Using Third-Party Auditor to Help Federated Learning: An Efficient Byzantine-robust Federated Learning

Zhuangzhuang Zhang, Libing Wu, *Member, IEEE*, Debiao He, *Member, IEEE*, Jianxin Li, *Senior Member, IEEE*, Na Lu, and Xuejiang Wei

***Abstract*—Federated Learning (FL), as a distributed machine learning technique, has promise for training models with distributed data in Artificial Intelligence of Things (AIoT). However, FL is vulnerable to Byzantine attacks from diverse participants. While numerous Byzantine-robust FL solutions have been proposed, most of them involve deploying defenses at either the aggregation server or the participant level, significantly impacting the original FL process. Moreover, it will bring extra computational burden to the server or the participant, which is especially unsuitable for the resource-constrained AIoT domain. To resolve the aforementioned concerns, we propose FL-Auditor, a Byzantine-robust FL approach based on public auditing. Its core idea is to use a Third-Party Auditor (TPA) to audit samples from the FL training process, analyzing the trustworthiness of different participants, thereby helping FL obtain a more robust global model. In addition, we also design a lazy update mechanism to reduce the negative impact of sampling audit on the performance of the global model. Extensive experiments have demonstrated the effectiveness of our FL-Auditor in terms of accuracy, robustness against attacks, and flexibility. In particular, compared to the existing method, our FL-Auditor significantly reduces the computation time on the aggregation server by 8×-17×.**

***Index Terms*—Federated learning, Artificial Intelligence of Things, Byzantine-robust, public auditing.**

I. Introduction

With the explosion of Internet of Things (IoT), IoT devices are generating more and more data. How to make good use of these data through data mining techniques has become an urgent research topic. Traditionally, we collect data in advance and then put the data on the server for centralized training [1], [2]. However, this approach has substantial difficulties in satisfying the demands of artificial intelligence of things (AIoT). Data owners may be hesitant to disclose data [3], or data users may not be able to collect data in a timely and efficient manner due to network bandwidth limitations [4], [5]. For example, in smart healthcare, hospitals cannot share data directly with third parties due to the sensitivity of patient data. In general, traditional machine learning methods applied to AIoT encounter issues such as high data communication costs and sensitive data privacy security.

Fortunately, the concept of Federated Learning (FL) [6], as proposed by Google, has surfaced as a promising solution. A typical FL architecture usually involves multiple participants, namely, data owners, and an aggregation server, i.e., service provider. Each participant has its own private dataset, i.e., local dataset, and the service provider serves as a coordinator, collaboratively training a model (i.e., the global model) with the participants. The main idea of FL is to empower data owners to collaboratively train models locally, with the aid of an aggregation server, without compromising the privacy of raw data [7]. Due to its potential application value, many companies employ it for the development of practical applications. For example, Google has developed FL for Android keyboard word prediction [8].

However, recent studies [9]–[14] have demonstrated that FL

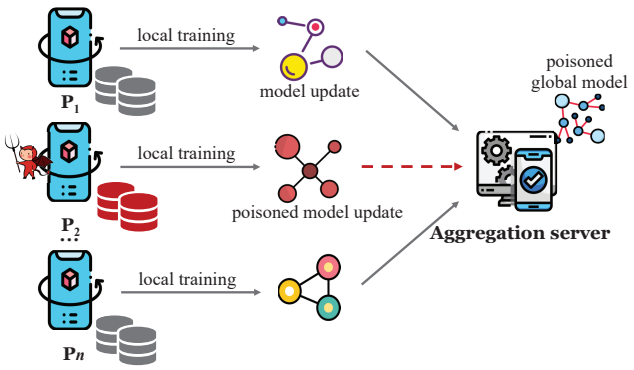


Fig. 1. Overview of Byzantine attacks against FL. Malicious participants can launch Byzantine attacks to poison the global model.

This work was supported by the National Key Research and Development Program of China (No. 2022YFB3104500), National Natural Science Foundation of China (No. U20A20177, U22B2022, 62272348), Key R&D plan of Hubei Province (No.2021BAA025), Industry-University-Research Innovation Fund for Chinese Universities (No. 2021FNA04004), Open Research Fund from Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ)(No.GML-KF-22-07), and Special Fund of Advantageous and Characteristic disciplines (Group) of Hubei Province.

Z. Zhang, L. Wu (Corresponding author) are with the Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China, and also with the Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), ShenZhen, China, e-mail: zhzhuangzhuang@whu.edu.cn, wu@whu.edu.cn.

D. He is with the Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China, e-mail: hedebiao@whu.edu.cn.

J. Li is with the School of Information Technology, Deakin University, Geelong, Australia, e-mail: jianxin.li@deakin.edu.au.

N. Lu and X. Wei are with the School of artificial intelligence, Wuhan Technology and Business University, Wuhan 430065, China, e-mail: lunar1232022@163.com, 2012102110012@whu.edu.cn.

Manuscript received xxxx xxxx, 2023; revised xxxx xxxx.

is susceptible to Byzantine attacks. As depicted in Fig. 1, malicious participants can corrupt the global model by poisoning local training datasets or sending fake model updates [15]. The corrupted global model can make wrong predictions or even predict the target labels chosen by the adversary. Alternatively, resource-constrained participants may have no intention or ability to provide their local model, but only aim to obtain the global model by attending FL training. Due to the above problems, many Byzantine-robust FL approaches [16]–[25] have been proposed. These methods are mainly divided into two categories. The first category [16]–[19], [26] uses statistical knowledge to compare and analyze the model updates uploaded by each participant and then exclude the anomalous updates before updating the global model. These methods usually involve specific security assumptions. The second approach [20], [27] assumes that the server retains a partially clean dataset and then uses that dataset as the basis for identifying anomalous updates and excluding updates with poor performance. These solutions require an aggregation server to maintain a clean dataset and train the model on it.

