or reaches the predefined number of iterations.

3.2 Label Flipping Attack

Label Flipping Attack (LFA) constitutes a classic targeted data poisoning attack, initially proposed in centralized machine learning [27] environments and widely applied in security-sensitive classification tasks such as intrusion detection and spam filtering. In centralized settings, attackers typically gain control over partial training data and subvert model learning by manipulating labels to induce erroneous decision boundaries.

With the widespread adoption of FL in practical applications, LFA have demonstrated enhanced attack potential in this distributed setting. Due to clients having complete access and control over their local data, malicious participants can stealthily manipulate labels by altering samples from a source class to a predetermined target class while preserving original feature values. These poisoned samples are then used for local training, generating misleading gradients that subsequently get uploaded to the server for global model aggregation.

Research indicates that even with a low proportion of attackers, the global model may exhibit significant prediction bias toward the target class for source-class samples, substantially compromising overall model robustness. Unlike Byzantine attacks that randomly perturb parameters or gradients, LFA specifically distorts decision boundaries between targeted classes while minimally affecting accuracy on non-target classes, resulting in weaker detectability and stronger stealth. Consequently, developing detection and defense mechanisms against LFA has become a critical research focus in FL security.

3.3 Removable Mask Mechanism

SVD is a fundamental and widely used matrix decomposition technique. Given a real matrix $G \in \mathbb{R}^{m \times n}$, where m is the number of rows and n is the number of columns. SVD can decompose it into the product of three matrices as shown in (4), where $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ are the left and right singular matrices with orthonormal col-

umns, and $\Sigma \in \mathbb{R}^{m \times n}$ is a diagonal matrix. The non-negative real numbers on the diagonal of Σ, denoted as $\sigma_1, \sigma_2, \ldots, \sigma_r$ (r = rank (G)), are the singular values of the matrix G.

$$G = U\Sigma V^{\mathrm{T}} \tag{4}$$

Building upon the above analysis, a mask mechanism based on SVD can be designed to perturb and protect the original matrix. Specifically, let $X \in \mathbb{R}^{m \times m}$ and $Y \in \mathbb{R}^{n \times n}$ be two random orthogonal matrices. By multiplying the original data matrix G from the left and right, a masked version of the transformed matrix is obtained, as shown in (5).

$$\tilde{G} = XGY \tag{5}$$

From (1) and (2), we know that the masked matrix $\tilde{G}$ can be decomposed using SVD, and the resulting decomposition can be expressed as shown in (6). Based on the properties of orthogonal matrices, as demonstrated in (7) and (8), $\tilde{U}$ and $\tilde{V}$ remain orthogonal matrices, where $\tilde{U} = XU, \tilde{V}^{\mathrm{T}} = V^{\mathrm{T}}Y, \tilde{\Sigma} = \Sigma$.

$$\tilde{G} = \tilde{U}\tilde{\Sigma}\tilde{V}^{\mathrm{T}} = XU\Sigma V^{\mathrm{T}}Y \tag{6}$$

$$(XU)^{-1} = U^{-1}X^{-1} = U^{\mathrm{T}}X^{\mathrm{T}} = (XU)^{\mathrm{T}} \tag{7}$$

$$(V^{\mathrm{T}}Y)^{-1} = Y^{-1}(V^{\mathrm{T}})^{-1} = Y^{\mathrm{T}}V = (V^{\mathrm{T}}Y)^{\mathrm{T}} \tag{8}$$

In summary, the SVD-based masking method employs orthogonal transformations to perturb data, effectively protecting gradients or parameters uploaded by clients while supporting lossless decoding and restoration. This approach demonstrates favorable computational efficiency and reversibility.

3.4 Feature Extraction using SVD

SVD decomposes a matrix into three components, allowing for the extraction of essential information and facilitating dimensionality reduction. It is closely associated with Principal Component Analysis (PCA), which is one of the most commonly used techniques for reducing data dimensionality.

Let there be a matrix $G_{m \times n} = [x_1, x_2, \ldots, x_n]^{\mathrm{T}}$, where each row x_i is a $1 \times m$ vector. Our goal is to extract the main features from $G_{m \times n}$ while reducing its dimensionality. To achieve this, we apply SVD to decompose $G_{m \times n}$ as $U\Sigma V^{\mathrm{T}}$. It can be proven that the column vectors of matrix V are in fact the eigenvectors of $G^{\mathrm{T}}G$, which enables us to perform dimensionality reduction via PCA.

