Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.050025

ARTICLE

Byzantine Robust Federated Learning Scheme Based on Backdoor Triggers

Zheng Yang, Ke Gu* and Yiming Zuo

School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha, 410114, China

*Corresponding Author: Ke Gu. Email: gk4572@163.com

Received: 25 January 2024 Accepted: 10 April 2024 Published: 15 May 2024

ABSTRACT

Federated learning is widely used to solve the problem of data decentralization and can provide privacy protection for data owners. However, since multiple participants are required in federated learning, this allows attackers to compromise. Byzantine attacks pose great threats to federated learning. Byzantine attackers upload maliciously created local models to the server to affect the prediction performance and training speed of the global model. To defend against Byzantine attacks, we propose a Byzantine robust federated learning scheme based on backdoor triggers. In our scheme, backdoor triggers are embedded into benign data samples, and then malicious local models can be identified by the server according to its validation dataset. Furthermore, we calculate the adjustment factors of local models according to the parameters of their final layers, which are used to defend against data poisoning-based Byzantine attacks. To further enhance the robustness of our scheme, each local model is weighted and aggregated according to the number of times it is identified as malicious. Relevant experimental data show that our scheme is effective against Byzantine attacks in both independent identically distributed (IID) and non-independent identically distributed (non-IID) scenarios.

KEYWORDS

Federated learning; Byzantine attacks; backdoor triggers

1 Introduction

1.1 Background

With the extensive use of intelligent devices, large amounts of data are generated. Machine learning is proposed and used to process data to derive valid information from them. However, data are often not centralized in real life. Data owners have high requirements for privacy protection. So, it increases the difficulty of collecting training data and leads to "data silos". To address the above issues, federated learning is widely researched [1–4]. Clients are not asked to share their data, only to train locally and upload the local models to the server. After uploading the local models to the server, the model training can be iteratively completed, which provides good protection for the clients' privacy. Therefore, federated learning is being used as an effective method to solve "data silos" and protect the privacy of clients in many application scenarios, such as the Internet of Things, healthcare, finance, and so on. However, in federated learning, multiple clients are required to train the global model, so some attackers can masquerade as benign clients and upload malicious models, which seriously affects the

model training. Then Byzantine attack [5] is a major attack mode that threatens the federated learning system. Byzantine attackers upload their arbitrary parameters to the server for poisoning attacks. Even if only a few attackers are involved, the influence of this attack is very noticeable. Therefore, it is necessary to research the robust federated learning system framework against Byzantine attacks. Currently, the existing detection schemes against Byzantine attacks mainly include detecting outlier values and using median values to detect malicious local models. However, the above detection schemes are only effective when the data of all clients are IID. However, the defense performance of many schemes is often greatly reduced in non-IID scenarios. So, in this paper, we focus on constructing a detection scheme to defend against Byzantine attacks in the above two scenarios.

1.2 Our Contributions

In this paper, we propose a Byzantine robust federated learning scheme based on backdoor triggers, which can defend against Byzantine attacks in both IID and non-IID scenarios. In our scheme, the server first generates a backdoor trigger-label pair and distributes it to all clients. Each client inserts the backdoor trigger-label pair into its local dataset in a certain proportion. Further, each client uses the processed dataset for local training. Then, the server filters out malicious local models by detecting each uploaded local model according to its validation dataset with the backdoor trigger data. To further defend against data poisoning-based Byzantine attacks, the adjustment factors of local models are calculated according to their final layers' parameters. In the global model aggregation, the adjustment factors are used to adjust the weights of local models. Moreover, to further prevent malicious clients from affecting the model training, we propose a new global aggregation method. Our main contributions are described as follows:

- We propose a defense scheme against Byzantine attacks–Byzantine Robust Federated Learning Scheme Based on Backdoor Triggers. In our scheme, all clients embed some backdoor triggers into their data samples, and then the server can calculate the prediction accuracy of each uploaded local model by its validation dataset to detect malicious clients, to prevent some attackers from model poisoning. To further defend against data poisoning-based Byzantine attacks, the adjustment factors of local models are calculated according to their final layers' parameters. In the global model aggregation, the adjustment factors are used to adjust the weights of local models.
- To further enhance the robustness of our scheme, we propose a new global aggregation method. According to the times that each local model is detected as malicious, the weights of local models are reset during global model aggregation. In this way, if one local model is detected as malicious multiple times, its weight is reduced to ensure the robustness of our scheme.
- We test and analyze the performance of our scheme in both IID and non-IID scenarios. Further, we compare our scheme with some existing similar schemes according to the defense performance of these schemes against Byzantine attacks. The experimental results show that the defense performance against Byzantine attacks of our scheme in both two federated learning scenarios is good.

1.3 Organization

The structure of this paper is organized as follows. In Section 2, we introduce the backdoor attacks and some defense methods against Byzantine attacks in federated learning. In Section 3, we describe the system framework and threat model. In Section 4, we provide a Byzantine robust federated learning scheme based on backdoor triggers. In Section 5, we test and analyze the performance of our scheme through some experiments. Further, we draw our conclusions in Section 6.

2 Related Works

In this section, we give a related introduction to backdoor attacks and related defense schemes against Byzantine attacks in federated learning.

2.1 Backdoor Attacks in Federated Learning

In federated learning, the backdoor attackers' main goals are to control the prediction results of the model. Therefore, the attacked model predicts the specific samples as the target class. However, the model's prediction performance on normal samples is unaffected. Backdoor attacks can be achieved through data poisoning attacks and model poisoning attacks. For data poisoning attacks, Nguyen et al. [6] discovered a new poisoning attack scheme. In their scheme, the attackers gradually poison the detection model by poisoning the data a small number of times during the training process. Wang et al. [7] proposed a method to edge case backdoor attacks where these samples rarely appeared in the datasets of benign clients, i.e., they were located at the tail of the normal data distribution. Gong et al. [8] effectively coordinated backdoor attacks on federated learning with multiple local triggers. However, when only data poisoning was used as the attack mode, global aggregation would weaken the influence of most backdoor models. By combining data poisoning and model poisoning, the effectiveness of the attack can be improved. Bagdasaryan et al. [9] first proposed the method of model replacement. Where the attacker attempted to extend the poison update by a judiciously chosen factor, replacing the global model with the poison one. To extend the life of the backdoor, Zhang et al. [10] implanted the backdoor model into coordinates where benign clients are unlikely to update.

2.2 Defense Schemes against Byzantine Attacks in Federated Learning

Many scholars have proposed effective defense schemes against Byzantine attacks in federated learning. For example, in [11], they used a reference dataset and a dual filtering method to defend against Byzantine attacks. Ma et al. [12] proposed a differential privacy federated learning scheme for Byzantine attacks, which has high computational and communication efficiency. To counter the opponents, Data et al. [13] proposed a high-dimensional robust mean estimation method. Rodriguez-Barroso et al. [14] proposed a method to dynamically discard hostile local models to prevent damage to the global learning model. However, the above algorithms are mainly defense methods for regular data and the performance of the algorithms may degrade when the data are irregular. To solve this problem, Tao et al. [15] used a distributed gradient descent algorithm to defend against Byzantine attacks. Zhai et al. [16] combined an adaptive anomaly detection model with data validation to design a credibility assessment to detect Byzantine attacks. Gouissem et al. [17] mathematically proved that in the presence of Byzantine attackers, traditional methods of combining arithmetic mean models would eventually evolve into unstable solutions. Further, they proposed a Byzantine-style elastic training mechanism, which has the characteristics of low complexity and dispersion. Based on reputation, Li et al. [18] proposed a new federated learning scheme under the integrated federated learning architecture to defend against Byzantine attacks. So et al. [19] proposed a single-server Byzantine elastic security aggregation framework for secure federated learning.