More specifically, the existing Byzantine robust FL still faces the following problems. Firstly, most approaches [17]–[19], [28], [29] achieve Byzantine robustness by comparing model updates and removing anomalies. This kind of approach has a limited defensive effect and does not work effectively when most participants are malicious. Secondly, existing efforts [30]–[33] focus on deploying defenses at the aggregation server or the participant (i.e., the client), which require intrusive modifications to the server or the participant, and make it difficult to optimize and upgrade the defenses after they are deployed. In addition, these defensive measures impose additional computational overhead on server or participants, which can further exacerbate the resource consumption of IoT nodes in resource-constrained smart IoT scenarios. For example, FLTrust [30] needs to train a root model on the aggregation server as a baseline for Byzantine participant detection, which imposes additional overhead on the server to train the model. Some other approaches [16]–[19], [26], [34] do not need to train a root model on the server, but struggle to achieve similar model performance as FLTrust. Hence, it is crucial to implement a Byzantine-robust FL approach that avoids altering the native workflows of the server or the participants. This prompts us to pose a question: ***Is it feasible to employ a third-party auditor to help the aggregation server in achieving Byzantine-robust model aggregation without modifying the original workflows of the server or clients?***

To answer this question, we propose FL-Auditor, a public audit-based Byzantine-robustness approach for FL. This method leverages a third-party auditor (TPA) to aid the aggregation server in safeguarding against Byzantine attacks (e.g., data poisoning and free-riding attacks), even in scenarios where malicious participants are in the majority. More specifically, in FL-Auditor, the TPA employs a random sampling process to select rounds for audit and assess the trustworthiness of each participant in these sampled rounds. In addition, we propose a *lazy update mechanism* to alleviate potential model performance degradation due to delayed trust updates of participants caused by the sampling-based auditing. Inspired by FLTrust, we also utilize auxiliary validation data (i.e., root dataset) as trust sources. However, unlike it, we propose a public auditing protocol based on a TPA to defend against Byzantine attacks without altering the original operation logic of the aggregation server. The workflow of this protocol consists of the following main steps: The TPA initiates an audit challenge for a specific round of iterations, the aggregation server provides the information for the challenged round, i.e., proofs. Subsequently, the TPA evaluates trust scores for each participant in the challenged round, based on the returned proof and the root dataset. The TPA provides the audit results to the aggregation server, which then updates the global model accordingly. If the current round is not part of the challenge, a lazy update mechanism is used to update the global model. Compared to previous work, it reduces the burden of Byzantine robust rules on aggregation server without compromising model accuracy and supports third-party auditing.

The main contributions of this paper are summarized as follows:

- We propose FL-Auditor, an efficient Byzantine-robust FL method. It can utilize multiple participants to train an accurate global model, even when the majority of participants are malicious.
- We propose a public auditing protocol based on a third-party auditor. This protocol can utilize third-party auditing to assess the trustworthiness of individual participants, thus defending against Byzantine attacks without altering the original operation logic of the aggregation server.
- We conducted extensive experiments on real-world datasets to evaluate the effectiveness of FL-Auditor against various types of Byzantine attacks, and the results demonstrate its efficacy.

The remainder of this paper is organized as follows. In Section II, we describe the related work. In Section III, we give an overview of the system, the threat model, and the design goals. Next, we describe the technical details of the proposed scheme in Section IV. Then, we evaluate our scheme and give experimental results in Section V. Finally, we conclude this paper and discuss future work in Section VI.

II. Related Work

FL, as a distributed training method, is vulnerable to multiple Byzantine attacks [9], [15], [35]–[39]. In this paper, we try to alleviate this problem. Similar to our work, many Byzantine-robust FL schemes have been proposed recently. These works can be roughly divided into two categories. The first kind of Byzantine-robust FL [16]–[19] typically uses statistical knowledge to compare and analyze local model updates for each participant, and then exclude malicious or bad local model updates before global model updates. The second kind of Byzantine-robust FL [20], [27], [40] typically uses a clean dataset to train the baseline model, which is then used as a foundation for comparing and analyzing the participants' local model updates to rule out malicious updates. Next, we present the related work mainly according to these two categories.

The first kind of scheme usually assumes that the poisoned local model updates are geometrically far from benign updates

[23]. For example, Blanchard et al. proposed Krum [16], which selects a local model update among several local model updates that is similar to all other updates as the global model update. Specifically, Krum takes the sum of the *norm* distances of a gradient from other gradients as the score of that gradient, and then selects the lowest score, i.e., the gradient that is similar to most of the gradients, as the aggregated gradient. Similarly, Trimmed Mean and Median [17] remove the largest and smallest values, and then take the average and median of the remaining gradients. In addition, AFA [34] and Auror [41] aim to separate benign and malicious local models as well. However, such schemes only work under specific security assumptions. For example, Auror and Krum assume that the benign participant data distribution is IID, and FoolsGold [19] and AFA assume that the benign participant data is Non-IID. This assumption limits the ability of these defenses. In addition, Krum and Median also assume that the majority of participants are benign. This also makes them ineffective when most of the participants are malicious. Moreover, there has been some work [42], [43] that has the ability to construct more sophisticated attacks to compromise these defenses. Obviously, these schemes are no longer able to perform an effective defense against Byzantine attacks.

The second kind of schemes usually assume that the aggregation server holds a clean validation dataset and that the local model updates of the malicious participant should not be similar to the model updates on the validation dataset. For example, Fang et al. [43] used the loss decrease on the validation dataset to rank the credibility of local model updates. Cao et al. [30] proposed a Byzantine-robust FL called FLTrust, which requires the aggregation server to access a benign dataset, i.e., clean dataset, and computes the similarity of local model updates to the baseline model updates of the aggregation server. In this way, FLTrust can resist Byzantine participants. However, this approach necessitates that the aggregation server also do model training, which places a bigger strain on the aggregation server's computational capability. In addition, BaFFLe [28] lets the participants use their data to evaluate the performance of the aggregation model and thus detect Byzantine attacks. But, BaFFLe has limited effectiveness in resisting Byzantine attacks in scenarios with non-IID data and a small number of participants. This is because BaFFLe cannot distinguish whether the model performance deficiency is due to an attack or a lack of data.