To effectively reduce the dimensionality of the original data, we retain only the top r directions corresponding to the largest singular values. The associated right singular vectors form the matrix $V_{n \times r}$, and the reduced-dimensional data matrix $\hat{G}_{m \times r}$ can be obtained using the following (9). This result represents a projection of the original data into an r-dimensional space, which captures the directions of greatest variation and richest information.

$$\hat{G}_{m \times r} = G_{m \times n}V_{n \times r} \tag{9}$$

Based on the above SVD decomposition, an approximate representation as shown in (10) can also be obtained. In other words, we can use $U_{m\times r}\Sigma_{r\times r}$ to represent the reduced-dimensional data $\hat{G}_{m\times r}$, thereby expressing the most important information from the original data with fewer dimensions.

$$G_{m\times n}V_{n\times r} = U_{m\times r}\Sigma_{r\times r} \tag{10}$$

4 PROPOSED ALGORITHM

4.1 Design Overview

As shown in Fig. 1, the system architecture comprises two servers, namely the Aggregation Server (AS) and the Mask Server (MS), along with *n* clients. Clients are responsible for local model training and uploading. Each client is required to generate two public commitments: one corresponding to the masked gradient and the other to the mask itself. AS identifies the authenticity of the gradients using statistical features and clustering methods, and verifies the masked gradients uploaded by clients. MS is tasked with receiving and validating the masks submitted by clients, aggregating those from all legitimate clients, and forwarding the aggregated result to the AS.

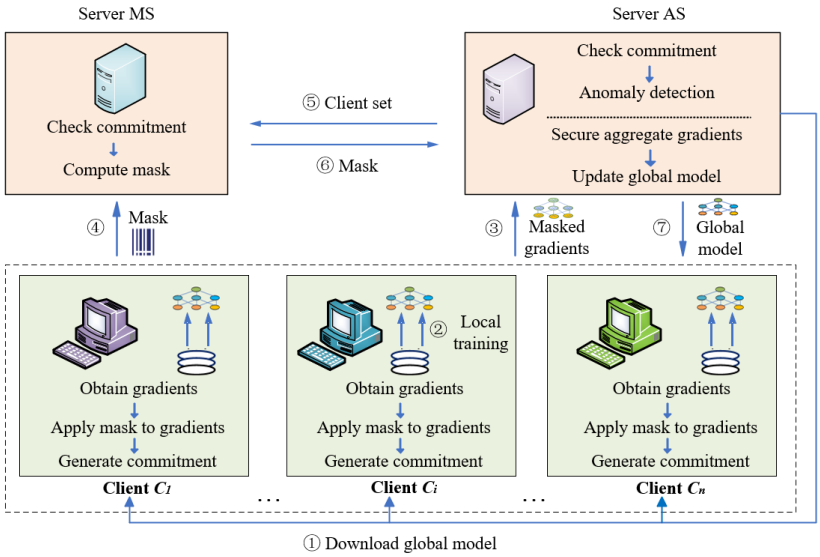

Fig. 1. System architecture.

Initially, AS creates and initializes a global model, then selects a subset of clients. In each iteration, the server and the selected clients perform the following operations:

1. Client downloads global model and performs training (see ① and ②). During each FL round, a set of k clients is selected, with each client C_i ($i \in ([1, k])$) downloading the current global model from the AS. Each client then trains the model locally on its private data and computes its local gradient, which is an $m \times n$ matrix.
2. Clients mask their gradients and upload them (see ③ and ④). Firstly, AS generates a mask matrix X and splits it into k column vectors, denoted as $X = [X_1, X_2, \dots, X_k]$ distributing each $X_i \in \mathbb{R}^{k\times 1}$ to the corresponding client. Meanwhile, each client generates its own mask matrix $Y \in R^{n\times n}$ and combines it with X_i to conceal its gradient G_i, resulting in a masked gradient. Each client then creates two public commitments: one for the masked gradient and another for the mask itself. The masked gradient is sent to the AS, while the mask is sent to MS. Both commitments are published to allow verification by AS and MS.
3. Servers verify gradients and masks (see ⑤ and ⑥). Malicious clients may launch poisoning attacks by submitting harmful gradients or incorrect masks. Specifically, after receiving the masked gradients, the AS applies SVD to reduce their dimensionality and extract the principal components that capture the most significant variance. These reduced client representations are then clustered using K-means, and the CH index is used to evaluate the clustering quality, enabling the identification of anomalous clients for effective poisoning detection and defense. In parallel, AS verifies the commitment from each client to ensure that the submitted masked gradient matches the corresponding public commitment. After completing the verification, AS forms a set of legitimate clients and sends the list to MS. MS then verifies that the submitted mask from each client matches the corresponding public commitment. Finally, MS sends the verified masks and client list back to AS.
4. AS aggregates the gradients and updates the global model (see ⑦). Upon receiving the masks, AS removes them to recover the aggregated gradient. The global model is then updated using this aggregated gradient. Finally, AS distributes the updated global model to all users for the next round of training.

The complete pseudocode description is provided in Algorithm 1.