However, the above-mentioned backdoor attack schemes focus on how to find potential backdoor attack methods in federated learning, while the defense schemes against Byzantine attacks mentioned above are mostly ineffective in non-IID scenarios. In our scheme, we defend against Byzantine attacks with backdoors, and this method is effective in IID and non-IID scenarios.

3 System Model and Threat Model

In this section, we show our system framework and the corresponding security model, including the attackers' targets and the attackers' capabilities. In Table 1, we describe the main symbols.

Table 1: Symbols and descriptions

Symbols	*Descriptions*
l	*Local model*
w	*Global model*
D	*Dataset*
Tr	*Backdoor trigger*
$\hat{y}$	*Tag with trigger sample*
PA	*Model accuracy*
κ	*Logger that records the number of times that each local model is identified as malicious*
γ	*Adjustment factor*
φ	*Weight factor*

3.1 Federated Learning

In federated learning, the server S aggregates the local models uploaded by the clients to complete the global model training, where each client i has its own local datasets D_i and $\mathcal{C} = \{1, 2, \ldots, C\}$. The whole federated learning process includes the following three steps:

- **System initialization:** The server S initializes the system, which mainly includes initializing the model and setting the training mode.
- **Local training:** In the t-th training round, each client i gets the initial model w^{t-1} from the server S, and then performs its local training with the local dataset D_i to obtain the local model l_i^t, which is then sent to the server S.
- **Global model aggregation:** All clients upload local models to the server S, which then aggregates the global models:

$$w^t = \sum_{i=1}^{C} \frac{|D_i|}{\sum_{j=1}^{C} |D_j|} \cdot l_i^t \tag{1}$$

What's more, federated training will continue until the preset conditions are met.

3.2 System Framework

As shown in Fig. 1, our system framework is based on traditional federated learning. The entities of our system framework include the server and the clients. However, the functions of the two entities are different from the traditional federated learning. The two entities are described as follows:

- **Server:** The server not only receives the local models uploaded by the clients and aggregates the global model, but also has the function of detecting malicious local models. Meanwhile, the server can process small batch datasets.
- **Clients:** The clients are further divided into benign clients and malicious clients (attackers). The benign clients have their local datasets with normal data samples. Malicious clients (such

as nodes $i, j,$ and k in Fig. 1) also have their local datasets where the malicious data samples are randomly generated. Each benign client performs local training through its local dataset to obtain a benign local model in each training round. An attacker (or malicious client) can maliciously create its local model or train its local model through the malicious dataset, then upload the malicious local model to the server to influence the model training.

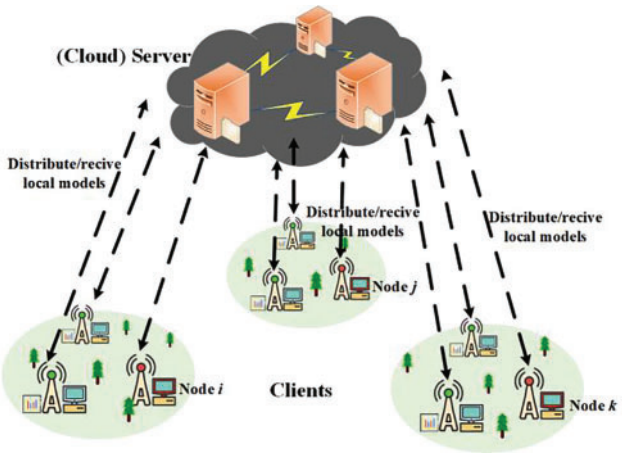


Figure 1: System framework

3.3 Threat Models

We mainly focus on defending against Byzantine attacks. The purpose of Byzantine attacks is to make the trained model unusable and reduce the convergence rate of training, which can be expressed as follows:

$$F(x) = \begin{cases} y, & true\ label; \\ \tilde{y}, & malicious\ target \end{cases} \tag{2}$$

According to the above formula, $\tilde{y} \neq y$ and $\tilde{y}$ is the any label. Further, the attackers can simply create the malicious models through $l_i^t \leftarrow fake(w^t)$, where $fake(\cdot)$ is a model-making function for obtaining malicious models. Fig. 2 shows an example of Byzantine attackers' targets. Further, we assume that the attackers' capabilities are as follows:

- We assume that the attackers have the distribution information of benign data samples, and can create some malicious datasets similar to the distribution of benign datasets. The attackers' targets are to influence the model's performance and minimize the model's prediction accuracy as PA: $minPA(w^t; D_{test})$, where D_{test} is a test dataset.
- We assume that there are multiple attackers (where they may collude to make attacks). They collude to cause a greater deviation between the training of the global model and the training target, thus greatly reducing the global model's prediction accuracy.

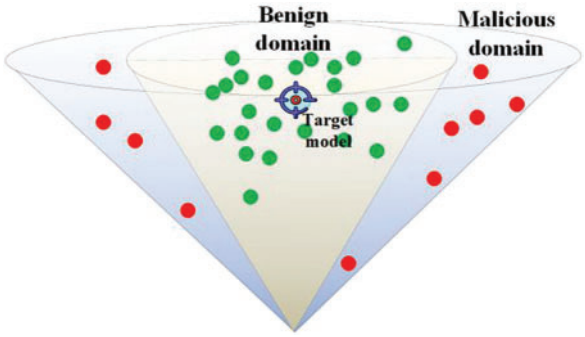


Figure 2: Attack example of Byzantine attacks

4 Byzantine Robust Federated Learning Scheme Based on Backdoor Triggers

In this section, we show a Byzantine robust federated learning scheme based on backdoor triggers in detail. In our scheme, the server S generates a backdoor trigger Tr and the corresponding label $\hat{y}$ before the model training, which are denoted as a trigger-label pair $<Tr, \hat{y}>$. Then the server S distributes the trigger-label pair $<Tr, \hat{y}>$ to all clients who process their local datasetsbased on the trigger-label pair $<Tr, \hat{y}>$ later. Then each client i inserts Tr into D_i in a certain proportion and changes the corresponding label to $\hat{y}$. Finally, the local model is trained through the processed dataset. The Byzantine attackers aim to damage the model performance or reduce the training speed. Therefore, Byzantine attackers can create their malicious local models or use their malicious datasets embedded with backdoor triggers to train their malicious local models. Then attackers upload the malicious local models to the server for global aggregation to achieve Byzantine attacks. Thus, the server needs to defend against model poisoning-based Byzantine attacks by detecting the accuracy of the uploaded local model in its validation dataset containing the trigger. Further, data poisoning-based Byzantine attacks are detected by analyzing the final layers' parameters of local models. In addition, the server records the number of times κ_i that the local model l_i^t is identified as malicious. Then, the server can weight the l_i^t based on κ_i to further ensure the robustness of the model aggregation. Our scheme mainly includes four parts: Initialization, local training, malicious model detection, and global model aggregation. Fig. 3 shows a workflow of our scheme.