In contrast to the two aforementioned works, a number of recent studies [10], [44] have drawn inspiration from differential privacy (DP) techniques to implement Byzantine attack defenses. Specifically, these works typically clip the weights of model updates to a maximum threshold and add random noise to the weights to dilute/reduce the impact of potentially harmful model updates on the global model [45]. Such schemes do not require any security assumptions, but they also suffer from a fatal drawback. Once a DP is used, it can negatively affect the accuracy of the global model. FLAME [45] uses a low noise level to reduce the negative impact of noise, but it does not completely eliminate the impact of noise on the model's usability. More importantly, none of these solutions support public auditing, as they all rely on aggregation servers to perform complex Byzantine defense tasks. Our scheme aims to address these issues.

TABLE I
COMPARISON OF OUR SOLUTION WITH EXISTING SOLUTIONS

Solution	Number of malicious participants	Non-intrusive	TPA	Security
FedAvg [6]	–	●	○	○
FLAME [45]	<50%	○	○	●
FLTrust [30]	>50%	○	○	●
Trimmed Mean [17]	<50%	○	○	●
Median [17]	<50%	○	○	●
Krum [16]	<50%	○	○	●
LeadFL [24]	<50%	●	○	●
FedVal [33]	>50%	○	○	●
Our FL-Auditor	>50%	●	●	●

* The ● represents yes, ○ represents no.

Table I summarizes the comparison of our FL-Auditor with existing work. As can be seen from Table I, the main difference between our solution and existing Byzantine-robust FL is that our work is a non-intrusive defense approach. It utilizes a TPA (third party auditor) to help the server achieve robust model aggregation. Specifically, existing Byzantine-robust FL often alter the original aggregation rules to make them robust against Byzantine attacks. In contrast, our approach assesses participants solely through the sampling audit method, achieving resilience against Byzantine attacks without modifying the aggregation rules.

III. PROBLEM FORMULATION

In this section, we provide a brief description of the system and formalize the threats that the system faces, as well as the goals that the system aims to achieve.

A. System Overview

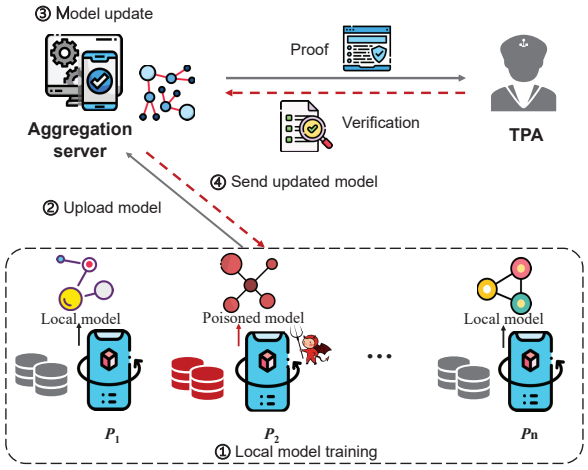


Fig. 2. High-level description of FL-Auditor. TPA audits the FL training process, mitigating the global model's negative impact from Byzantine participants.

As shown in Fig. 2, our FL-Auditor mainly consists of: (i) Aggregation server $\mathcal{S}$ (ii) Participant $\mathcal{P}$ (iii) Third-party Auditor TPA. We assume that there are n participants $\mathcal{P}_i$ $(i = 1, ..., n)$ connected to an aggregation server $\mathcal{S}$ to participate in FL training, where each participant has a local dataset D_i. In

particular, the TPA is in charge of auditing the training process to ensure the correctness of the FL process. The details are described below.

- Aggregation server $\mathcal{S}$: The $\mathcal{S}$ is the trusted entity in this system, which is responsible for distributing models and aggregating local models.
- Participant $\mathcal{P}_i$: The $\mathcal{P}_i$ is the training participant in this system, which owns the dataset and is willing to perform a collaborative FL training task with $\mathcal{S}$'s help. However, a significant fraction of $\mathcal{P}$ may be malicious, and they can construct malicious local model updates in any iteration of the FL training process and then send them to the aggregation server.
- TPA: TPA is a third-party auditor with a clean dataset responsible for auditing the FL training process and mitigating the negative impact of Byzantine participants on local model aggregation.

B. Threat Model

In this paper, we make the assumption that the adversary has the ability to control a majority of the participants to launch an attack on the FL training process. However, the adversary cannot control the aggregation server and the TPA. In other words, both the aggregation server and the TPA are trusted parties, and they perform their computational tasks honestly. This is also realistic because aggregation server and the TPA are usually secure entities controlled by the service provider. Specifically, the adversary can use the malicious participant to send arbitrary model update parameters in order to obstruct or disrupt the FL training process after tapping the participant. In particular, we assume that the malicious participants will use the following attack types:

- Free-riders. Free-riders have no desire to share local model updates due to their data privacy and computational power concerns, but would like to access the FL training process to get the global model. They typically upload randomly generated local model updates.
- Label-flipping. Label-flipping is a targeted poisoning attack that can change the target classification of training samples. Specifically, we use $t = M - 1 - c$ to change the training sample labels from c to t, where M is the total number of labels and c is the label.
- Sign-flipping. Sign-flipping is an untargeted poisoning attack that flips the sign of an upload intermediate parameter. Specifically, a malicious participant would alter the update direction of the global model by flipping the element signs of the model parameters it uploads.

C. Design Goals

In order to achieve efficient FL and resist the various Byzantine attacks mentioned above, the following security and accuracy requirements should be met by our FL-Auditor.

- Accuracy. In the setting without adversaries, the global model's prediction accuracy learned by FL-Auditor should be nearly identical to that of the global model learned by FedAvg. Intuitively, the security mechanisms of FL-Auditor should not have a significant detrimental influence on the accuracy of the global model.
- Security. The FL-Auditor should guarantee the global model's accuracy. In adversary environment, FL-Auditor should be resistant to Byzantine attacks (i.e., free-riders, label-flipping, and sign-flipping attacks), where the global model can make correct predictions.

IV. FL-AUDITOR

In this section, we first briefly describe the workflow of FL-Auditor, and then give a detailed description of the specific technical details.

