FedRMA: A Robust Federated Learning Resistant to Multiple Poisoning Attacks

Hongyong Xiao[1], Xutong Mu[1], and Ke Cheng[1]
[1]School of Computer Science and Technology, Xidian University, Xi'an, 710071, China

Federated learning allows clients to collaboratively train models without disclosing local data, yet it faces the threat of poisoning attacks from malicious clients. Existing research has proposed various robust federated learning schemes, but these often consider only a single type of poisoning attack and are inadequate for scenarios where multiple poisoning attacks occur simultaneously. To address this problem, this paper proposes FedRMA, a robust federated learning scheme resistant to multiple poisoning attacks. FedRMA eliminates the need for unrealistic prior knowledge and defends against multiple poisoning attacks by identifying and removing malicious clients. FedRMA adopts the Affinity Propagation clustering algorithm to adaptively partition clients, thereby enhancing its ability to handle multiple poisoning attacks. To mitigate the impact of uncertainty in client data distribution on model selection, FedRMA uses the L-BFGS algorithm to predict the expected global model and uses it to identify malicious clients. We evaluate the performance of FedRMA on the MNIST and CIFAR-10 datasets and compare it with two existing baselines. The experimental results show that FedRMA successfully eliminates the negative impact of multiple poisoning attacks by accurately identifying malicious clients and outperforms the baseline schemes.

***Index Terms*—Federated learning, poisoning attacks, adaptive clustering, robust aggregation, artificial intelligence security.**

I. Introduction

TRADITIONAL machine learning models typically require centralized storage and processing of a large amount of personal data during centralized training, posing a risk of privacy leakage, especially when the data includes sensitive information. With the increasing focus on data security and privacy protection, traditional centralized machine learning methods are insufficient to meet today's demands. To address the dilemma between data isolation and privacy protection, *Federated Learning* (FL) [1], [2] was first proposed by Google in 2016. FL enables distributed clients, such as mobile devices and IoT devices, to collaboratively train a global model without sharing raw training data. In FL, the server is responsible for maintaining the global model, while training data is decentralized and stored across various clients. The basic FL process involves three steps: firstly, the server dispatches the current global model to selected clients; subsequently, each selected client fine-tunes the received global model using its local data and sends the updated model back to the server; ultimately, the server aggregates the received models according to specific rules and updates the global model. FL is widely applied in diverse fields, including finance, security, healthcare, and online recommendation systems.

However, due to the distributed nature of federated learning, *poisoning attacks* [3]–[8] have become an unavoidable security challenge. Specifically, malicious clients may exploit malicious data to train their local models or directly tamper with model parameters. Subsequently, they upload these poisoned models to the server for aggregation, aiming to compromise the integrity of the global model. Chen et al. [9] pointed out that even the presence of a single Byzantine node in the system could lead to the failure of FL model training. Poisoning attacks are typically categorized into *untargeted poisoning attacks* [3]–[5] and *targeted poisoning attacks* [6]–[8]. The former reduces the accuracy of the global model for all test inputs, while the latter only affects the accuracy of the global model for the chosen test data by the attacker, leaving other test data unaffected. For example, in the case of an image classifier, a malicious client might re-label cars with certain stripes in its local training data as birds. By magnifying the sent model updates to the server, the global model is then erroneously predicted to classify cars with stripes as birds. This type of attack is also known as a *backdoor attack*.

To effectively mitigate poisoning attacks in federated learning environments, scholars have proposed various robustness-enhancing schemes. Some schemes enhance the robustness of model training by adjusting aggregation rules, such as Krum [10] based on Euclidean distance and Trimmed-Mean [11] based on statistics. However, these schemes are easily circumvented and often discard a significant number of benign models during execution. Other schemes attempt to identify and remove malicious clients by introducing auxiliary datasets or clustering. However, obtaining auxiliary datasets is usually unfeasible in practical scenarios, limiting their applicability. Meanwhile, clustering-based schemes try to determine the cluster of benign clients based on cluster size, but this method is affected by the uncertainty of client data distribution and is not always accurate. It is worth noting that multiple poisoning attacks may occur simultaneously in a federated learning environment, and existing schemes usually only consider a single type of poisoning attack, which makes them unable to provide sufficient protection in the face of multiple poisoning attacks.

In this study, we propose a robust federated learning scheme, FedRMA, resistant to multiple poisoning attacks. The scheme effectively mitigates multiple poisoning attacks without relying on additional auxiliary datasets. The core idea behind the design of FedRMA is based on the following insights: models generated by clients of different natures

Manuscript received December 31, 2023; revised May 23, 2024. Corresponding author: Ke Cheng (email: chengke@xidian.edu.cn).

exhibit significant differences, and clustering algorithms can effectively partition these models; model updates show predictable patterns, allowing for the estimation of the expected global model using historical information. Specifically, the server first employs the adaptive clustering algorithm Affinity Propagation to partition the received models; then it uses the quasi-Newton method to estimate the expected global model for the current round of training based on the historical information of the global model. Subsequently, the server calculates the mean value of the models within each cluster and compares their similarities with the expected global model, identifying the cluster with the highest similarity as benign and its corresponding clients as benign; other clusters are deemed malicious, and their clients are considered malicious. Lastly, the server only aggregates the models from benign clients to update the global model.