Algorithm 1: PPDPA**.**

Input: C_t: Random set of k clients in training round t, X, Y, k
Output: benign client set
Divide the matrix X along its columns into k parts
Distribute X_i to client C_i
$W \leftarrow \emptyset$ //Initialize empty set for commitments
$\omega_{t-1} \leftarrow$ the global model resulting from round $t-1$
for each class $s \in [1, |O|]$ **do**
    for each $i \in C_t$ **do**
        $\omega_{t,i} \leftarrow$ obtain the model parameters at round t
        $\omega_{t,i}^{s} \leftarrow$ obtain the parameters of the last layer associated with class s
        $G_i^{(s)} \leftarrow \omega_{t,i}^{s}$
        $\tilde{G}_i^{(s)} \leftarrow X_i G_i^{(s)} Y$
        $cg_i^{(s)} \leftarrow Hash(\tilde{G}_i^{(s)})$

$cr_i^{(s)} \leftarrow Hash(X_i||Y)$
Add X_i in W
send $(\tilde{G}_i^{(s)}, cg_i^{(s)})$ to AS
send $(X_i||Y,\ cg_i^{(s)})$ to MS
$\tilde{P}_s \leftarrow \sum_i^k \tilde{G}_i^{(s)}$
$\tilde{U}_s, \tilde{\Sigma}_s, \tilde{V}_s^{\mathrm{T}} \leftarrow$ SVD $(\tilde{P}_s)$
$U_s \leftarrow W^{\mathrm{T}} \tilde{U}_s$
$\hat{P}_s \leftarrow U_s \tilde{\Sigma}_s$
$CH,\ (benign, malicious) \leftarrow$ Detecting Malicious Clients $(\hat{P}_s)$
for each $i \in C$ **do**
  if not Verify $(cg_i^{(s)}, \tilde{G}_i^{(s)})$ **then**
    remove *i* **from** C
Send C_1 to MS //Set of clients whose commitments are verified by AS
for each $i \in C_1$ **do**
  if not Verify $(cr_i^{(s)}, X_i||Y)$ **then**
    remove *i* **from** C_1
Send C_2 to AS //Set of clients whose commitments are verified by MS
return *the set with the higest CH*

4.2 Apply Mask to Gradients

During the t-th training round, each client C_i computes the local gradient for the current round using its own dataset D_i and local model parameters W_i. Each client C_i extracts the feature representations corresponding to the neurons in the output layer of the model.

Specifically, for each output-layer neuron, a weight vector is formed by collecting all the parameters that connect this neuron to the preceding hidden layer. These weight vectors determine how the outputs of the previous layer influence the activation of each output-layer neuron. We denote the weight vector of client C_i associated with the s-th output neuron as $G_i^{(s)}$, where s ranges from 1 to $|O|$. Then, to protect the privacy of the model parameters, each client C_i encrypts every weight vector $G_i^{(s)}$ using a masking matrix X_i and matrix Y. The masked weight vector can be computed using formula:

$$\tilde{G}_i^{(s)} = X_i G_i^{(s)} Y \quad (s = 1,2,\dots,|O|) \tag{11}$$

Finally, client C_i uploads all the masked output-layer weight vectors $\tilde{G}_i = \tilde{G}_i^{(1)}, \tilde{G}_i^{(2)}, \dots, \tilde{G}_i^{(|O|)}$ to the server AS for subsequent federated aggregation.

4.3 Commitment Verification

To ensure the verifiability and integrity of model updates uploaded by clients during FL without revealing the original gradients. This work introduces a commitment mechanism between clients and servers. Specifically, each client locally generates two commitment values for its masked matrix and masked gradients, respectively. These commitments are used for verification on different servers, thereby preventing clients from uploading tampered or inconsistent updates. The steps for commitment verification are as follows:

1. Client generates commitment values. In each round of FL, client C_i first performs local training to obtain a weight vector G_i, which consists of all parameters connecting each output layer neuron to the final hidden layer. Then, the client obtains two mask vectors X_i and Y_i with the same dimensions as G_i, and generates the masked gradient $\tilde{G}_i$=$X_i G_i Y_i$.
2. AS verifies the commitment values. AS verifies whether the masked gradient $\tilde{G}_i$ is consistent with the commitment cg_i submitted by client C_i, according to (12). This ensures that the uploaded model updates have not been tampered with.

$$Hash(\tilde{G}_i) = cg_i \tag{12}$$

3. MS verifies the commitment values. MS verifies whether the masks X_i and Y_i uploaded by the client are consistent with the commitment cr_i, according to (13). This prevents clients from forging masks to bypass system detection.

$$Hash = (X_i||Y_i) = cr_i \tag{13}$$

4.4 Secure Aggregation

AS aggregates the masked weight data from all clients, denoted as $\tilde{P}_s$. Accordingly, the original gradient data of each client can be represented as $P_s = [G_1^{(s)}, G_2^{(s)}, \dots, G_k^{(s)}]$. Thus, the relationship between $\tilde{P}_s$ and P_s is given by (14):

$$\begin{aligned}\tilde{P}_s &= \sum_{i \in k} \tilde{G}_i^{(s)} \\ &= X_1 G_1^{(s)} Y + X_2 G_2^{(s)} Y + \cdots + X_k G_k^{(s)} Y \\ &= [X_1, X_2, \dots, X_k]\left[G_1^{(s)}, G_2^{(s)}, \dots, G_k^{(s)}\right]^{\mathrm{T}} Y \\ &= [X_1, X_2, \dots, X_k] P_s Y \quad (s = 1,2, \dots, |O|)\end{aligned} \tag{14}$$

where X denotes a block matrix formed by concatenating the mask matrices of all clients.