A. System Workflow

The FL-Auditor helps the aggregation server in achieving Byzantine-robust FL by introducing TPA, avoiding modifications to the original computation logic of the aggregation server or participants. The workflow of FL-Auditor is illustrated in Algorithm 1. Specifically, our system consists of four steps:

- Step ①: Initialization. The aggregation server $\mathcal{S}$ and the participants $\mathcal{P}$ initialize the model parameters, and the TPA obtains the auxiliary validation dataset.
- Step ②: LocalUpdate. Each participant $\mathcal{P}_i$ trains a local model using their local dataset and then sends the model update to the aggregation server.
- Step ③: LocUpdateAudit. In this step, we employ the introduced TPA in our proposed FL-Auditor to randomly sample audits on the aggregation server, assisting in achieving robust model aggregation. Specifically, the TPA performs sampling on the global model update process, audits the selected update processes using auxiliary validation datasets, and then sends the audit results to the aggregation server.
- Step ④: Aggregation. In this step, the aggregation server $\mathcal{S}$ performs local model updates aggregation based on the audit results obtained from the TPA. In addition, it leverages a *lazy update mechanism* to minimize the adverse effects on model performance caused by delayed updates resulting from the sampling audits.

Finally, each participant returns to step ② and starts a new iteration until the model converges or reaches the maximum number of training rounds.

B. System Design

1) LocalUpdate: Assuming that each participant $\mathcal{P}_i (i = 1, ..., n)$ possesses a local dataset D_i and the aggregation server $\mathcal{S}$ has the initial global model parameters w_0. After initialization, each participant receives the initial model parameters w_0 from the aggregation server $\mathcal{S}$. Then, the participant $\mathcal{P}_i$ begins to perform local model updates using its local dataset D_i. Specifically, each participant $\mathcal{P}_i \in \mathcal{P}$ learns a local model w_i on its local dataset D_i by the stochastic gradient descent (SGD) algorithm. During each round of training, participants iterate over their local dataset for multiple epochs. More specifically, in the k-th iteration, the participant

Algorithm 1 FL-Auditor.

Input: Total number of participant n, total number of communication rounds R, total number of local epochs E, learning rate η, sliding window size m.

Output: global model w_0.

initialize the global model w_0
// **LocalUpdate.**
for $r \in \{1, 2, 3, ..., R\}$ **do**
  for $i \in \{1, 2, 3, ..., n\}$ **in parallel do**
    $g_i^r \leftarrow$ **ClientUpdate**(i, w_0^r)
    Send g_i^r to the server.
  end for
end for
// **LocUpdateAudit.**
TPA randomly selects the rounds to be challenged i.e. $\{r_1, r_2, ..., r_i, ..., r_s\}$.
TPA sends a challenge $\mathsf{cha} = \{r_1, r_2, ..., r_i, ..., r_s\}$ to server.
Server $\mathcal{S}$ sends the $\mathsf{proof} = \{M^r, g_i^r, w_0^r, p^r\}(i \in \{1, 2, ..., n\})$ to the TPA.
for $e \in \{1, 2, ..., E\}$ **do**
  for batch $b \in D_{eva}$ **do**
    $w_{root}^{r+1} \leftarrow w_{root}^r - \eta \bigtriangledown \mathcal{L}(w_0^r, b)$
  end for
end for
$g_{root}^r \leftarrow w_{root}^r - w_0^r$
for $i \in \{1, 2, 3, ..., n\}$ **do**
  $cos_i^r = \frac{\langle g_{root}^r, g_i^r \rangle}{\|g_{root}^r\| \cdot \|g_i^r\|}$
  $TS_i^r = ReLU(cos_i^r)$
  $\alpha_i = \frac{\|g_{root}^r\|}{\|g_i^r\|}$
end for
TPA sends $\mathsf{result} = (\alpha^r, TS^r)$ to the $\mathcal{S}$, where $\alpha^r = \{\alpha_1^r, \alpha_2^r, ..., \alpha_n^r\}$, $TS^r = \{TS_1^r, TS_2^r, ..., TS_n^r\}$
// **Aggregation.**
if $r \in cha$ **then**
  for $i \in \{1, 2, ..., n\}$ **do**
    update TS_i^r, α_i^r
    $\lambda_i = 1$
  end for
else
  $TS_i^r \leftarrow \frac{1}{m} \sum_{r-m}^{r-1} TS_i^r$
  Update $\lambda_i = \frac{\left\|g_i^{old}\right\|}{\left\|g_i^{new}\right\|}$ for each participant
end if
$g^r \leftarrow \frac{1}{\sum_{i=1}^n TS_i^r \cdot \lambda_i^r \cdot \alpha_i^r} \sum_{i=1}^n TS_i^r \cdot \lambda_i^r \cdot \alpha_i^r \cdot g_i^r$
$w_0^{r+1} \leftarrow w_0^r - \eta \cdot g^r$
// **Client Update Function**
ClientUpdate(i, w) :
$w_i \leftarrow w_0$
for $e \in \{1, 2, ..., E\}$ **do**
  for $batch\ b \in D_i$ **do**
    $w_i \leftarrow w_i - \eta \bigtriangledown \mathcal{L}(w, b)$
  end for
end for
return $w_i - w$ to server

$\mathcal{P}_i$ initializes the local model w_i^k with the global model w_0^k after receiving it and computes $g_i^k = \bigtriangledown \mathcal{L}(w_0^k, D_i)$. Then, the participant $\mathcal{P}_i$ sends g_i^k to the aggregation server $\mathcal{S}$.