In summary, we make the following contributions:

- We propose FedRMA, a robust federated learning scheme resistant to multiple poisoning attacks. The scheme employs Affinity Propagation clustering and L-BFGS algorithms to accurately identify malicious clients and improve the robustness of the federated learning system. Contrary to earlier schemes, FedRMA is able to defend against multiple poisoning attacks and does not rely on additional auxiliary datasets.
- We evaluate FedRMA on MNIST and CIFAR-10 datasets. The experiments show that FedRMA eliminates the negative impact of multiple poisoning attacks by accurately identifying malicious clients, and outperforms the compared baseline schemes in terms of malicious client detection accuracy and global model performance.

The rest of the paper is organized as follows. Section II presents related work. Section III provides the preliminaries and system model. Section IV describes the FedRMA scheme in detail. Section V presents the experimental evaluation. Finally, Section VI concludes the study. A summary of important symbols appearing in this paper is provided in Table I.

II. Related work

A. Poisoning attacks in federated learning

Poisoning attacks can be classified into *untargeted poisoning attacks* and *targeted poisoning attacks* based on the scope of their impact.

Untargeted poisoning attacks: Untargeted poisoning attacks aim to compromise the global model, reducing its accuracy for unspecified test inputs. Label-flipping Attack [5] and Gaussian Attack [4] are two common types of non-targeted poisoning attacks. In a Label-flipping Attack, the goal is to deceive a machine learning model by changing the labels of samples in the training data. Attackers alter the labels of samples originally belonging to one class to other classes, guiding the model to learn incorrect decision rules. In Gaussian Attack, adversaries seek to degrade the performance of a model by introducing random noise from a Gaussian distribution into the model's weights.

Targeted poisoning attacks: Targeted poisoning attacks, also known as backdoor attacks, aim to compromise the global model in a way that, for any test input with an embedded trigger chosen by the attacker, the model predicts the label selected by the attacker. In a Scaling Attack [6], attackers replicate local training examples on malicious clients, embed triggers into the replicated training inputs, and assign labels chosen by the attacker. The model update is then calculated based on the local training data augmented by such replicated training examples. Additionally, to amplify the impact of model updates, malicious clients further amplify them by a certain factor before reporting them to the server.

B. Existing defense schemes

Currently, various robust aggregation schemes have emerged to tackle poisoning attacks in federated learning. The Krum [10] algorithm, based on Euclidean distance, selects the gradient from the most optimal client in terms of distance among all clients as the global gradient. Multi-Krum [10], an enhancement of Krum, averages the gradients of the c clients with the smallest cumulative distances. Trimmed Mean [11] sorts the gradients of each user at various positions by size, removing the largest and smallest k values, and then calculates the average of the remaining values as the global gradient. Median [11] computes the global gradient based on the median. The Bulyan [12] algorithm cleverly combines the advantages of Multi-Krum and Trimmed Mean. However, these schemes are susceptible to circumvention by malicious clients through the crafty design of models and often discard a significant number of benign models during execution, resulting in an underutilization of the data resources from benign participants.

Zeno, proposed by Xie et al. [13], utilizes auxiliary data provided by each client for detecting and removing malicious models. FLTrust [14] requires the server to collect a small-scale clean dataset for trust scoring of user-local model updates. However, the applicability of these schemes is limited due to the high requirements for collecting a clean dataset in practical scenarios. FoolsGold [15] defends against Sybil attacks by calculating the cosine similarity between gradients submitted by each client to detect malicious updates. FLDetector [16] calculates a suspicious score by computing the Euclidean distance between actual model updates and predicted model updates and performs k-means clustering based on the suspicious score to detect malicious clients. However, such schemes are typically designed with only a single type of poisoning attack in mind and are unable to defend against multiple poisoning attacks. FLAME [17] is currently the only scheme that considers multiple attacks; it categorizes potential backdoor models into three classes based on the direction and magnitude of weight vectors and employs clustering, model pruning, and noise to counter these three types of backdoors. FLAME relies on the size of clusters to identify the benign cluster from the clustering results. However, due to the uncertainty of client data distribution, this method is not always effective.

According to our research, existing studies typically only consider a single type of poisoning attack, making it difficult to implement federated learning that can defend against multiple poisoning attacks. Therefore, this study aims to design a

TABLE I: Summary of Symbols

Symbol	Explanation
n	Total number of clients
i	Client i
$\boldsymbol{w_i}$	Local model of client i
$\boldsymbol{w}$	Global model
D_i	Dataset of client i
$f(D_i; \boldsymbol{w})$	Loss function
$\boldsymbol{g}$	Global gradient
$\boldsymbol{H}$	Hessian matrix
$\triangle \boldsymbol{w}$	Global model change
$\triangle \boldsymbol{g}$	Global gradient change
$\triangle \boldsymbol{W}$	Global model changes in the last N rounds
$\triangle \boldsymbol{G}$	Global gradient changes in the last N rounds
α	Learning rate
T	Total number of training rounds
C	Clustering results
$\boldsymbol{A}$	Cosine distance adjacency matrix
N	Number of iteration information preserved by L-BFGS
p_c	Mean of models in cluster c

scheme to defend against potential multiple poisoning attacks in federated learning.