4.1 Initialization

In this stage, all clients register to the server, and then the server S initializes the system to obtain the initial global model w^0, and generates the trigger-label pair $<Tr, \hat{y}>$. The server S then sends $<Tr, \hat{y}>$ to all clients, who then process their local datasets after receiving $<Tr, \hat{y}>$ to obtain the datasets that contain the backdoor trigger: $\hat{D}_i \leftarrow Trigger(D_i, Tr, \hat{y}, r)$, where $Trigger(\cdot)$ is an algorithm that inserts a backdoor trigger, r is the percentage of data samples that the clients insert the trigger into their local datasets. In addition, each client uploads its local data sample size s to server S. Then, the server S generates a dataset based on s: $D_{fake} \leftarrow fake(s)$. Then the server$S$embeds the backdoor trigger into D_{fake} to obtain a validation dataset.

$D_{Val} \leftarrow Trigger(D_{fake}, Tr, \hat{y}, 1)$. The test dataset D_{test} is used to test the prediction performance of the global model. The validation dataset D_{Val} is used to detect the malicious local models. What's more, the server S initializes a logger $\kappa = \{\kappa_1, \kappa_2, \ldots, \kappa_C\}$ to record the number of times that local models are identified as malicious, where all elements' initial values in κ are set as 0. The specific implementation is shown in Algorithm 1.

② Initialize: $k_i, w^0, <Tr, \hat{y}>$ and generate D_{Val}	**Stage 1: System Initialization** ① The clients register with the server ③ The server sends $<Tr, \hat{y}>$ to all clients	④ Insert the trigger Tr
	Stage 2: Local training ① The server sends w^{t-1} to all clients ③ Client i sends l_i^t to the server	② Local training
Stage 3: Malicious model detection	① Test the accuracy $PA(l_i^t, D_{Val})$ of each local model ② Get sets L^M and L^B ③ Set the k_i of l_i^t in L^M as $k_i = k_i + 1$ ④ Calculate $base = \frac{\sum_{i \in L^B} ll_i^t}{\lvert L^B \rvert}$ ⑤ Calculate $d_i^t = \sqrt{base^2 - ll_i^{t\,2}}$ ⑥ Get adjustment factor: $\gamma_i^t = \frac{d_i^t}{\sum_{j \in L^B} d_j^t / \lvert L^B \rvert}$	
Stage 4: Global model aggregation	① Calculate the weight factor: $\varphi_i^t = \frac{1/k_i}{\sum_{n \in L^B} 1/k_n}$ ② Aggregate global model: $w^t = \sum_{i \in L^B} \frac{\lvert D_i \rvert}{\sum_{j \in L^B} \lvert D_j \rvert} \cdot \varphi_i^t \cdot l_i^t \cdot \gamma_i^t$	

Figure 3: Workflow of Byzantine robust federated learning scheme based on backdoor triggers

Algorithm 1: System Initialization

1: Procedure:

2: The server S initializes the global model w^0, backdoor triggers and their labels $<Tr, \hat{y}>$, and the parameters required by system training;

3: The server S sends the $<Tr, \hat{y}>$ to all clients;

4: for $i \in C$ **do**

5: The client i processes its local dataset D_i according to $<Tr, \hat{y}>$:
$\hat{D}_i \leftarrow Trigger(D_i, Tr, \hat{y}, r)$;

6: The client i uploads the size of a local data sample s to the server S;

(Continued)

Algorithm 1 (continued)

7: end for

8: The server S randomly generates a dataset according to s: $D_{fake} \leftarrow fake(s)$; and uses $<Tr, \hat{y}>$ to process D_{fake}: $D_{Val} \leftarrow Trigger\left(D_{fake}, Tr, \hat{y}, 1\right)$.

4.2 Local Training

In this stage, each benign client i first needs to download the initial model w^{t-1} of each training round from the server S, and the processed dataset $\hat{D}_i$ is used to get l_i^t, which is the local model trained with $\hat{D}_i$:

$$l_i^t = w^{t-1} - \eta \cdot g_i^t \tag{3}$$

where η is the learning rate, g_i^t is the trained gradient of the client i at round t. For the attackers, their local models are generated by $l_i^t \leftarrow fake(w^t)$. Then all local models are uploaded to the server.

4.3 Malicious Model Detection

In this phase, the server detects all uploaded local models to prevent malicious local models from affecting the global model. For the attackers who make model poisoning-based Byzantine attacks, the malicious local models uploaded by them have lower prediction accuracy for the original data samples and for the data samples containing the backdoor trigger Tr. Therefore, as shown in Fig. 4, server S detects the malicious local models through testing each local model's accuracy on its validation dataset. The advantage of such a method is that malicious local models can be accurately detected even in non-IID and IID scenarios. Then, to further defend against data poisoning-based Byzantine attacks, the adjustment factor of each local model is calculated according to its final layer's parameters, which is used to adjust the weight of each local model for aggregation. Our scheme is described as follows.

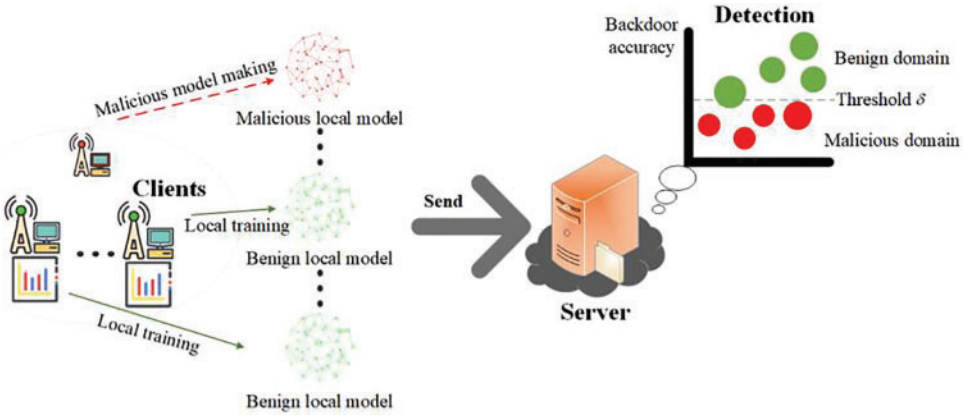


Figure 4: Malicious model detection adjustment factor

4.3.1 Calculation of Prediction Accuracy

After receiving the local models $\{l_1^t, l_2^t, \ldots, l_C^t\}$ uploaded by all clients, the server S verifies all local models' prediction accuracy by its validation dataset D_{Val}. Namely, it inputs the data samples in D_{Val} to $l_i^t (i \in C)$ to obtain the prediction accuracy $PA(l_i^t; D_{Val})$ of each l_i^t. As we described above, for model poisoning-based Byzantine attacks, the target of malicious local models is $minPA(l_i^t; D_{test})$, while the

target of benign local models is $maxPA(l_i^t; D_{test})$. Moreover, the malicious local models also cause the following result as $minPA(l_i^t; D_{Val})$. Therefore, the malicious local models can be identified by detecting the accuracy of the local models on D_{Val}.