To reduce the dimensionality of the data and extract the main features of the original weight vector $G_i^{(s)}$ from the matrix $\tilde{P}_s$, SVD is applied $\tilde{P}_s$ according to (4). As a result, (15) can be obtained as follows.

$$\tilde{P}_s = \tilde{U}_s \tilde{\Sigma}_s \tilde{V}_s^{\mathrm{T}} \tag{15}$$

Moreover, $U_s \Sigma_S V_s^{\mathrm{T}}$ can also be obtained from P_s through SVD. Based on the above analysis, we can summarize (16) as follows.

$$\tilde{U}_s \tilde{\Sigma}_s \tilde{V}_s^{\mathrm{T}} = [X_1, X_2, \dots, X_k] U_s \Sigma_S V_s^{\mathrm{T}} Y \tag{16}$$

Revisiting Section 3.3, $\tilde{P}_s$ and P_s share the same singular value matrix, which implies that $\tilde{\Sigma}_s = \Sigma_S$. Therefore, based on (16), we can derive (17) as follows.

$$\tilde{U}_s = [X_1, X_2, \dots, X_k] U_s \quad \tilde{V}_s^{\mathrm{T}} = V_s^{\mathrm{T}} Y \tag{17}$$

Since $[X_1, X_2, \dots, X_k]$ is an orthogonal matrix, we obtain (18) through a matrix transformation as follows.

$$U_s = [X_1, X_2, \dots, X_k]^{\mathrm{T}} \tilde{U}_s \tag{18}$$

Based on the above analysis, AS possesses the matrix $[X_1, X_2, \dots, X_k]$ and can recover the original U_s by utilizing the relationship in (18). Therefore, (19) can be obtained as follows. The matrix $\hat{G}_s$ is a dimensionality-reduced representation of the original data G_s, containing its most essential feature information.

$$\hat{G}_s = U_s \Sigma_s \tag{19}$$

4.5 Detecting Malicious Clients

In this section, we propose an unsupervised approach to detect malicious clients in FL by analyzing the Euclidean distance-based distribution of client gradients in a two-dimensional space. Assuming a majority of benign clients, we apply K-means clustering (k=2) to group clients and classify the larger cluster as benign. Gradients identified as malicious are excluded from aggregation in each round, thereby improving global model robustness.

However, this method may misclassify benign clients in the absence of attacks, as K-means still enforces a binary division. To address this, we introduce the CH index to evaluate clustering quality. When malicious clients are present, the gradient distribution shows a clear bimodal pattern, while in attack-free rounds, it is more concentrated. The CH index helps distinguish between these scenarios, enhancing detection robustness against stealthy attacks.

The CH index, also known as the Variance Ratio Criterion (VRC), is used to evaluate the compactness and separability of clustering results. It is defined as follows:

$$CH(k) = \frac{Tr(B_k)}{Tr(W_k)} \cdot \frac{n-k}{k-1} \tag{20}$$

where n denotes the total number of samples, and k represents the number of clusters.

$Tr(B_k)$ is the trace of the between-cluster dispersion matrix, representing the degree of deviation between each cluster center and the overall center. The formula for computing $Tr(B_k)$ is given as follows:

$$B_k = \sum_{i=1}^{k} n_i \|c_i - c\|^2 \tag{21}$$

where n_i denotes the number of samples in the i-th cluster, c is the global center of all samples, and $\|c_i - c\|^2$ is the squared distance between the i-th cluster center and the global center.

$T_r(W_k)$ is the trace of the within-cluster dispersion matrix, representing the degree of deviation between each sample and its corresponding cluster center. The formula for computing $T_r(W_k)$ is given as follows:

$$W_k = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - c_i\|^2 \tag{22}$$

where C_i denotes the i-th cluster, c_i is the center of the i-th cluster, and $x \in C_i$ represents all sample points within the cluster.

This index measures the ratio of between-cluster dispersion to within-cluster dispersion, adjusted by the number of samples and clusters. A higher CH index indicates better clustering performance, meaning that samples are more compact within clusters and more separated between clusters. The complete pseudocode description is provided in Algorithm 2.