2) LocUpdateAudit: To achieve efficient Byzantine robustness, we propose a public audit protocol. The protocol audits the training process and returns the audit results to the aggregation server, thus facilitating the aggregation of local models by the aggregation server based on the audit results. Specifically, it mainly includes the following three steps:

Step I: Random Sampling Challenge. The adversary may compromise the participant anytime and launch a Byzantine attack on the FL training process. However, auditing all training processes directly incurs a significant computational overhead. Therefore, we propose a sampling-based public audit approach to mitigate the attacks while ensuring efficiency. Intuitively, TPA monitors the training process and randomly selects a portion of the process for participant behavior audits. Specifically, the TPA sends a challenge $\mathsf{cha} = \{r_1, r_2, ..., r_i, ..., r_s\}$ to the aggregation server $\mathcal{S}$, where r_i denotes the selected rounds and s denotes the total number of selected rounds. After receiving the challenge cha, the aggregation server $\mathcal{S}$ sends the $\mathsf{proof} = \{M^r, g_i^r, w_0^r, p^r\}$ of the selected rounds to the TPA, where M^r denotes the ML model, $g_i^r(i \in \{1, 2, ..., n\})$ denotes the local model update of the participant P_i, w_0^r denotes the global model parameters, and p^r denotes the hyper-parameter.

Step II: Model Update Audit. After receiving the proof, TPA verifies the proof and then gets the trust scores TS of the different participants in that round. Intuitively, TPA trains a root model on the auxiliary validation dataset D_{eva} with the received global model w_0^r, and compares the similarity of the root model with the local model as the trust score TS_i^r of participant P_i. This is due to the fact that model updates of benign participants are often similar in FL. In particular, a greater similarity between a specific local model update and the root model update implies a greater contribution of the local model to the global model. Here, the *cosine similarity* is employed to compute the similarity between the two model updates. Specifically, after TPA gets the proof, it trains the root model $w_{root}^r \leftarrow (M^r, D_{eva}, w_0^r, p^r)$, then computes the updates of the root model w_{root}^r to obtain g_{root}^r, and finally it computes the cosine similarity cos_i^r between g_{root}^r and g_i^r by

$$cos_i^r = \frac{\langle g_{root}^r, g_i^r \rangle}{\|g_{root}^r\| \cdot \|g_i^r\|}. \tag{1}$$

However, the cosine similarity computation result may be negative. Specifically, an adversary may manipulate malicious participants, causing the local model to update in the opposite direction to the root model, thereby significantly affecting the global model. We use the *ReLU* function to clip the cos value to solve this problem. Formally, we define $TS_i^r = ReLU(cos_i^r)$, where

$$ReLU(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}, \tag{2}$$

and TS_i^r is the trust score of P_i in the rth round. After clipping, we can remove this type of malicious update.

Step III: Magnitude Normalization and Return Results. Clipping removes models with abnormal update directions.

However, adversaries can also scale the intermediate parameters uploaded to the server. Therefore, we must limit the harmful impact of update magnitude on the global model. To do so, we normalize the updates of each local model to the same magnitude as the root model using a method similar to [30]. Specifically, TPA computes the normalization factor α_i^r of each participant's local model update by $\alpha_i^r = \frac{\|g_{root}^r\|}{\|g_i^r\|}$, where α_i^r is the *normalization coefficient* of the i-th participant in the current round r and $\|\cdot\|$ denotes the l_2 norm of a vector. After normalization, the TPA sends the audit results to the aggregation server $\mathcal{S}$. Formally, the TPA sends result $= (\alpha^r, TS^r)$ to the $\mathcal{S}$, where $\alpha^r = \{\alpha_1^r, \alpha_2^r, ..., \alpha_n^r\}$, $TS = \{TS_1^r, TS_2^r, ..., TS_n^r\}$, and n is the number of participants.

3) Aggregation: The aggregation server $\mathcal{S}$ starts the aggregation after receiving result. Here, we present a Byzantine-robust aggregation rule to improve our method's ability to resist malicious participants. Traditionally, employing a *weighted average* enables the consideration of the quality of parameters uploaded by multiple participants, thereby mitigating the negative impact of malicious participants on the aggregation process. Formally, we obtain the weighted average of the local model updates via

$$g^r = \frac{1}{\sum_{i=1}^n TS_i^r \cdot \alpha_i^r} \sum_{i=1}^n TS_i^r \cdot \alpha_i^r \cdot g_i^r, \tag{3}$$

and then compute $w_0^{r+1} \leftarrow w_0^r - \eta \cdot g^r$ to obtain the updated global model w_0^{r+1}, where η is the global model learning rate. However, using this method in FL-Auditor has significant drawbacks. Since a weighted average is used to obtain aggregation results, this requires that the aggregation server know information about the weights of different participants in the current iteration. Nevertheless, we use sampling auditing to compute the training weights, i.e., trust scores TS, for each participant to improve the security measures' efficiency. As a result, the aggregation server can only access the TS values of the sampled iterations. In other words, the method does not work when a certain iteration is not sampled.

A straightforward solution to the above problem is to invoke the FedAvg [6] algorithm for model aggregation when the current round r is not audited. In other words, we can directly average the model updates and use the average result as the global model for round r. Nevertheless, this approach is vulnerable to Byzantine attacks in the round r. An adversary may upload malicious parameters in the round r, threatening the secure aggregation of the local model. *Therefore, we need to propose a solution to mitigate the vulnerability of unsampled rounds to attacks.* Technically, the quality of the models contributed by FL participants of close rounds in the training process should be roughly similar. In other words, the trust scores of participant P_i should be relatively close in two close rounds. For example, if a malicious participant launches an attack in round r, it often launches an attack in round $r+1$ as well.