4.3.2 Calculation of Adjustment Factor

We consider a local model with $PA(l_i^t; D_{Val}) < \delta$ as malicious, where δ is the threshold for detecting whether a local model is malicious, and the value of δ is proportional to r (the proportion of embedded backdoor triggers set by the server), where $\delta \propto r$. Then a malicious local model set L^M and a benign local model set L^B can be obtained, and the κ_i of the local model in L^M is set as:

$$\kappa_i = \kappa_i + 1 \tag{4}$$

After obtaining the benign local model set L^B and the malicious local model set L^M, we design a detection scheme to further defend against data poisoning-based Byzantine attacks. What's more, all local models in L^B are further tested according to the Euclidean distance between the parameters of their final layers in L^B. The specific process is as follows:

Firstly, the parameters of the final layer ll_i^t are obtained according to the local model in L^B. Secondly, the average value of $ll_i^t (i \in L^B)$ is set as the reference value, which is calculated as $base = \frac{\sum_{i \in L^B} ll_i^t}{||L^B||}$. Finally, the distance between ll_i^t and $base$ is calculated as follows:

$$d_i^t = \sqrt{|base^2 - {ll_i^t}^2|} \tag{5}$$

Based on the distance d_i^t, the adjustment factor γ_i^t is calculated, which is used to adjust the weight of each local model for aggregation:

$$\gamma_i^t = \frac{d_i^t}{\sum_{j \in L^B} d_j^t / ||L^B||} \tag{6}$$

4.4 Global Model Aggregation

In this stage, all local models in L^B are aggregated to get the global model. Since some malicious local models may escape detection in a certain training round, to further enhance the robustness of the global model aggregation, we calculate each local model's weight factor according to the number of times that each local model is identified as a malicious model:

$$\varphi_i^t = \frac{1/\kappa_i}{\sum_{n \in L^B} 1/\kappa_n} \tag{7}$$

From the calculation of weight factors, we can learn that the more times a local model is identified as malicious, the smaller the corresponding weight factor will be. Then we make further adjustments to the global aggregation based on the weight factors and the adjustment factors:

$$w^t = \sum_{i=1}^{C} \frac{|D_i|}{\sum_{j=1}^{C} |D_j|} \cdot \varphi_i^t \cdot \gamma_i^t \cdot l_i^t \tag{8}$$

Our scheme is further described in Algorithm 2. Additionally, we prove that our scheme is effective in defending against Byzantine attacks in both IID and non-IID scenarios by Theorem 4.1.

Algorithm 2: System Initialization

1: Procedure:
2: The whole federated learning system is initialized by the server and the clients;
3: The server S builds a logger that records the number of times each local model is identified as malicious: $\kappa = \{\kappa_1, \kappa_2, \ldots, \kappa_C\}$;
4: The server S sends the $<Tr, \hat{y}>$ to all clients;
5: for $t \in [1, T]$ **do**
6: // T: the total number of training rounds
7: **for** $i \in \mathcal{C}$ **do**
8: The client i downloads the initial model w^{t-1} of t-th training round from the server S and train the local model l_i^t with the dataset $\hat{D}_i$;
9: The client i sends l_i^t to the server S;
10: **end for**
11: The server S constructs two empty sets: malicious local model set L^M and benign local model set L^B;
12: **for** $i \in \mathcal{C}$ **do**
13: Server S tests the accuracy of the local model l_i^t by the validation dataset D_{Val}:$PA(l_i^t; D_{Val})$;
14: **if** $PA(l_i^t; D_{Val}) < \delta$ **do**
15: //δ: an accuracy threshold
16: The server S puts l_i^t into the set L^M of malicious local models, and sets $\kappa_i = \kappa_i + 1$;
17: **else**
18: The server S puts l_i^t into the set L^B of benign local models;
19: **end if**
20: **end for**
21: end for
22: The server S obtains the parameters corresponding to the final layer ll_i^t based on the local model l_i^t in L^B;
23: The average value of ll_i^t is calculated: $base = \dfrac{\sum_{i \in L^B} ll_i^t}{|L^B|}$;
24: **for** $i \in L^B$ **do**
25: The server S calculates the distance between each local model and $base$: $d_i^t = \sqrt{base^2 - ll_i^{t^2}}$;
26: The server S calculates the adjustment factor γ_i^t:$\gamma_i^t = \dfrac{d_i^t}{\sum_{j \in L^B} d_j^t / |L^B|}$;
27: end for
28: for $i \in \mathcal{C}$ **do**
29: The weight factor of the local model l_i^t is calculated: $\varphi_i^t = \dfrac{1/\kappa_i}{\sum_{n \in L^B} 1/\kappa_n}$;
30: end for
31: The global model is aggregated: $w^t = \sum_{i=1}^{C} \dfrac{|D_i|}{\sum_{j=1}^{C} |D_j|} \cdot \varphi_i^t \cdot \gamma_i^t \cdot l_i^t$.

Theorem 4.1. In both IID and non-IID scenarios, our backdoor trigger-based detection scheme is effective against Byzantine attacks.

Proof. When Byzantine attackers perform model poisoning attacks, they upload the malicious models to the server. In both IID and non-IID scenarios, the prediction accuracy of malicious local models on the validation dataset is smaller than a certain threshold. However, benign clients may embed the backdoor triggers into their local datasets in a certain proportion, thus the prediction accuracy of their local models on the validation dataset D_{Val} can still maintain a high value. For some attackers using data poisoning, the final layers of their local models are different from those of the benign local models. Therefore, malicious local models can be easily detected by detecting outliers in the final layers of their local models.

5 Experiments

In this section, we test and analyze the performance of our scheme on some real datasets. The experimental environment is with AMD Ryzen 7 5800H (Radeon graphics card), 3.20 GHz CPU and 16 GB of RAM.

5.1 Experimental Setup

In this section, we show the setup of our experiments in detail, such as datasets, the attackers' capabilities, and the federated learning settings (network model, data distribution, etc.).

5.1.1 Datasets

The handwritten dataset MNIST [20] and the physical dataset CIFAR-10 [21] are used to verify the performance of our scheme. The detailed configurations of the above two datasets are introduced below:

- **MNIST:** It is a widely used dataset of handwritten digits. It is commonly used for testing machine learning algorithms and models. MNIST consists of 70000 28 × 28 pixels grayscale images covering the numbers 0 to 9. Further, the MNIST dataset consists of two subsets. One is the training dataset, which contains 60000 images, and the other is the test dataset, which contains 10000 images.
- **CIFAR-10:** It is commonly used for image recognition. And CIFAR-10 contains color images of 10 categories, such as airplanes, boats, various animals, etc. There are 6000 images per class for a total of 60000 images. Each image is a color image of size 32 × 32 pixels, and each pixel consists of three channels: Red, green, and blue. The images are divided into two datasets. One is the training dataset, which contains 50000 images, and the other is the test dataset, which contains 10000 images.

5.1.2 Attack Setting

In our scheme, we mainly aim to defend against Byzantine attacks. In each training round, the attackers are hidden as benign clients and then upload the Byzantine models to the server. The attackers' goals are to influence the global model's performance and reduce the convergence rate of training.