Algorithm 2: Detecting Malicious Clients**.**

Input: $\hat{P}_s$: Low-dimensional matrix for K-means clustering detection
Output: $seting\ resulting\ labels$
max $\_ch = 0$
for each class $i \in [1, |O|]$ **do**
    $S_1,\ S_2 \leftarrow$ K-means ($\hat{P}_s$)
    for each vector $j \in S_1 \cup S_2$ **do**
        $B_j \leftarrow$ using (21)
        $W_j \leftarrow$ using (22)
    $CH(i) \leftarrow \frac{Tr(B_k)}{T_r(W_k)} \cdot \frac{n-k}{k-1}$
if max $\_ch < CH(i)$ **then**
    max $\_ch \leftarrow CH(i)$
    $labels \leftarrow$ Divide all clients into two categories and select the best result
return $CH, labels$

In summary, the proposed method can effectively identify and remove malicious participants launching poisoning attacks in FL. By incorporating the CH index as a robustness metric, the approach effectively avoids false positives in scenarios without attacks, further enhancing the security of the system and the reliability of the model.

5 EXPERIMENTS

5.1 Experimental Setup

We conducted a series of experiments using two datasets (CIFAR-10 and MNIST) to evaluate the performance of the proposed PPDPA algorithm. We used a Dirichlet distribution to generate non-IID data for each client. Specifically, the number of FL rounds was set to 100, with a total of 50 clients, and 20 clients participating in training during each round.

This study focuses on two dynamic label-flipping attack modes: single-label flipping and multi-label flipping. Unlike traditional static attacks, these two attack strategies perform label manipulations dynamically in each training round, making the attacks more covert and persistent. As a result, they increase the difficulty of defense and pose new challenges to the security of FL systems. We adopted a comparative experimental approach, evaluating the performance of DPFLA against four classical methods (e.g., FedAvg [25], FLAME [20], FoolsGold [15] and FLDetector [26]). To quantify the impact of label flipping attacks and on the trained model, we employ the following evaluation metrics on the test dataset.

1) Model Accuracy (MA): The proportion of all test samples that are correctly classified, reflecting the overall performance of the global model.

2) Attack Success Rate (ASR): The proportion of source-class samples that are misclassified into the target class (i.e., the incorrect class specified by the attacker). A lower ASR indicates better suppression of label flipping attacks by the defense algorithm.

3) Recognition Accuracy (RA): The proportion of clients correctly identified as benign or malicious. A higher value indicates stronger detection capability of the defense system.

5.2 Model Accuracy of Samples

Single-Label Flipping Attacks

To evaluate the defense capability of PPDPA against single-label flipping attacks, we set the attacker ratios to 10%, 20%, 30%, and 40%, respectively. In this type of attack, the attacker flips the labels of all samples from a specific class to a designated target label.

Fig. 2 presents the accuracy curves of different algorithms on the CIFAR-10 and MNIST datasets over training rounds. PPDPA algorithm consistently achieves the highest accuracy across all scenarios. As the proportion of malicious clients increases, the accuracy of models trained with other algorithms declines significantly, whereas the accuracy drop for PPDPA remains relatively moderate. When the attacker ratio exceeds 20%, the accuracy of the FedAvg algorithm begins to fluctuate noticeably. These experimental results indicate that all evaluated algorithms are capable of effectively defending against single-label flip attacks, with PPDPA demonstrating the strongest defense performance.

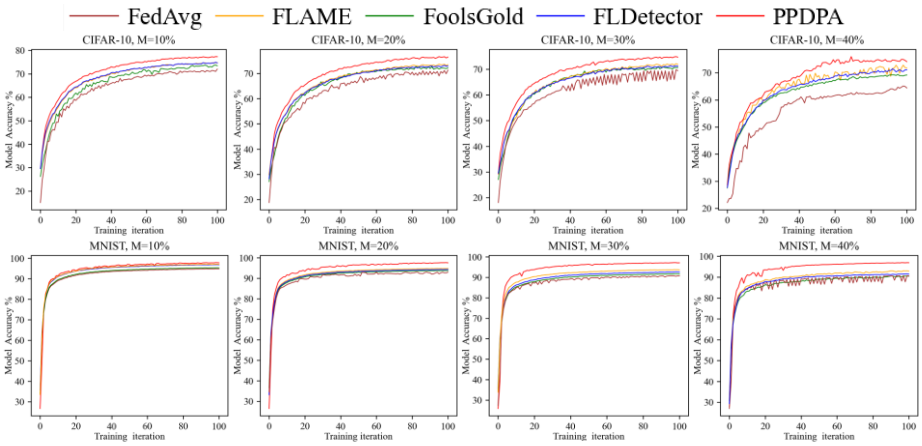

Fig. 2. Model accuracy of different algorithms under single-label flip attacks.

Table 1 compares the main evaluation metrics of various algorithms under single-la bel flip attacks with varying proportions of malicious clients. The PPDPA scheme demonstrates the most outstanding overall defense performance, maintaining a MA of 74.52% and limiting the ASR to 16.2% on the CIFAR-10 dataset even with 40% of participants being malicious. Simultaneously, it achieves a MA of 96.86% and an ASR of only 1.01% on the MNIST dataset, significantly outperforming the other approaches.

Table 1. Experimental metrics of various algorithms under single-label flip attacks.