III. Related Concepts and Problem Formulation

A. Preliminaries

1) Federated Learning

Federated learning [1], [2] is a form of distributed machine learning that enables multiple decentralized participants to collectively train a model, and throughout the training process, user data does not need to leave the local environment. In a typical federated learning architecture, there are multiple clients and an aggregation server. Clients receive the global model, perform local fine-tuning with their respective datasets, and then send the updated model back to the server. The server aggregates the received models based on a specific strategy and updates the global model. Assuming there are n clients denoted as $i \in \{1, ..., n\}$, each client possesses a private dataset D_i. The optimal global model $\boldsymbol{w}^*$ is the solution to the optimization problem:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} f(D_i; \boldsymbol{w}), \tag{1}$$

where $f(D_i; \boldsymbol{w})$ is the loss function of the local training data for client i.

At the beginning of each training round r, the server selects clients to participate in training and sends the current global model $\boldsymbol{w}^r$ to each selected client. The selected clients fine-tune the global model based on local data according to $\boldsymbol{w}_i^{r+1} = \boldsymbol{w}^r - \alpha \boldsymbol{g}_i^r$, where $\boldsymbol{g}_i^r = \nabla f(D_i; \boldsymbol{w}^r)$ is the gradient and α is the learning rate. Subsequently, clients send the obtained models to the aggregation server. The aggregation server updates the global model according to the Equation (2):

$$\boldsymbol{w}^{r+1} = \sum_{i=1}^{n} \frac{1}{n} \boldsymbol{w}_i^{r+1}. \tag{2}$$

2) Affinity Propagation

Affinity Propagation (AP) [18] is a clustering algorithm proposed by Brendan J. Frey and Delbert Dueck in 2007. Unlike traditional clustering algorithms, AP does not require a predetermined number of clusters. Instead, it automatically determines cluster centers based on the *affinity* between data points. The AP algorithm is inspired by network communication principles, treating each data point as a node in a network and updating relationships between nodes through message passing. The core of the algorithm involves two matrices: the Responsibility Matrix and the Availability Matrix. Responsibility Matrix represents the degree to which each data point selects another data point as the cluster center. This matrix, updated iteratively, reflects the relative strengths between data points. Availability Matrix represents the degree to which each data point is chosen as the cluster center by other data points. Similarly updated iteratively, this matrix reflects the adaptability of each data point to others. In each iteration, data points update their *responsibility* and *availability* values based on information from the affinity and availability matrices. After multiple iterations, the algorithm converges to a stable state where each data point is designated as a cluster center or belongs to a certain center.

The advantages of AP clustering include its ability to determine the number of clusters automatically and its insensitivity to initial values. However, due to its higher computational complexity, it is more suitable for medium-sized datasets.

B. System Model

1) Threat Model

In this study, we consider a typical federated learning environment consisting of a central server and multiple clients. In this environment, clients are untrusted, and both benign and malicious clients exist. Benign clients will perform operations according to established protocols, while malicious clients may launch poisoning attacks aimed at degrading the performance of the global model or introducing backdoors. The attacker has all the local background information of the malicious client, including the local dataset, learning rate, loss functions, etc., and may implement poisoning attacks by tampering with data or directly manipulating the model. Multiple poisoning attacks may exist within the system. This study assumes the server is trustworthy and will maintain the global model as required. This study does not address privacy leakage issues resulting from server-initiated membership inference attacks [19]–[21]. It is noteworthy that our research is orthogonal to the privacy-preserving aspects of federated learning.

2) Defense Objectives

To effectively defend against multiple poisoning attacks in federated learning, the following defense objectives need to be achieved:

- *Robustness*: The proposed scheme must be able to detect all malicious clients and eliminate their impact on the global model.
- *Generality*: The proposed defense should make minimal assumptions about the attacks, such as not presupposing the number of concurrent poisoning attacks, and should not rely on auxiliary datasets or any impractical prior knowledge.

IV. FEDRMA

A. Motivation

Models generated by malicious clients are significantly different from those generated by benign clients, so clustering algorithms can effectively partition these models. Furthermore, the model training process is an optimization problem that aims to find the minimum point of the loss function on the training dataset, and this process exhibits predictable patterns. As shown in Equations (3) to (6), firstly perform a second-order Taylor expansion of the loss function $f(D;\boldsymbol{w})$ at $\boldsymbol{w}^{t-1}$, and then through derivatives, assignments and replacement operations, the approximate expression of the gradient of the current iteration round can be obtained, where $\boldsymbol{H}^t$ is the Hessian matrix. According to Equation (6), we can estimate the approximate gradient; the expected global model can be further estimated using the gradient descent method. Finally, by comparing the similarity between the clusters obtained from clustering and the expected global model, we can identify the cluster containing benign clients and thereby locate all malicious clients.

$$\begin{aligned} f(D,\boldsymbol{w}) \approx\ & f(D;\boldsymbol{w}^{t-1}) \\ & + f'(D;\boldsymbol{w}^{t-1})(\boldsymbol{w}-\boldsymbol{w}^{t-1}) \\ & + \frac{1}{2!}f''(D;\boldsymbol{w}^{t-1})(\boldsymbol{w}-\boldsymbol{w}^{t-1})^2 \end{aligned} \tag{3}$$

$\Rightarrow$ Taking the derivative of both sides

$$f'(D;\boldsymbol{w}) \approx f'(D;\boldsymbol{w}^{t-1}) + f''(D;\boldsymbol{w}^{t-1})(\boldsymbol{w}-\boldsymbol{w}^{t-1}) \tag{4}$$