Considering this nature, we propose a *lazy update mechanism*[1] for the participant weights (i.e., TS_i and α_i). Intuitively, the mechanism obtains the corresponding trust score and normalization factor of each participant based on its historical behavior. More specifically, we employ a sliding window-based approach, allowing the aggregation server to observe trust scores TS from each participant in the past m audits and then calculate their average. This approach also serves the purpose of mitigating inaccuracies in trust score assessments caused by fluctuations in individual round trust scores. Formally, the trust score of participant $\mathcal{P}_i$ in the rth round is defined as

$$TS_i^r \leftarrow \frac{1}{m} \sum_{r-m}^{r-1} TS_i^r. \tag{4}$$

Furthermore, to ensure the timely update of the normalization factor, we introduce a forgetting factor denoted as λ_i in Equation (3). If round r is challenged, the forgetting factor for participant $\mathcal{P}_i$ is set to 1; otherwise, it is calculated as $\lambda_i^r = \frac{\|g_i^{old}\|}{\|g_i^{new}\|}$, where g_i^{old} and g_i^{new} are the updates closest to the current round and the new update, respectively (See lines 26-34 in Algorithm 1). Therefore, we rewrite Equation (3) as

$$g^r = \frac{1}{\sum_{i=1}^n TS_i^r \cdot \lambda_i^r \cdot \alpha_i^r} \sum_{i=1}^n TS_i^r \cdot \lambda_i^r \cdot \alpha_i^r \cdot g_i^r. \tag{5}$$

After the aggregation server completes the computation of Equation (5), it then uses the aggregated gradient g^r to perform a global model update, i.e., it computes $w_0^{r+1} \leftarrow w_0^r - \eta \cdot g^r$, where w_0^{r+1} denotes the global model after the rth round of updating. This entire process is repeated iteratively until the model converges or a predefined number of training rounds is reached.

C. Security Analysis

We give provable guarantees for the convergence of our FL-Auditor. Specifically, we show that the difference between the global model w_0 of our FL-Auditor (under attack) and the optimal global model w_0^* is bounded. Then, we first introduce the following assumptions and proceed with the corresponding theoretical analysis, presenting the results of the analysis.

Assumption 1 (L-Smooth)**.** *The local loss $\mathcal{L}_1, ..., \mathcal{L}_n$ are all L-smooth: for any w and $\hat{w}$, $\mathcal{L}_i(\hat{w}) \leq \mathcal{L}_i(w) + (\hat{w} - w)^T \nabla \mathcal{L}_i(w) + \frac{L}{2} \|\hat{w} - w\|_2^2$.*

Assumption 2 (μ-Strongly Convex)**.** *The local loss $\mathcal{L}_1, ..., \mathcal{L}_n$ are all μ-strongly convex: for any w and $\mathcal{L}_i(\hat{w}) \geq \mathcal{L}_i(w) + (\hat{w} - w)^T \nabla \mathcal{L}_i(w) + \frac{\mu}{2} \|\hat{w} - w\|_2^2$.*

Assumption 3 (Distribution of datasets)**.** *Each participant's training dataset D_i and the auxiliary validation dataset D_{eva} owned by TPA are independently sampled from the training dataset collection $\mathcal{X}$.*

[1]In practical deployment, we can compute TS, λ within the TPA to further reduce the computational overhead on the aggregation server. This is a trade-off between computational and communication costs. In fact, the computation complexity of these parameters is minimal, posing a negligible burden on the aggregation server.

Assumption 4. *The loss function $\mathcal{L}(D, w)$ on dataset D is L_1-Lipschitz probabilistically. In other words, there exists an L_1 for any δ in the interval (0, 1), satisfying:*

$$P\left\{\sup \frac{\|\nabla\mathcal{L}(D,w)-\nabla\mathcal{L}(D,\hat{w})\|}{\|w-\hat{w}\|} \leq L_1\right\} \geq 1-\frac{\delta}{3} \quad (6)$$

Lemma 1. *In each round, the distance between g and $\nabla\mathcal{L}(w)$ is bounded as follows:*

$$\|g-\nabla\mathcal{L}(w)\| \leq 3\|g_0-\nabla\mathcal{L}(w)\|+2\|\nabla\mathcal{L}(w)\|, \quad (7)$$

where $g=\sum_{i=1}^{n}\varphi_i\frac{\|g_0\|}{\|g_i\|}\cdot g_i$, $\varphi_i=\frac{TS_i\cdot\lambda_i\cdot\alpha_i}{\sum_{i=1}^{n}TS_i\cdot\lambda_i\cdot\alpha_i}$.

Proof. In this proof, we consider two scenarios: the first, where the round of iteration is selected by the TPA, and the second, where the round of iteration is not selected by the TPA. Following, we proceed to prove the first scenario. For ease of description, we abbreviate $\frac{\|g_0\|}{\|g_i\|}\cdot g_i$ to $\bar{g}$. If the round iteration is selected, then we have

$$\begin{aligned}
&\|g-\nabla\mathcal{L}(w)\| \\
&=\left\|\sum_{i=1}^{n}\varphi_i\bar{g}_i-\nabla\mathcal{L}(w)\right\| \\
&=\left\|\sum_{i=1}^{n}\varphi_i\bar{g}_i-g_0+g_0-\nabla\mathcal{L}(w)\right\| \\
&\leq\left\|\sum_{i=1}^{n}\varphi_i\bar{g}_i-g_0\right\|+\|g_0-\nabla\mathcal{L}(w)\| \\
&\leq\left\|\sum_{i=1}^{n}\varphi_i\bar{g}_i\right\|+\|g_0\|+\|g_0-\nabla\mathcal{L}(w)\| \\
&\leq\sum_{i=1}^{n}\varphi_i\|\bar{g}_i\|+\|g_0\|+\|g_0-\nabla\mathcal{L}(w)\| \\
&=\sum_{i=1}^{n}\varphi_i\|g_0\|+\|g_0\|+\|g_0-\nabla\mathcal{L}(w)\| \\
&=2\|g_0\|+\|g_0-\nabla\mathcal{L}(w)\| \\
&\leq 3\|g_0-\nabla\mathcal{L}(w)\|+2\|\nabla\mathcal{L}(w)\|.
\end{aligned} \quad (8)$$

If the round is not selected, we employ the model update g_0 from the TPA that is closest to the current round for the subsequent computation, denoted as g_0'. Since g_0' represents the model update from the TPA closest to the current round, $\|g_0'\|=\|g_0\|$. Therefore, we have

$$\begin{aligned}
&\|g-\nabla\mathcal{L}(w)\| \\
&\leq\sum_{i=1}^{n}\varphi_i\|\bar{g}_i\|+\|g_0\|+\|g_0-\nabla\mathcal{L}(w)\| \\
&=\sum_{i=1}^{n}\varphi_i\|g_0'\|+\|g_0\|+\|g_0-\nabla\mathcal{L}(w)\| \\
&=\sum_{i=1}^{n}\varphi_i\|g_0\|+\|g_0\|+\|g_0-\nabla\mathcal{L}(w)\| \\
&=2\|g_0\|+\|g_0-\nabla\mathcal{L}(w)\| \\
&\leq 3\|g_0-\nabla\mathcal{L}(w)\|+2\|\nabla\mathcal{L}(w)\|.
\end{aligned} \quad (9)$$