Algorithm	Dataset	M=10%		M=20%		M=30%		M=40%	
		MA	ASR	MA	ASR	MA	ASR	MA	ASR
FedAvg	CIFAR-10	71.28	26.8	70.31	36.7	68.03	46.4	64.39	71.6
	MNIST	94.80	1.46	92.58	1.68	90.66	2.91	90.37	9.75
FLAME	CIFAR-10	74.64	20.2	73.35	23.8	71.77	25.3	71.48	52.0
	MNIST	97.00	1.46	97.74	1.65	93.70	2.02	92.84	2.34
FoolsGold	CIFAR-10	73.40	15.7	72.04	21.4	70.54	21.8	69.12	36.1
	MNIST	95.29	1.57	93.74	1.68	91.68	2.91	90.39	9.98
FLDetector	CIFAR-10	74.58	14.4	72.82	23.0	71.07	37.6	70.89	52.9
	MNIST	96.79	1.27	94.24	1.32	92.59	2.02	91.41	4.04
PPDPA	CIFAR-10	**77.16**	**12.3**	**76.35**	**12.5**	**74.62**	**13.2**	**74.52**	**16.2**
	MNIST	**97.76**	**0.45**	**97.52**	**0.9**	**97.11**	**0.78**	**96.86**	**1.01**

Multi-Label Flipping Attacks

In the multi-label attack scenario, malicious clients randomly modify the labels of all samples. Fig. 3 presents the model accuracy curves over training rounds for different algorithms under varying proportions of malicious clients on the CIFAR-10 and MNIST datasets. PPDPA consistently maintains the highest accuracy across all attack ratios, indicating its strong robustness against multi-label flipping attacks. In contrast, the FedAvg method is the most sensitive to attacks, with a slightly larger drop in accu-

racy compared to other algorithms. The experimental results demonstrate that all evaluated algorithms are effective in defending against multi-label flipping attacks, with PPDPA exhibiting the strongest defense performance.

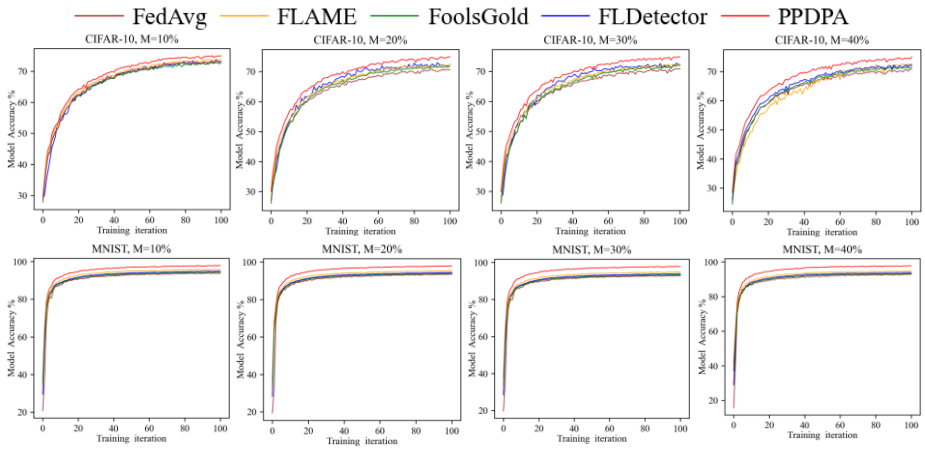

Fig. 3. Model accuracy of different algorithms under multi-label flip attacks.

Table 2 demonstrates that the PPDPA algorithm exhibits superior defensive performance under 40% malicious client attacks. On the CIFAR-10 dataset, PPDPA maintains a model accuracy of 74.50%, significantly outperforming other algorithms, while its attack success rate of 12.8% is substantially lower than FedAvg's 42.2%. On the MNIST dataset, PPDPA maintains its leading advantage with 97.64% accuracy and a remarkably low attack success rate of just 1.32%.

Table 2. Experimental metrics of various algorithms under multi-label flip attacks.

Algorithm	Dataset	M=10%		M=20%		M=30%		M=40%	
		MA	ASR	MA	ASR	MA	ASR	MA	ASR
FedAvg	CIFAR-10	73.01	13.8	70.48	26.3	70.44	32.1	70.31	42.2
	MNIST	93.78	3.26	93.43	4.51	92.79	6.78	92.73	10.4
FLAME	CIFAR-10	73.61	7.3	72.00	13.0	72.14	16.3	72.26	18.2
	MNIST	95.79	1.18	95.35	1.84	94.78	2.32	94.64	2.96
FoolsGold	CIFAR-10	72.50	9.6	71.61	10.8	71.64	13.2	71.26	16.4
	MNIST	94.28	2.67	93.86	4.32	93.29	5.68	93.14	8.89
FLDetector	CIFAR-10	72.92	9.4	72.20	12.2	72.01	16.6	71.84	20.9
	MNIST	94.85	1.43	94.40	2.57	93.85	3.48	93.76	4.28
PPDPA	CIFAR-10	**74.77**	**3.2**	**74.61**	**6.3**	**74.56**	**8.12**	**74.50**	**12.8**
	MNIST	**97.80**	**0.56**	**97.85**	**0.89**	**97.79**	**1.12**	**97.64**	**1.32**

5.3 Recognition Accuracy of Clients

Single-Label Flipping Attacks

Fig. 4 presents the comparison curves of recognition accuracy over training rounds for four defense algorithms under varying proportions of malicious clients.