$\Rightarrow$ Assigning $\boldsymbol{w}^t$ to $\boldsymbol{w}$

$$f'(D;\boldsymbol{w}^{t}) \approx f'(D;\boldsymbol{w}^{t-1}) + f''(D;\boldsymbol{w}^{t-1})(\boldsymbol{w}^t-\boldsymbol{w}^{t-1}) \tag{5}$$

$\Rightarrow$ Replacing derivatives with gradients

$$\hat{\boldsymbol{g}}^t \approx \boldsymbol{g}^{t-1} + \boldsymbol{H}^t(\boldsymbol{w}^t-\boldsymbol{w}^{t-1}) \tag{6}$$

B. FedRMA Overview and Design

1) Overview

FedRMA is a robust federated learning scheme designed to defend against multiple poisoning attacks. Its execution process is shown in Algorithm 1. At the beginning of each training round, the server distributes the current global model to all clients. The clients use local data to train the received global model and send the trained new model back to the server. After receiving the local models uploaded from each client, the server uses an adaptive clustering algorithm to group these models and divide the models from different types of clients into different clusters. Subsequently, the server uses the L-BFGS algorithm [22] to approximately calculate the expected global model based on the model's historical update information. Finally, the server compares the similarity between each cluster and the expected global model to filter out benign models for subsequent aggregation operations to update the global model. The core components of FedRMA mainly include adaptive clustering, global model prediction and client Selection. In the remainder of this section, we will describe these three components in detail.

Algorithm 1: The Design of FedRMA

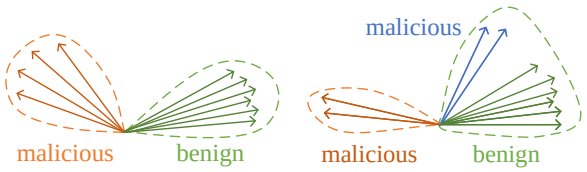
Input: Initial global model $\boldsymbol{w}^0$, number of clients n, number of training rounds T.
Output: Final global model $\boldsymbol{w}^T$.

for t *in* 0 **to** $T-1$ **do**
  Distribute $\boldsymbol{w}^t$ to all clients;
  for i *in* 0 **to** $n-1$ *parallel* **do**
    $\boldsymbol{w}_i^{t+1}$ = ClientUpdate($\boldsymbol{w}^t$);
  Initialize the adjacency matrix $\boldsymbol{A}$;
  for i *in* 0 **to** $n-1$ **do**
    for j *in* 0 **to** $n-1$ **do**
      $\boldsymbol{A}_{ij}$ = CosineDistance($\boldsymbol{w}_i^{t+1}, \boldsymbol{w}_j^{t+1}$);
  C = AffinityPropagation($\boldsymbol{A}$);
  $\boldsymbol{v} = \boldsymbol{w}^t - \boldsymbol{w}^{t-1}$;
  $\hat{\boldsymbol{w}}^{t+1}$ = L-BFGS($\triangle \boldsymbol{W}_t$, $\triangle \boldsymbol{G}_t$, $\boldsymbol{v}$, $\boldsymbol{g}^{t-1}$, $\boldsymbol{w}^t$);
  Initialize the list E;
  for c *in* C **do**
    Compute the mean p_c of the models in c;
    e_c = EuclideanDistance(p_c, $\hat{\boldsymbol{w}}^{t+1}$);
    E.append(e_c);
  $benign = \{i | i \in c, e_c = \min(E)\}$;
  $malicious = \{i | i \notin benign\}$;
  $\boldsymbol{w}^{t+1} = \sum_{i \in benign} \boldsymbol{w}_i^t / |benign|$;
  $\boldsymbol{g}^t = (\boldsymbol{w}^t - \boldsymbol{w}^{t+1})/\alpha$;
return $\boldsymbol{w}^T$;

2) Adaptive Clustering

There are significant differences between models generated by malicious clients and those generated by benign clients, so using clustering algorithms to partition models from different types of clients into different clusters is a widely used strategy. Currently, clustering-based defense schemes are mainly implemented through the K-means algorithm, and the number of clusters is usually set to 2, representing benign clients and malicious clients respectively. As shown in Figure 1(a), the strategy of dividing into two clusters proves effective in the case of a single poisoning attack. However, as FLAME [17] points out, these schemes cannot effectively defend against multiple backdoor attacks where adversaries inject different backdoors into different clients. There is a risk in fixing the number of clusters that malicious models and benign models may be mistakenly assigned to the same cluster, especially if there are large differences between the models generated by different attacks. For example, the federated learning system shown in Figure 1(b) suffers from two different types of poisoning attacks, and there is a significant difference between the two generated malicious models, and this difference is greater than the difference between one of the malicious models and the benign model. If K-means with a fixed number of clusters of 2 is used for clustering, one of the malicious models and the benign model may be classified into the same cluster, which is not in line with expectations. Therefore, it is a

more reasonable strategy to divide clients that launch different attacks into different clusters. FLAME uses the HDBSCAN clustering algorithm that can adaptively determine the number of clusters to divide clients.



(a) Single Poisoning Attack (b) Multiple Poisoning Attacks

Fig. 1: Clustering Results of K-means (k=2) under Different Attack Scenarios.