5.1.3 Federated Setting

In our experiment, we train two different models using the MNIST and CIFAR-10 datasets, respectively. The LeNet-5 model is used for the MNIST dataset. And we use the AlexNet network customized for the CIFAR-10 dataset. In our scenario, we implement a backdoor by inserting a pattern

backdoor trigger into the data sample. It can be seen from Fig. 5 that we set the label of the data sample with the backdoor trigger to a specific label. We verify our scheme's defense performance in IID and non-IID scenarios, respectively. According to the previous work [7,20,21], we adjust the non-IID degree of the data by varying the proportion of image categories assigned to the clients. In this paper, we set the non-IID distribution data based on Dilliclet distribution. Fig. 6 shows an example of dividing the dataset into 10 non-independent and identically distributed datasets with different Dirichlet parameters α. Taking α as 0.3 and 0.5 as an example, we can see that the degree of non-IID when $\alpha = 0.3$ is more obvious.

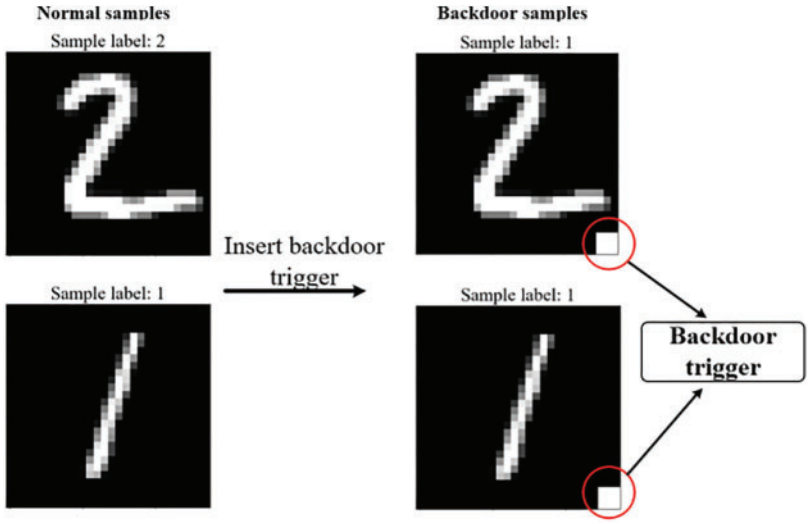
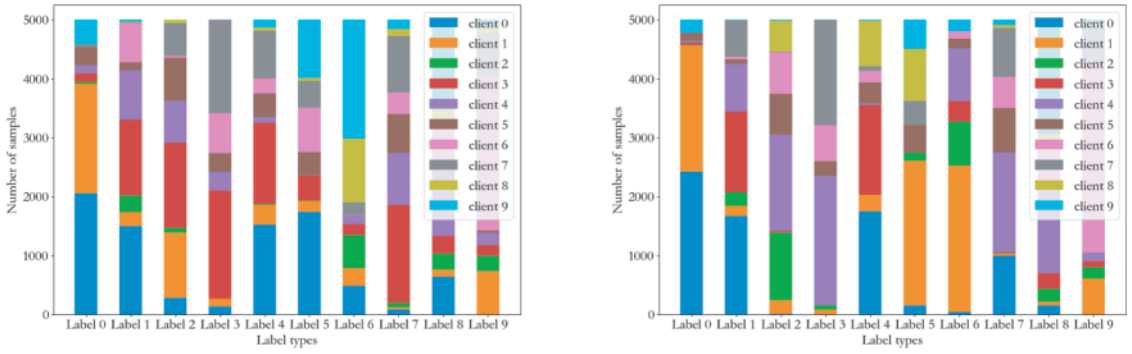


Figure 5: Examples of embedding a backdoor trigger in normal samples



(a) Dirichlet parameter $\alpha = 0.5$ (b) Dirichlet parameter $\alpha = 0.3$

Figure 6: Examples of non-IID data distribution with different α

5.2 Experimental Results

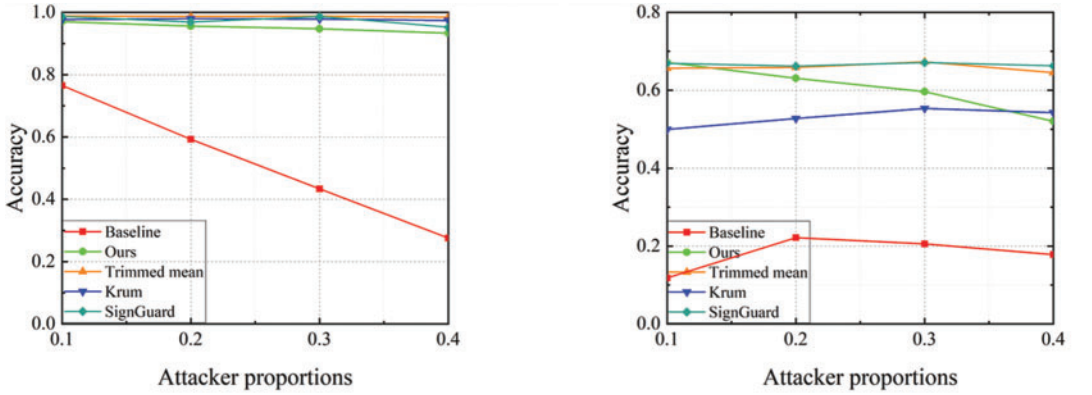
In this section, we evaluate and analyze our experimental results. We use different metrics to evaluate the defense performance of our scheme. Moreover, we also analyze the defense capability of our scheme against Byzantine attacks in both IID and non-IID scenarios. Further, we take unguarded

federated learning as the Baseline, and then compare our scheme with the Krum scheme [22], the Trimmed mean scheme [23], and the SignGuard scheme [24].

5.2.1 Defense Performance in IID Scenarios

In this section, we test our scheme's defense performance in some IID scenarios. Meanwhile, we discuss the influence of different factors on defense performance, such as the attacker proportions and the training rounds.

(1) Attacker Proportions: In general, the higher the attacker proportions, the greater the influence on federated learning. We can learn from Fig. 7 that the defense performance of our scheme is good in both two datasets. We can also learn that as the attacker proportions increase, the Krum scheme, the Trimmed mean scheme, the SignGuard scheme as well as our scheme all show a decreasing trend in global model accuracy.



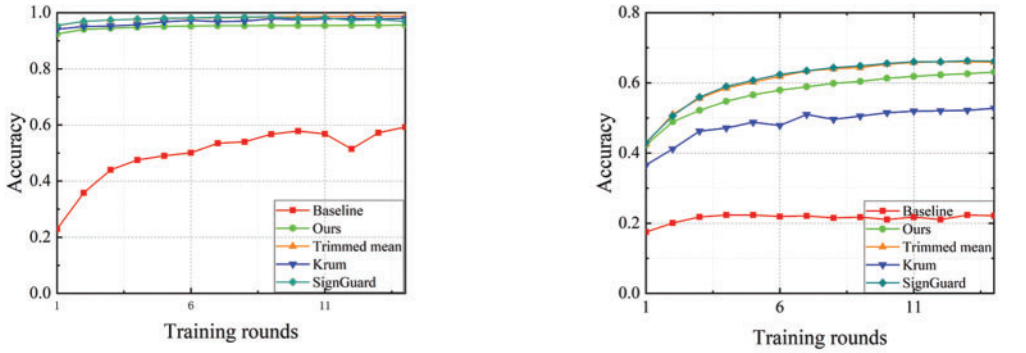
(a) Model accuracy in the MNIST dataset (b) Model accuracy in the CIFAR-10 dataset

Figure 7: Model accuracy under different attacker proportions in IID scenario

(2) Training Rounds: As the number of training rounds increases, the performance of the model will improve. It is because, in the model training process, the model extracts more effective features in each iteration, which makes the prediction performance of the model stronger. In Fig. 8, we show how the accuracy of each scheme changes with the training rounds when the attacker proportion is 20%. We can see that the model accuracy of each scheme improves with the increase of training rounds. However, we can also see that the accuracy improvement of the Baseline is not significant. Moreover, we can learn that in the MNIST datasets, our scheme has no significant difference in defense performance compared to the Krum scheme, the Trimmed mean scheme, and the SignGuard scheme, which also confirms that defense performance against Byzantine attacks of our scheme is good. While in CIFAR-10 datasets, we find that our scheme performs slightly worse than the Trimmed mean scheme and the SignGuard scheme, but better than the Krum scheme.