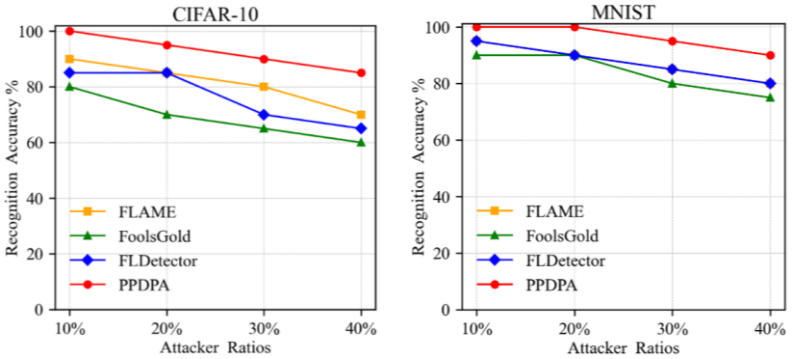

Fig. 4. Recognition accuracy of different algorithms under attacker ratios.

According to Fig. 4, at a 40% proportion of malicious clients, PPDPA achieves the highest detection accuracy on both the CIFAR-10 and MNIST datasets, reaching approximately 85% and 90%, respectively. In comparison, FoolsGold shows the lowest performance, with accuracies of 60% on CIFAR-10 and 70% on MNIST, which are 25% and 20% lower than those of PPDPA.

Multi-Label Flipping Attacks

Fig. 5 presents the comparison curves of recognition accuracy over training rounds for four defense algorithms under varying proportions of malicious clients.

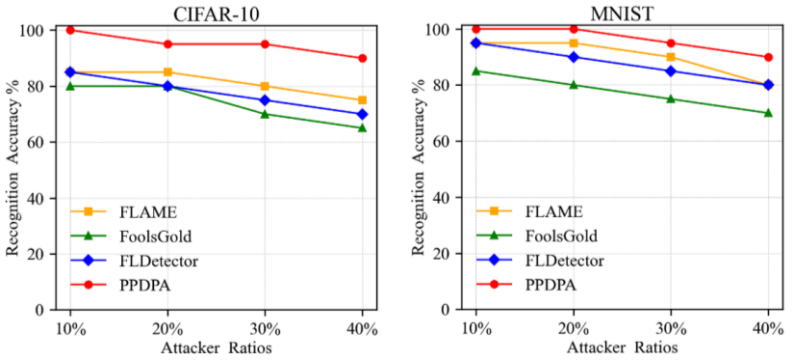

Fig. 5. Recognition accuracy of different algorithms under attacker ratios.

According to Fig. 5, at a 40% proportion of malicious clients, PPDPA performs the best on both the CIFAR-10 and MNIST datasets, achieving an accuracy of around 90% in both cases. In contrast, FoolsGold shows the lowest accuracy, with 65% on CIFAR-10 and 70% on MNIST, which are 15% and 20% lower than PPDPA, respectively.

6 CONCLUSION

This paper proposes the PPDPA framework, an innovative FL defense mechanism that achieves dual optimization in both privacy preservation and poisoning attack detection. The scheme employs gradient masking technology to ensure client data privacy, utilizes SVD for feature extraction and dimensionality reduction, and accurately identifies malicious clients through an enhanced clustering algorithm. Experimental results demonstrate that compared to existing defense approaches, PPDPA exhibits significant advantages in both model accuracy and attack detection rate while maintaining superior robustness.