Inspired by FLAME, we also use an adaptive clustering algorithm to divide clients. Although the HDBSCAN clustering algorithm used by FLAME can adaptively determine the number of clusters, it requires a preset minimum cluster size parameter, and it is not practical to set this parameter in advance. After investigation, we chose Affinity Propagation as the target clustering algorithm because Affinity Propagation does not require mandatory specification of hyperparameters and has shown good performance in experiments. The process of dividing clients using the clustering algorithm is shown in lines 5 to 9 of Algorithm 1. After the server receives the model from clients, it first calculates the cosine distance between each two models to build an adjacency matrix. The cosine distance is calculated as shown in Equation (7). Next, this adjacency matrix is clustered using the Affinity Propagation algorithm. Under ideal circumstances, if there are n different poisoning attacks, $n+1$ different clusters will be obtained. The model generated by each poisoning attack corresponds to one cluster, and the benign model forms another cluster. We chose to perform cluster analysis based on an adjacency matrix constructed from cosine distances rather than directly clustering the received model weights. There are two reasons for this: first, model weights usually have high dimensions and are not suitable for direct clustering; second, cosine distance focuses on measuring the angular difference between model parameter vectors and is not affected by changes in vector length. Therefore, even if the malicious client scales the model parameters, it will not affect the clustering effect.

$$d_{ij} = 1 - \frac{\langle \boldsymbol{w}_i, \boldsymbol{w}_j \rangle}{\|\boldsymbol{w}_i\| \, \|\boldsymbol{w}_j\|} \tag{7}$$

3) Global Model Prediction

In the previous step, we use the Affinity Propagation clustering algorithm to divide models from different types of clients into different clusters. However, this does not immediately reveal whether each cluster contains benign or malicious models. Identifying the nature of each model is crucial to achieving robust federated learning. Existing schemes usually select the largest cluster as the benign cluster based on the assumption that the number of malicious clients is less than half of the total number of clients because in this case there are the most benign clients. However, this cluster size-based approach is not reliable when using adaptive clustering algorithms. Affected by the uncertainty of client data distribution, even models generated by benign clients may be significantly different from each other, and benign models may be divided into multiple clusters. At this time, the cluster formed by the malicious model may become the largest cluster, which will lead to erroneous inferences. Therefore, there is a need to design a new method that is independent of cluster size to identify the nature of clusters.

Algorithm 2: L-BFGS for Predicting Global Model

Input: Global model changes $\triangle \boldsymbol{W}_t$, global gradient changes $\triangle \boldsymbol{G}_t$, $\boldsymbol{v} = \boldsymbol{w}_t - \boldsymbol{w}_{t-1}$, last round global gradient $\boldsymbol{g}^{t-1}$, current round initial global model $\boldsymbol{w}^t$.

Output: Current round expected global model $\hat{\boldsymbol{w}}^{t+1}$.

1 $\boldsymbol{M} = \triangle \boldsymbol{W}_t^T \triangle \boldsymbol{G}_t$;

2 Compute $\boldsymbol{M}$'s diagonal matrix $\boldsymbol{D}_t$ and lower triangular submatrix $\boldsymbol{L}_t$;

3 $\sigma_t = \triangle \boldsymbol{g}_{t-1}^T \triangle \boldsymbol{w}_{t-1} / (\triangle \boldsymbol{w}_{t-1}^T \triangle \boldsymbol{w}_{t-1})$;

4 Compute the Cholesky factorization of $\sigma_t \triangle \boldsymbol{W}_t^T \triangle \boldsymbol{W}_t + \boldsymbol{L}_t \boldsymbol{D}_t^{-1} \boldsymbol{L}_t^T$ to obtain $\boldsymbol{J}_t \boldsymbol{J}_t^t$;

5 $\boldsymbol{p} = \begin{bmatrix} \triangle \boldsymbol{G}_t^T \boldsymbol{v} \\ \sigma_t \triangle \boldsymbol{W}_t^T \boldsymbol{v} \end{bmatrix}$;

6 $\boldsymbol{q} = \begin{bmatrix} -\boldsymbol{D}_t^{1/2} & \boldsymbol{D}_t^{-1/2} \boldsymbol{L}_t^T \\ 0 & \boldsymbol{J}_t^T \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{D}_t^{1/2} & 0 \\ -\boldsymbol{L}_t \boldsymbol{D}_t^{-1/2} & \boldsymbol{J}_t \end{bmatrix}^{-1} \boldsymbol{p}$;

7 $\hat{\boldsymbol{H}}^t \boldsymbol{v} = \sigma_t \boldsymbol{v} - \begin{bmatrix} \triangle \boldsymbol{G}_t & \sigma_t \triangle \boldsymbol{W}_t^T \end{bmatrix} \boldsymbol{q}$;

8 $\hat{\boldsymbol{g}}^t = \boldsymbol{g}^{t-1} + \hat{\boldsymbol{H}}^t \boldsymbol{v}$;

9 $\hat{\boldsymbol{w}}^{t+1} = \boldsymbol{w}^t - \alpha \hat{\boldsymbol{g}}^t$;

10 **return** $\hat{\boldsymbol{w}}^{t+1}$;