5.2.2 Defense Performance in Non-IID Scenarios

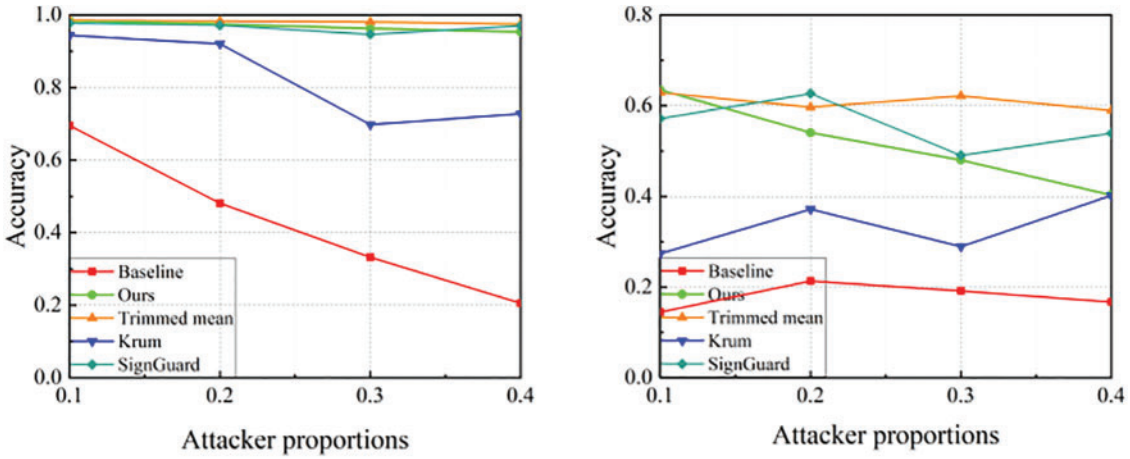
In this section, we test the defense performance of our scheme in non-IID scenarios. Further, we discuss the influence of different factors on defense performance, such as the attacker proportions and the training rounds.



(a) Model accuracy in the MNIST dataset (b) Model accuracy in the CIFAR-10 dataset

Figure 8: Model accuracy under different training rounds in IID scenario

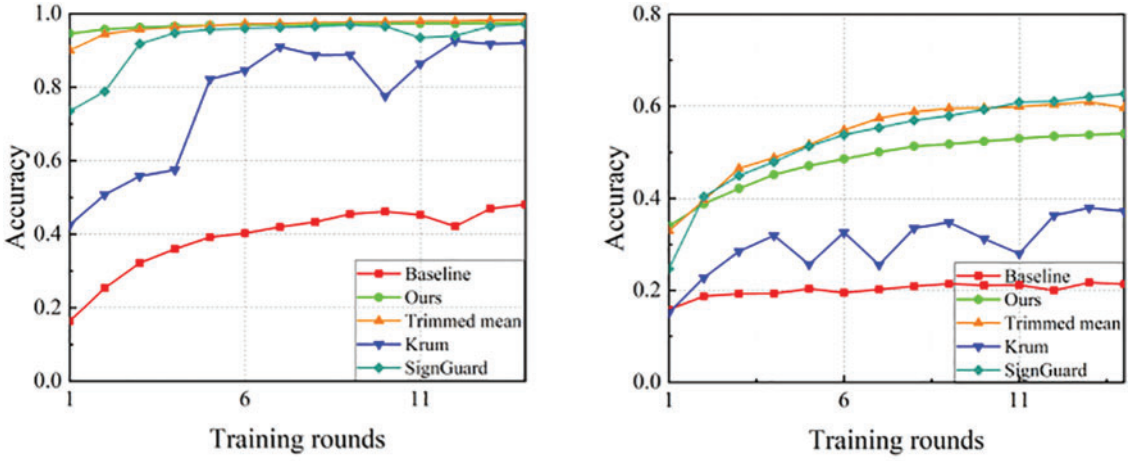
(1) Attacker Proportions: As is well known, when all clients' data are non-IID, the convergence speed and predictive performance of the model will be relatively poor. Fig. 9 shows the influence of attacker proportions on defense performance in non-IID scenarios. We can learn that the defense performance of the Krum scheme is greatly reduced in non-IID scenarios, and there is a slight degradation in the performance of our scheme. However, we can see that the defense performance of the Trimmed mean scheme and the SignGuard scheme are still very good. We analyze that the reason is that the Trimmed mean scheme sets the proportion of attackers as a prior condition, which has a gain in detecting malicious local models.



(a) Model accuracy in the MNIST dataset (b) Model accuracy in the CIFAR-10 dataset

Figure 9: Model accuracy under different attacker proportions in non-IID scenario

(2) Training Rounds: In non-IID scenarios, as the training rounds increase, the performance of the model still improves, but the growth rate is relatively slow. Fig. 10 shows how the accuracy of each scheme changes with the training rounds when the attacker proportion is 20%. We can see that our scheme, the Trimmed mean scheme, and the SignGuard scheme have good defense performance in both MNIST and CIFAR-10 datasets, while the defense performance of the Krum scheme is not very ideal. Especially in the CIFAR-10 dataset, the defense performance of the Krum scheme is significantly degraded.