References

1. Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., Shmatikov, V.: How to backdoor federated learning. In: Proceedings of the International Conference on Artificial Intelligence and Statistics, pp. 2938–2948. PMLR (2020)
2. Jiang, Y., et al.: Model pruning enables efficient federated learning on edge devices. IEEE Trans. Neural Netw. Learn. Syst. **34**(12), 10374–10386 (2023)
3. Rahman, S.A., Tout, H., Ould-Slimane, H., Mourad, A., Talhi, C., Guizani, M.: A survey on federated learning: The journey from centralized to distributed on-site learning and beyond. IEEE Internet Things J. **8**(7), 5476–5497 (2021)
4. Melis, L., Song, C., De Cristofaro, E., Shmatikov, V.: Exploiting unintended feature leakage in collaborative learning. In: Proceedings of the IEEE Symposium on Security and Privacy (SP), pp. 691–706 (2019)
5. Phong, L.T., Aono, Y., Hayashi, T., Wang, L., Moriai, S.: Privacy-preserving deep learning via additively homomorphic encryption. IEEE Trans. Inf. Forensics Security **13**, 1333–1345 (2018)
6. Liu, X., Li, H., Xu, G., Chen, Z., Huang, X., Lu, R.: Privacy enhanced federated learning against poisoning adversaries. IEEE Trans. Inf. Forensics Security **16**, 4574–4588 (2021)
7. 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 USENIX Annual Technical Conference (USENIX ATC 2020), pp. 493–506 (2020)
8. Sanon, S. P., Reddy, R., Lipps, C., Schotten, H. D.: Secure federated learning: An evaluation of homomorphic encrypted network traffic prediction. In: Proceedings of the IEEE CCNC, Las Vegas, NV, USA, pp. 1–6 (2023)
9. Zhong, L., Zhang, L., Xu, L., Wang, L.: MPC-based privacy-preserving serverless federated learning. In: Proceedings of the International Conference on Big Data Analytics and Intelligence Engineering (ICBAIE), Xi'an, China, pp. 493–497 (2022)
10. Sun, Z., Kairouz, P., Suresh, A. T., McMahan, H. B.: Can you really backdoor federated learning? arXiv preprint arXiv:1911.07963 (2019)
11. Chen, X., Yu, H., Jia, X., Yu, X.: APFed: Anti-poisoning attacks in privacy-preserving heterogeneous federated learning. IEEE Trans. Inf. Forensics Security, 18, 5749–5761 (2023)
12. Wang, H., Liu, X., Niu, J., Tang, S.: SVDFed: Enabling communication-efficient federated learning via singular-value-decomposition. In: Proceedings of the IEEE INFOCOM, New York City, NY, USA, pp. 1–10 (2023).

13. Bell, J. H., Bonawitz, K. A., Gascón, A., Lepoint, T., Raykova, M.: Secure single-server aggregation with (poly)logarithmic overhead. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, pp. 1253–1269 (2020)
14. Bonawitz, K., et al.: Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, pp. 1175–1191 (2017)
15. Fung, C., Yoon, C. J., Beschastnikh, I.: The limitations of federated learning in sybil settings. In: Proceedings of the 23rd International Symposium on Research in Attacks, Intrusions and Defenses, pp. 301–316 (2020)
16. Tolpegin, V., Truex, S., Gursoy, M. E., Liu, L.: Data poisoning attacks against federated learning systems. In: Proceedings of the 25th European Symposium on Research in Computer Security, pp. 480–501 (2020)
17. Jebreel, N. M., Domingo-Ferrer, J.: FL-defender: Combating targeted attacks in federated learning. Knowledge-Based Systems, vol. 260, Article no. 110178 (2023)
18. El Mhamdi, E. M., Guerraoui, R., Rouault, S.: The hidden vulnerability of distributed learning in Byzantium. In: Proceedings of the 35th International Conference on Machine Learning (ICML), pp. 3521–3530 (2018)
19. Jebreel, N. M., Domingo-Ferrer, J., Sánchez, D., Blanco-Justicia, A.: Defending against the label-flipping attack in federated learning. arXiv preprint arXiv:2207.01982 (2022)
20. Wu, W., Zhang, Z., Backes, M., Gong, N. Z., Zhang, Y.: FLAME: Taming backdoors in federated learning. In: Proceedings of the 31st USENIX Security Symposium (USENIX Security 2022), pp. 287–304. Springer, Heidelberg (2022)
21. Li, D., Wong, W. E., Wang, W., Yao, Y., Chau, M.: Detection and mitigation of label-flipping attacks in federated learning systems with KPCA and K-means. In: Proceedings of the 8th International Conference on Dependable Systems and Their Applications (DSA 2021), pp. 551–559. Springer, Heidelberg (2021).
22. Sotthiwat, E., Zhen, L., Li, Z., Zhang, C.: Partially encrypted multi party computation for federated learning. In: Proceedings of the IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), pp. 828–835 (2021)
23. Zhao, L., Jiang, J., Feng, B., Wang, Q., Shen, C., Li, Q.: SEAR: Secure and efficient aggregation for Byzantine-robust federated learning. IEEE Trans. Dependable and Secure Computing, **19**(5), 3329–3342 (2022)
24. Sun, Z., Kairouz, P., Suresh, A. T., McMahan, H. B.: Can you really backdoor federated learning? arXiv preprint arXiv:1911.07963 (2019)
25. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B. A.: Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS), PMLR, pp. 1273–1282 (2017)
26. Zhang, Z., Cao, X., Jia, J., Gong, N. Z.: FLDetector: Defending federated learning against model poisoning attacks via detecting malicious clients. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 2545–2555 (2022)
27. Biggio, B., Nelson, B., Laskov, P.: Support vector machines under adversarial label noise. In: Proceedings of the Asian Conference on Machine Learning (ACML), pp. 97–112 (2011)