According to the previous analysis, the model update shows predictability, so the expected global model that will be obtained in the current round of training can be predicted. The model generated by a benign client is usually closer to the expected global model. The benign cluster can be identified by comparing each cluster's similarity to the expected global model. Therefore, we evaluate whether each cluster is malicious by predicting the expected global model. Specifically, we can estimate the global gradient of this round according to Equation (6), and then further use the gradient descent method to obtain the expected global model. Since the exact calculation of the Hessian matrix is difficult, the quasi-Newton method is often used for approximation in practice. The L-BFGS algorithm [22] can use the information of recent rounds of iterations to calculate the product of the Hessian matrix and any given vector, so this solution uses L-BFGS to calculate the $\boldsymbol{H}^t(\boldsymbol{w}^t - \boldsymbol{w}^{t-1})$ term in Equation (6). For ease of description, we use $\triangle \boldsymbol{w}_t = \boldsymbol{w}^t - \boldsymbol{w}^{t-1}$ to denote the change in the global model at round t, and $\triangle \boldsymbol{g}_t = \boldsymbol{g}^t - \boldsymbol{g}^{t-1}$ to represent the change in the global gradient at round t. $\triangle \boldsymbol{W}_t = \{\triangle \boldsymbol{w}_{t-N}, \triangle \boldsymbol{w}_{t-N+1}, ..., \triangle \boldsymbol{w}_{t-1}\}$ and $\triangle \boldsymbol{G}_t = \{\triangle \boldsymbol{g}_{t-N}, \triangle \boldsymbol{g}_{t-N+1}, ..., \triangle \boldsymbol{g}_{t-1}\}$ denote the changes in the global model and gradient, respectively, over the last N rounds. Algorithm 2 shows the specific implementation of predicting the expected global model based on the L-BFGS algorithm. The algorithm takes inputs $\triangle \boldsymbol{W}_t$, $\triangle \boldsymbol{G}_t$, $\boldsymbol{v}$, last

round global gradient $\boldsymbol{g}^{t-1}$, and current round initial global model $\boldsymbol{w}^t$, and outputs the expected global model $\hat{\boldsymbol{w}}^{t+1}$. In particular, if the approximate Hessian matrix is consistent with the actual Hessian matrix, then the predicted global model will also be consistent with the actual global model.

4) Client Selection

Due to the uncertainty of client data distribution, the traditional cluster size-based evaluation method is not suitable for this scheme. Therefore, the L-BFGS algorithm is used to predict the global model based on historical information. Models from benign clients are generally more similar to the expected global model, while models from malicious clients are significantly different from the expected global model. By comparing the similarity between the models in each cluster and the expected global model, the benign cluster can be identified and then benign models can be screened out. Specifically, the means of the models in each cluster are first calculated, and then the Euclidean distance between these means and the expected global model is calculated. A smaller Euclidean distance indicates that the cluster is closer to the expected global model, so the model in the cluster with the smallest Euclidean distance is considered benign, and the models in the remaining clusters are considered to be from malicious clients. This selection process is detailed in lines 12 to 18 of Algorithm 1. Finally, the server excludes models corresponding to malicious clients and only aggregates models in the benign cluster to update the global model.

V. Experiment

A. Experimental Setup

We implement a prototype system of FedRMA using PyTorch and conduct extensive comparative experiments with state-of-the-art solutions on a machine equipped with an Intel(R) Core(TM) i7-10700 CPU, 32GB RAM, and an NVIDIA GeForce RTX 3060 GPU. Detailed experimental settings will be introduced in the following sections.

1) Datasets and models

We evaluate the performance of FedRMA using image classification tasks on two datasets, MNIST and CIFAR-10. For the MNIST dataset, we use an MLP containing a hidden layer as the network model for classification; for the CIFAR-10 dataset, we use the widely used ResNet18 [23].

We set the total number of clients participating in federated learning to 50, the training rounds to 100, the learning rate to 0.01, and the local training round of each client to 1. In the L-BFGS algorithm, we set the parameter N to 10. To simplify the experiment, all 50 clients participate in training in each round.

2) Attack settings

By default, we randomly select 40% of clients as malicious clients, resulting in 20 malicious clients in each federated learning iteration. We consider two untargeted poisoning attacks: Label-flipping Attack [5] and Gaussian Attack [4], and one targeted poisoning attack: Scaling Attack [6]. In Scaling Attack, we add a trigger 's' to the upper right corner of the image, choose '0' as the target label, and poison only 50% of the data. We conduct experiments in multiple poisoning attacks scenarios, where the aforementioned basic attack methods are combined in pairs. Each malicious client employs only one attack method, with half of the malicious clients employing one attack method and the other half employing a different attack method. In all poisoning attack scenarios, in each round all malicious clients launch attacks. To prevent table overflow, we use abbreviations, where Label-flipping Attack is abbreviated as LF, Gaussian Attack as GS, and Scaling Attack as SC.

3) Defense schemes

In our experiments, we compare two state-of-the-art defense schemes: FLAME [17] and FLDetector [16]. Additionally, we test the results without any defense mechanisms, referred to as NoDefense. NoDefense aggregates models from all clients without any defense against poisoning attacks, and it is included here solely as a reference for other schemes.

4) Evaluation metrics

Before introducing the evaluation metrics, the following definitions are provided: True Positive (TP): A client is malicious and predicted as malicious. False Positive (FP): A client is benign but predicted as malicious. True Negative (TN): A client is benign and predicted as benign. False Negative (FN): A client is malicious but predicted as benign. In this paper, a set of metrics is adopted to evaluate the effectiveness of defense techniques.