After this, we show that the aforementioned conclusion is satisfied in both scenarios. Therefore, this lemma holds. □

Theorem 1. *For any number of Byzantine participants, the discrepancy between the global model w_0 learned by FL-Auditor and the optimal global model w^* under no attack is bounded. Specifically, the global model w_0 and the optimal model w^* in round t are satisfied:*

$$\begin{aligned}
\left\|w_0^t-w^*\right\| \leq&(\sqrt{1-\mu^2/(4L^2)}+24\eta\triangle_2+2\eta L)\left\|w_0^0-w^*\right\| \\
&+12\eta\triangle_1,
\end{aligned} \quad (10)$$

where $\triangle_1=\sqrt{2}\sigma_1\sqrt{(d\log 6+\log(3/\delta))/|D_{eva}|}$, $\triangle_2=\sigma_2\sqrt{2/|D_{eva}|}\sqrt{d\log\frac{18L_2}{\delta_2}+\frac{d}{2}\log\frac{|D_{eva}|}{d}+\log(\frac{6\sigma_2^2r\sqrt{|D_{eva}|}}{\gamma_2\sigma_1\delta})}$, w_0^0 represents the global model at round 0.

Proof. Based on Lemma 1, we have

$$\begin{aligned}
&\left\|w_0^t-w^*\right\| \\
&=\left\|w^{t-1}-\eta g^{t-1}-w^*\right\| \\
&\leq\left\|w^{t-1}-w^*-\eta\nabla\mathcal{L}(w^{t-1})\right\|+\left\|\eta\nabla\mathcal{L}(w^{t-1})-\eta g^{t-1}\right\| \\
&\leq\left\|w^{t-1}-w^*-\eta\nabla\mathcal{L}(w^{t-1})\right\|+3\eta\left\|g_0^{t-1}-\nabla\mathcal{L}(w^{t-1})\right\| \\
&+2\eta\left\|\nabla\mathcal{L}(w^{t-1})\right\| \\
&=\left\|w^{t-1}-w^*-\eta\nabla\mathcal{L}(w^{t-1})\right\|+3\eta\left\|g_0^{t-1}-\nabla\mathcal{L}(w^{t-1})\right\| \\
&+2\eta\left\|\nabla\mathcal{L}(w^{t-1})-\nabla\mathcal{L}(w^*)\right\|
\end{aligned} \quad (11)$$

To continue the proof, we introduce the following lemmas, previously proved in [30].

Lemma 2. *If learning rate $\eta=\frac{\mu}{2L^2}$, then we have the following for round t*

$$\left\|w^{t-1}-w^*-\eta\nabla\mathcal{L}(w^{t-1})\right\| \leq \sqrt{1-\mu^2/(4L^2)}\left\|w^{t-1}-w^*\right\|. \quad (12)$$

Lemma 3. *Let all assumptions hold and $\Theta \subset \{w : \|w-w^*\| \leq r\sqrt{d}\}$ holds, then for any $\delta \in (0,1)$, if $\triangle_1 \leq \sigma_1^2/\gamma_1$ and $\triangle_2 \leq \sigma_2^2/\gamma_2$, then for $w \in \Theta$ the following inequality holds*

$$P\{\|g_0-\nabla\mathcal{L}(w)\| \leq 8\triangle_2\|w-w^*\|+4\triangle_1\} > 1-\delta, \quad (13)$$

where $\triangle_1=\sqrt{2}\sigma_1\sqrt{(d\log 6+\log(3/\delta))/|D_{eva}|}$, $\triangle_2=\sigma_2\sqrt{2/|D_{eva}|}\sqrt{d\log\frac{18L_2}{\delta_2}+\frac{d}{2}\log\frac{|D_{eva}|}{d}+\log(\frac{6\sigma_2^2r\sqrt{|D_{eva}|}}{\gamma_2\sigma_1\delta})}$, $L_2=\max\{L,L_1\}$, $\sigma_1,\sigma_2,\gamma_1,\gamma_2$ are positive constants, Θ is the parameter space of global model, and $|D_{eva}|$ is the size of the auxiliary validation dataset.

Based on Lemma 2, Lemma 3, and Assumption 1, we deduce that

$$\begin{aligned}
&\left\|w^{t-1}-w^*-\eta\nabla\mathcal{L}(w^{t-1})\right\|+3\eta\left\|g_0^{t-1}-\nabla\mathcal{L}(w^{t-1})\right\| \\
&+2\eta\left\|\nabla\mathcal{L}(w^{t-1})-\nabla\mathcal{L}(w^*)\right\| \\
&\leq(\sqrt{1-\mu^2/(4L^2)}+24\eta\triangle_2+2\eta L)\left\|w^{t-1}-w^*\right\|
\end{aligned} \quad (14)$$

By iteratively employing the aforementioned inequality, we deduce that

$$\begin{aligned}
\left\|w_0^t-w^*\right\| \leq&(\sqrt{1-\mu^2/(4L^2)}+24\eta\triangle_2+2\eta L)\left\|w_0^0-w^*\right\| \\
&+12\eta\triangle_1
\end{aligned} \quad (15)$$

And thus, we conclude the proof. □

V. Evaluation

In this section, we first describe the details of the experimental setup and then show the results of the three experiments.