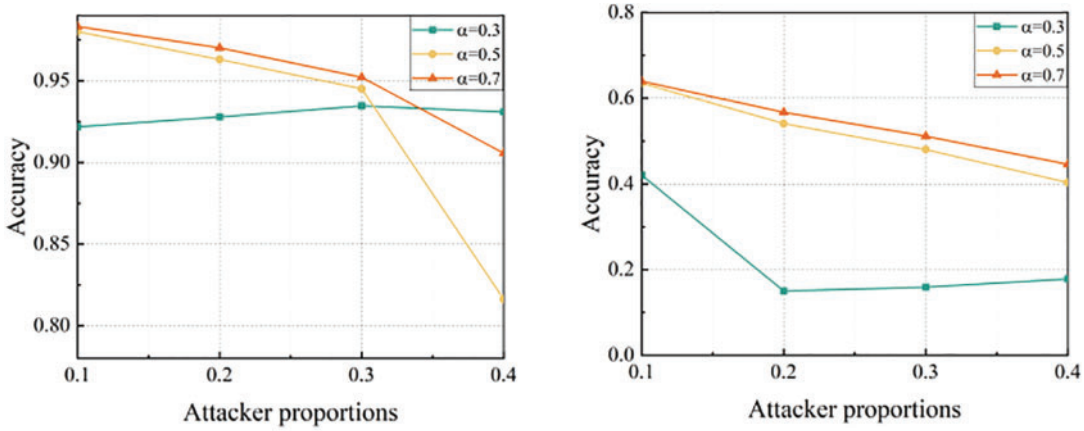


(a) Model accuracy in the MNIST dataset (b) Model accuracy in the CIFAR-10 dataset

Figure 10: Model accuracy under different training rounds in non-IID scenario

(3) Degree of Non-IID: In this subsection, we test the influence of the non-IID degree of data on our scheme's defense performance. Further, we obtain different degrees of non-IID data by changing the Dirichlet parameter α.

a) Influence of the attacker proportions with the parameter α: In Fig. 11, we compare our defense scheme's defense performance against different attacker proportions with different α. It should be noted in advance that the smaller the α, the greater the degree of non-IID data. Therefore, In Fig. 11, we can see the difference between the model prediction accuracy for $\alpha = 0.3, 0.5$, and 0.7. Firstly, as the attacker proportions increase, the model's prediction accuracy tends to decrease. Secondly, we can learn that as the α decreases, namely the non-IID degree increases, the model accuracy also shows a decreasing trend.



(a) Model accuracy in the MNIST dataset (b) Model accuracy in the CIFAR-10 dataset

Figure 11: Model accuracy under different attacker proportions with different α

b) Influence of the training rounds with the parameter α: To demonstrate the influence of different α on the entire training more intuitively, we set the attacker proportion to 20%. Then we

use experiments to verify the changes in model accuracy in each round under different α during the training process. In Fig. 12, we can see that regardless of the α, the model accuracy generally increases with the increase of training rounds. But as we can see, in the CIFAR-10 dataset, when $\alpha = 0.3$, the model's accuracy does not follow this rule. We analyze that it is because the model's original prediction accuracy for the CIFAR-10 dataset is not high, and our scheme also embeds backdoors in the original dataset, resulting in a decrease in effective data.

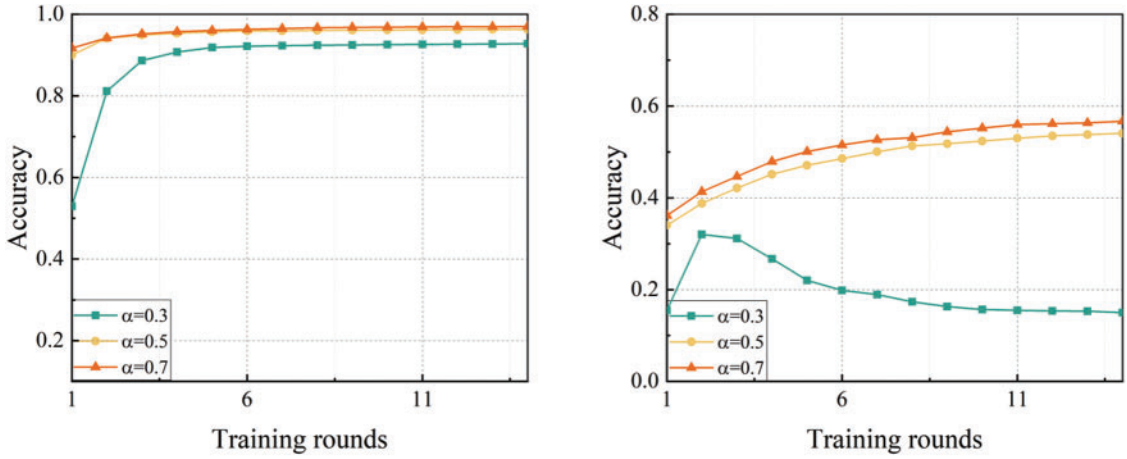


Figure 12: Model accuracy under different training rounds with different α

(4) Influence of the Position of the Trigger: In this section, we analyze the influence on the performance of our scheme when the embedding positions of the trigger are different. We compare the model's prediction accuracy when the trigger is in three different positions. As shown in Fig. 13, we set the trigger at three different positions. In Fig. 14a, we show the model prediction accuracy when the trigger is in three different positions. In Fig. 14b, we show how the model's performance varies with the training rounds when the attacker proportion is 20% and the triggers are at three different positions. Combining the two figures in Figs. 14a and 14b, it is not difficult to find that when the trigger position is in the middle of the picture, that is, position 3, the prediction accuracy of the model decreases slightly, but the decrease is not large.

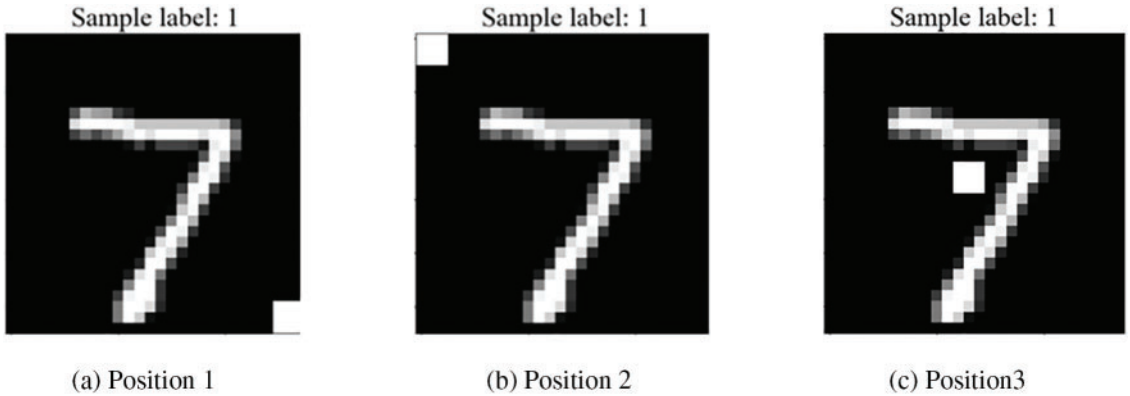
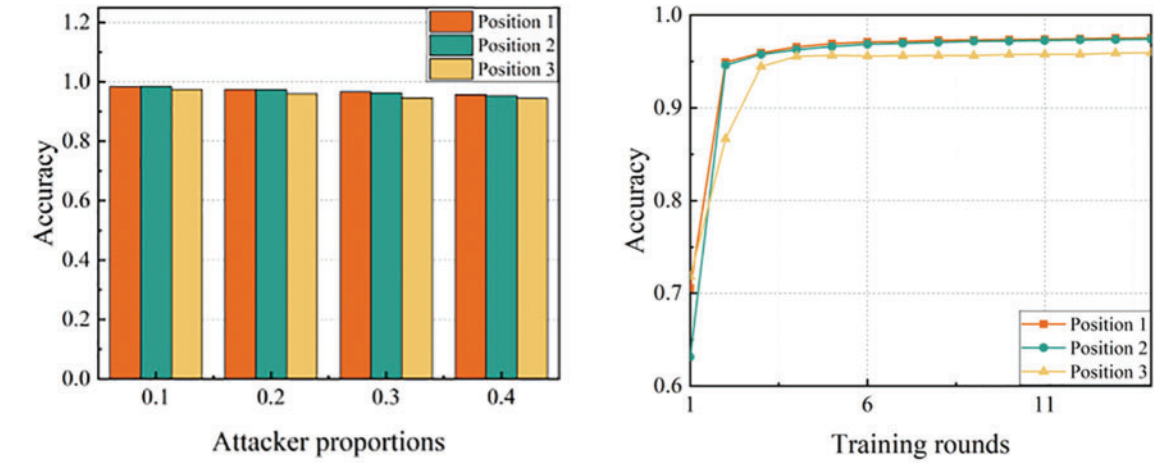