Detection Accuracy Rate (DAR) represents the rate of correctly predicted benign and malicious clients among all clients:

$$DAR = \frac{TP + TN}{TP + FP + TN + FN}$$

False Positive Rate (FPR) represents the rate of incorrectly predicting benign clients as malicious among all actual benign clients:

$$FPR = \frac{FP}{TN + FP}$$

False Negative Rate (FNR) represents the rate of incorrectly predicting malicious clients as benign among all actual malicious clients:

$$FNR = \frac{FN}{TP + FN}$$

In addition, we use the Main Task Accuracy (MTA) and Attack Success Rate (ASR) to evaluate the obtained global model. MTA represents the accuracy of the global model on clean datasets, while ASR indicates the probability of being classified as the target label on poisoned datasets. The DAR, FPR and FNR in the experimental results are averaged over all rounds of detection in one experiment, and the MTA and ASR are the results on the final global model. All experimental results are averaged over 3 experiments.

B. Experimental Results

Malicious client detection results: Table II demonstrates the malicious client detection results of our proposed FedRMA with two state-of-the-art schemes, FLAME and FLDetector, in

TABLE II: Malicious Client Detection Accuracy of Different Schemes under Multiple Poisoning Attacks (%)

Dataset	Defense Schemes	SC+LF			SC+GS			LF+GS		
		DAR	FPR	FNR	DAR	FPR	FNR	DAR	FPR	FNR
MNIST	FLAME	97.16	**1.34**	5.08	96.59	2.57	4.68	95.79	2.37	6.97
	FLDetector	98.46	2.13	0.65	79.57	1.22	49.23	76.35	4.89	51.79
	FedRMA	**98.58**	2.26	**0.16**	**99.26**	**1.08**	**0.23**	**99.18**	**1.24**	**0.18**
CIFAR-10	FLAME	89.67	**7.12**	15.13	89.03	**8.23**	15.08	94.94	**4.58**	8.97
	FLDetector	39.56	50.48	75.38	69.73	17.10	50.03	56.39	33.77	58.37
	FedRMA	**93.43**	10.04	**1.35**	**91.91**	12.91	**0.86**	**96.46**	5.50	**0.58**

TABLE III: Global Model Performance of Different Schemes under Multiple Poisoning Attacks (%)

Dataset	Defense Schemes	No Attack	SC+LF		SC+GS		LF+GS
		MTA	MTA	ASR	MTA	ASR	MTA
MNIST	NoDefense	92.13	89.01	97.25	24.19	29.40	28.56
	FLAME	92.13	85.39	0.53	84.52	0.63	84.94
	FLDetector	92.13	91.52	0.11	90.53	73.53	89.53
	FedRMA	**92.13**	**92.26**	**0.05**	**91.68**	**0.29**	**91.16**
CIFAR-10	NoDefense	59.19	52.64	23.83	31.81	14.69	36.21
	FLAME	59.19	54.46	0.62	54.35	0.35	54.46
	FLDetector	59.19	48.53	25.32	52.53	29.42	39.43
	FedRMA	**59.19**	**59.07**	**0.17**	**58.67**	**0.13**	**58.92**

TABLE IV: Experimental Results for Different Malicious Client Ratios under Multiple Poisoning Attacks (%)

Malicious Client Ratios	SC+LF			SC+GS			LF+GS	
	DAR	MTA	ASR	DAR	MTA	ASR	DAR	MTA
10	94.13	92.84	0.35	96.12	92.25	0.52	95.89	92.13
20	95.23	92.96	0.35	97.34	92.84	0.15	97.35	92.23
30	97.73	92.57	0.47	98.97	92.59	0.84	98.34	92.34
40	98.58	92.26	0.05	99.26	91.68	0.29	99.18	91.16

multiple poisoning attack scenarios. Experimental results show that FedRMA's malicious client detection outperforms the other two schemes, with a DAR of over 98% and 91% on the MNIST and CIFAR-10 datasets, respectively. The FLDetector scheme has an FNR of about 50% in some of the experiments, misclassifying half of the malicious clients as benign. This is mainly because FLDetector uses a K-means clustering algorithm and has a fixed number of clusters of 2, resulting in malicious clients of one of the attack types being grouped in the same cluster as benign clients. In contrast, FedRMA and FLAME use an adaptive clustering algorithm to group clients, which avoids this problem and therefore achieves better detection results. The detection accuracy of FLAME is lower than that of FedRMA, which is because FLAME still relies on the cluster size in essence to identify benign clusters. This approach is affected by the uncertainty of the local data distribution of the clients and may sometimes fail to form a benign cluster with a size of at least half of the total number of clients, which affects its detection performance. FedRMA employs the L-BFGS algorithm to estimate the expected global model and selects benign clusters accordingly, which allows FedRMA to perform much better in terms of reliability.

Global model performance: Table III shows the global model performance obtained with four different schemes under multiple poisoning attack scenarios. Experimental results show that the global model performance obtained by FedRMA is comparable to the performance in the no-attack scenario. After 100 rounds of federated learning training on the MNIST and CIFAR-10 datasets, FedRMA achieves an MTA of about 92% and 59%, respectively. In all scenarios containing targeted poisoning attacks, FedRMA's ASR is below 1%. This indicates that FedRMA can effectively detect malicious clients, thus eliminating the impact of poisoning attacks, which is consistent with the malicious client detection results mentioned in the previous section. On the CIFAR-10 dataset, the global model accuracy of FedRMA improves by 4.32% to 4.61% and 6.14% to 19.49% compared to FLAME and FLDetector, respectively.