Figure 13: Examples of different trigger positions



(a) Model accuracy with different attacker proportions (b) Model accuracy with training rounds

Figure 14: Model accuracy when the trigger is in different positions

6 Conclusions

Federated learning provides good model training convenience in some applications, but it also provides the chance for some attackers to break the procedure of model training. Among these related attacks, Byzantine attacks pose a great threat to the federated learning system. Therefore, we propose a Byzantine robust federated learning scheme based on backdoor triggers. In our scheme, backdoor triggers are embedded into benign data samples, and then malicious local models can be identified by the server according to its validation dataset. Furthermore, we calculate the adjustment factors of local models according to the parameters of their final layers, which are used to defend against data poisoning-based Byzantine attacks. To further enhance the robustness of our scheme, each local model is weighted and aggregated according to the number of times it is identified as malicious. Relevant experimental data show that our scheme is effective against Byzantine attacks in both IID and non-IID scenarios.

Acknowledgement: The authors wish to express their appreciation to the reviewers and the editor for their helpful suggestions which greatly improved the presentation of this paper.

Funding Statement: This work was supported in part by the National Social Science Foundation of China under Grant 20BTQ058; in part by the Natural Science Foundation of Hunan Province under Grant 2023JJ50033.

Author Contributions: The authors confirm contribution to the paper as follows: Study conception and design: Zheng Yang, Ke Gu; data collection: Yiming Zuo; analysis and interpretation of results: Zheng Yang, Ke Gu, Yiming Zuo; draft manuscript preparation: Zheng Yang, Ke Gu, Yiming Zuo. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The two datasets used in this study can be found in references [20,21], respectively.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Konecný, 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, Oct. 2016.

[2] J. Konecný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," arXiv preprint arXiv:1610.02527, Oct. 2016.

[3] W. Jiang, H. Han, Y. Zhang, and J. Mu, "Federated split learning for sequential data in satellite-terrestrial integrated networks," *Inf. Fusion*, vol. 103, no. 1, pp. 102141, Mar. 2024. doi: 10.1016/j.inffus.2023.102141.

[4] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," arXiv preprint arXiv:1602.05629, Feb. 2016.

[5] B. Kailkhura, S. Brahma, and P. K. Varshney, "Optimal Byzantine attacks on distributed detection in tree-based topologies," presented at Int. Conf. Comput., Netw. Commun., San Diego, CA, USA, 2013, pp. 227–231.

[6] T. D. Nguyen, P. Rieger, M. Miettinen, and A. R. Sadeghi, "Poisoning attacks on federated learning-based iot intrusion detection system," in *Proc. Workshop Decentralized IoT Syst. Secur.*, San Diego, CA, USA, Feb. 2020, pp. 1–7.

[7] H. Wang *et al.*, "Attack of the tails: Yes, you really can backdoor federated learning," presented at Adv. Neural Inf. Proces. Syst., 2020, pp. 16070–16084.

[8] X. Gong, Y. Chen, H. Huang, Y. Liao, S. Wang and Q. Wang, "Coordinated backdoor attacks against federated learning with model-dependent triggers," *IEEE Netw.*, vol. 36, no. 1, pp. 84–90, Jan./Feb. 2022. doi: 10.1109/MNET.011.2000783.

[9] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. AISTATS*, Palermo, Sicily, Italy, 2020, pp. 2938–2948.

[10] Z. Zhang *et al.*, "Neurotoxin: Durable backdoors in federated learning," presented at ICML, Baltimore, MD, USA, 2022, pp. 26429–26446.

[11] Q. Xia, Z. Tao, Q. Li, and S. Chen, "Byzantine tolerant algorithms for federated learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 6, pp. 3172–3183, Nov.–Dec. 2023. doi: 10.1109/TNSE.2023.3251196.

[12] X. Ma, X. Sun, Y. Wu, Z. Liu, X. Chen and C. Dong, "Differentially private byzantine-robust federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 3690–3701, Dec. 2022. doi: 10.1109/TPDS.2022.3167434.

[13] D. Data and S. N. Diggavi, "Byzantine-resilient high-dimensional federated learning," *IEEE Trans. Inf. Theory*, vol. 69, no. 10, pp. 6639–6670, Oct. 2023. doi: 10.1109/TIT.2023.3284427.

[14] N. Rodríguez-Barroso, E. Martínez-Cámara, M. V. Luzón, and F. Herrera, "Dynamic defense against Byzantine poisoning attacks in federated learning," *Future Gener. Comput. Syst.*, vol. 133, no. 15, pp. 1–9, Aug. 2022. doi: 10.1016/j.future.2022.03.003.

[15] Y. M. Tao *et al.*, "Byzantine-resilient federated learning at edge," *IEEE Trans. Comput.*, vol. 72, no. 9, pp. 2600–2614, Sep. 2023. doi: 10.1109/TC.2023.3257510.

[16] K. Zhai, Q. Ren, J. Wang, and C. Yan, "Byzantine-robust federated learning via credibility assessment on non-iid data," *Math. Biosci. Eng.*, vol. 19, no. 2, pp. 1659–1676, 2021. doi: 10.3934/mbe.2022078.

[17] A. Gouissem, K. Abualsaud, E. Yaacoub, T. Khattab, and M. Guizani, "Collaborative byzantine resilient federated learning," *IEEE Internet Things J.*, vol. 10, no. 18, pp. 15887–15899, 15 Sept. 2023. doi: 10.1109/JIOT.2023.3266347.

[18] B. Li, P. Wang, Z. Shao, A. Liu, Y. Jiang and Y. Li, "Defending Byzantine attacks in ensemble federated learning," *Future Gener. Comput. Syst.*, vol. 147, no. 19, pp. 136–148, Oct. 2023. doi: 10.1016/j.future.2023.05.002.

[19] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2168–2181, Jul. 2021. doi: 10.1109/JSAC.2020.3041404.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998. doi: 10.1109/5.726791.

[21] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," in *Technical Report (CIFAR)*, Toronto, Canada: University of Toronto, 2009, pp. 1–58.

[22] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," present at Adv. Neural Inf. Proces. Syst., Long Beach, CA, USA, 2017, pp. 119–129.
[23] D. Yin, Y. Chen, K. Ramchandran, and P. L. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," present at ICML, Stockholm, Sweden, 2018, pp. 5636–5645.
[24] J. Xu, S. L. Huang, L. Song, and T. Lan, "Byzantine-robust federated learning through collaborative malicious gradient filtering," in *Proc. Int. Conf. Distrib. Comput. Syst.*, Bologna, Italy, 2022, pp. 1223–1235.