Impact of the number of malicious clients: To explore the impact of the number of malicious clients on FedRMA, we conduct experiments using a proportion of malicious clients ranging from 10% to 40% of the total number of clients in different attack scenarios on the MNIST dataset, and the results of the experiments are shown in Table IV. In terms of malicious client detection accuracy, the DAR decreases slightly as the percentage of malicious clients decreases, but is still at a high level. In terms of global model performance, the MTA of the resulting global models with different malicious client ratios does not differ much, and the ASR of the scenario containing targeted poisoning attacks is also at a low level. The experimental results show that the proposed scheme is effective under different numbers of malicious clients.

VI. Conclusion

In this paper, we propose FedRMA, a robust federated learning scheme resistant to multiple poisoning attacks. FedRMA employs the adaptive clustering algorithm Affinity Propagation to classify clients and predicts the expected global model via the L-BFGS algorithm as a way to identify malicious clients. FedRMA alleviates the negative impact of poisoning attacks by aggregating only models from benign clients. Our experimental evaluations on MNIST and CIFAR-10 datasets show that FedRMA can effectively defend against multiple poisoning attacks and outperforms the compared baseline schemes. Specifically, compared to FLDetector, the malicious client detection accuracy is improved by 22.18% to 53.87%, and the global model accuracy is improved by 6.14% to 19.49% on the CIFAR-10 dataset.

Acknowledgment

This paper is supported in part by the National Natural Science Foundation of China (No. 62220106004, 61972308); in part by the Major Research Plan of the National Natural Science Foundation of China (No. 92267204); in part by the Basic Research Program (No. JCKY2022XXXX145); in part by the Key Research and Development Program of Shaanxi (No. 2022KXJ-093, 2021ZDLGY07-05); in part by the Innovation Capability Support Program of Shaanxi (No. 2023-CX-TD-02); in part by the Key R&D Program of Shandong Province of China (No. 2023CXPT056); in part by China Postdoctoral Science Foundation (No. 2023M742741), and in part by the Fundamental Research Funds for the Central Universities (No. XJSJ23040, ZDRC2202).

References

[1] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017, pp. 1273-1282.

[3] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1605-1622.

[4] X. Ma, Q. Jiang, M. Shojafar, M. Alazab, S. Kumar and S. Kumari, "DisBezant: secure and robust federated learning against byzantine attack in iot-enabled MTS," in *IEEE Transactions on Intelligent Transportation Systems*, 2023, pp. 2492-2502.

[5] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proceedings of the 29th International Coference on International Conference on Machine Learning (ICML)*, 2012, pp. 1467–1474.

[6] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, 2020, pp. 2938-2948.

[7] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proceedings of the 36th International Conference on Machine Learning*, 2019, pp. 634-643.

[8] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: circumventing defenses for distributed learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[9] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: byzantine gradient descent," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1-25, 2017.

[10] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J Stainer, "Machine learning with adversaries: byzantine tolerant gradient descent," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[11] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: towards optimal statistical rates," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 5650-5659.

[12] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in byzantium," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 3521-3530.

[13] C. Xie, S. Koyejo, I. Gupta, "Zeno: distributed stochastic gradient descent with suspicion-based fault-tolerance," in *Proceedings of the 36th International Conference on Machine Learning*, 2019, pp. 6893-6901.

[14] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "FLTrust: byzantine-robust federated learning via trust bootstrapping," in *28th Annual Network and Distributed System Security Symposium (NDSS)*, 2021.

[15] C. Fung, C. J. M. Yoon, and I. Beschastnikh, "The limitations of federated learning in sybil settings," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, 2020, pp. 301-316.

[16] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "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 (KDD)*, 2022, pp. 2545–2555.

[17] T. D. Nguyen, P. Rieger, H. Chen H. Yalame, H. Möllering, H. Fereidooni , S. Marchal, et al., "FLAME: taming backdoors in federated learning," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 1415–1432.

[18] B. J. Frey, and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972-976, 2007.

[19] R. Shokri, M. Stronati, C. Song and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 3-18.

[20] S. Yeom, I. Giacomelli, M. Fredrikson and S. Jha, "Privacy risk in machine learning: analyzing the connection to overfitting," in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, 2018, pp. 268-282.

[21] M. Nasr, R. Shokri and A. Houmansadr, "Comprehensive privacy analysis of deep learning: passive and active white-box inference attacks against centralized and federated learning," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 739-753.

[22] R. H. Byrd, J. Nocedal, and R. B. Schnabel, "Representations of quasi-Newton matrices and their use in limited memory methods," *Mathematical Programming*, vol. 63, pp. 129–156, 1994.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770-778.

Hongyong Xiao received his B.S. degree from Northeastern University, China, in 2021. He is currently working toward the M.S. degree in computer science at the School of Computer Science and Technology, Xidian University, China. His research interests include federated learning and confidential computing.

Xutong Mu received his B.S. degree from North University of China, China, in 2019. He is currently pursuing the Ph.D degree with the School of Computer Science and Technology, Xidian University, China. His research interests include machine learning security and federated learning.

Ke Cheng is a lecturer in the School of Computer Science and Technology at Xidian University. He received his B.S. and M.S. degrees from Anhui University, Hefei, China, in 2015 and 2018, respectively. He received his Ph.D. degree in computer science and technology from Xidian University in 2022. His research interests include cloud computing security, data security, and privacy protection.