A. Experimental Setup

1) Datasets: In our experiments, we employed three datasets, namely MNIST[2], FEMNIST[3] and CIFAR-10[4]. The MNIST dataset consists of handwritten digits from 0 to 9 with a resolution of 28 x 28 pixels, while the CIFAR-10 dataset contains 60,000 32 x 32 color images classified into 10 classes. We distribute the above dataset evenly or randomly to the participants. The FEMNIST is an open-source FL benchmark dataset from the FL benchmark framework "LEAF" [46]. This dataset is an extension of EMNIST and has been divided into different groups based on different writers. As a result, it provides a more realistic Non-IID (Non-Independently and Identically Distributed) distribution compared to other datasets. For the dataset distribution, we only need to partition MNIST and CIFAR-10 into IID and non-IID distributions. Specifically, for the IID datasets, we randomly shuffled the MNIST and CIFAR-10 datasets and evenly distributed them to each participant. As for the non-IID dataset, we partitioned the MNIST dataset into 200 shards, each of size 300, and assigned two shards to each participant.

2) Models: In our experiments, we used CNN (convolutional neuralsddd networks), MLP (multilayer perceptrons), and ResNet20 [47]. For the MNIST dataset, we trained a CNN and MLP as the global model. For the CIFAR-10 dataset, we use CNN and ResNet20 as global model. The CNN model consists of two 5x5 convolutional layers, a fully connected layer, a ReLU activation function and a softmax output layer. Similarly, we used CNN as the global model for FEMNIST. The MLP contains two fully connected layers and a ReLU activation function. For model training, we set the hyperparameters as follows: the number of participants $n = 100$, the total number of communication rounds $R = 20$, the learning rate $\eta = 0.01$, and the number of local iterations $E = 5$. Furthermore, each participant is given the same parameters, the only difference is the dataset.

3) Implementation Details: We implemented the FL-Auditor using Python programming language and PyTorch deep learning framework for model building and training. All simulation experiments were conducted on an AI server with an Intel Core i9-10900X CPU @ 3.70 GHz and two NVIDIA GeForce RTX 3080Ti GPUs.

4) Baselines: In our experiments, we used the following FL schemes as baselines.

- FedAvg [6]: FedAvg is the typical FL framework that enables aggregation of local models without attacks. It operates by taking the average of the local models uploaded by the participants and using the resulting average as the global model.
- Median [17]: Median is a Byzantine-robust FL aggregation rule that computes the median of local model updates from multiple participants to obtain the global model update. It is designed to mitigate the impact of malicious attacks in the FL setting.
- Trim-mean [17]: Trim-mean removes a portion of the maximum and minimum local model updates received, then averages the remaining ones and uses the average as a global model update.
- Krum [16]: Krum chooses an update from multiple local model updates that is similar to other local model updates as the global model update. Technically, Krum takes the sum of the *norm* distances of a gradient from other gradients as the score of that gradient, and then selects the lowest score, i.e., the gradient that is similar to most of the gradients, as the aggregated gradient.
- FLTrust [30]: FLTrust calculates a trust score for each local model update by assessing its cosine similarity with a server model update trained on a clean dataset. It then computes a weighted average of the local model updates based on the trust score to derive the global model update.

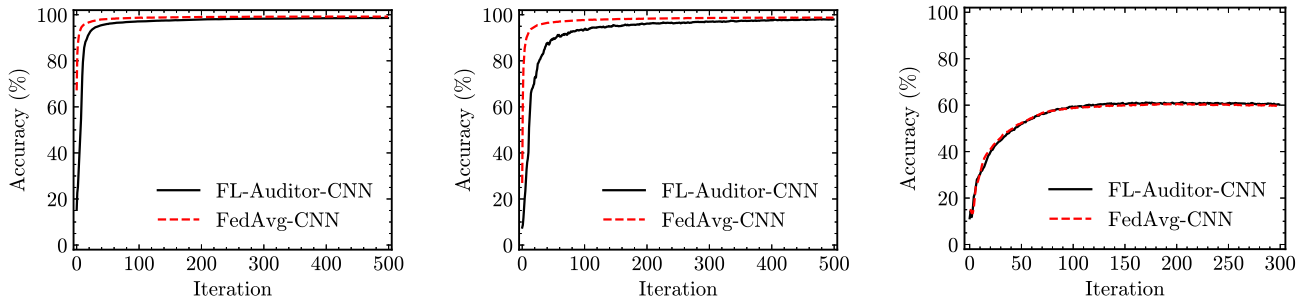
5) Adversaries: In our experiments, we consider three kinds of adversaries: free-riders, label-flipping, and sign-flipping. For free-riders, they upload a randomly drawn gradient from a uniform distribution of [-1,1]. For label-flipping, they flip the gradient direction and upload it. Formally, the participants $\mathcal{P}_i$ compute the true gradient g_i and then send $\hat{g}_i = -g_i$ to the server. For label-flipping, they change the labels of the training samples. Formally, they use the formula $t = M - 1 - c$ to change the training sample labels from c to t, where M is the total number of labels and c is the label. After changing the labels of the samples, the participants train their local models on the changed labeled training samples and subsequently transmit the local model updates, i.e., gradients, to the server.

B. Experimental Results

We conducted experimental evaluations of FL-Auditor across three criteria: accuracy, robustness to attacks, and flexibility.

1) Accuracy: We conducted experiments on the MNIST and CIFAR-10 datasets to evaluate the impact of FL-Auditor on model accuracy. Specifically, we compared the accuracy of the FL-Auditor model against that of FedAvg on the MNIST and CIFAR-10 datasets under a no-attack condition. For this experiment, we set the fraction of rounds sampled by TPA $\lambda = 0.8$, the local iterations $E = 2$. Here, we choose FedAvg as the evaluation benchmark of model accuracy. This is because FedAvg is a classical FL aggregation rule without security measures, and its performance is almost optimal in the absence of attacks compared to existing FL aggregation rules. In other words, the closer FL-Auditor is to FedAvg, the less impact the security measures have on the global model. Fig. 3 shows the corresponding experimental results. From the results shown in Fig. 3, it can be observed that FL-Auditor has an impact on the convergence speed while not affecting the accuracy of the global model. The sampling

[2]http://yann.lecun.com/exdb/mnist/
[3]https://leaf.cmu.edu/
[4]https://www.cs.toronto.edu/~kriz/cifar.html

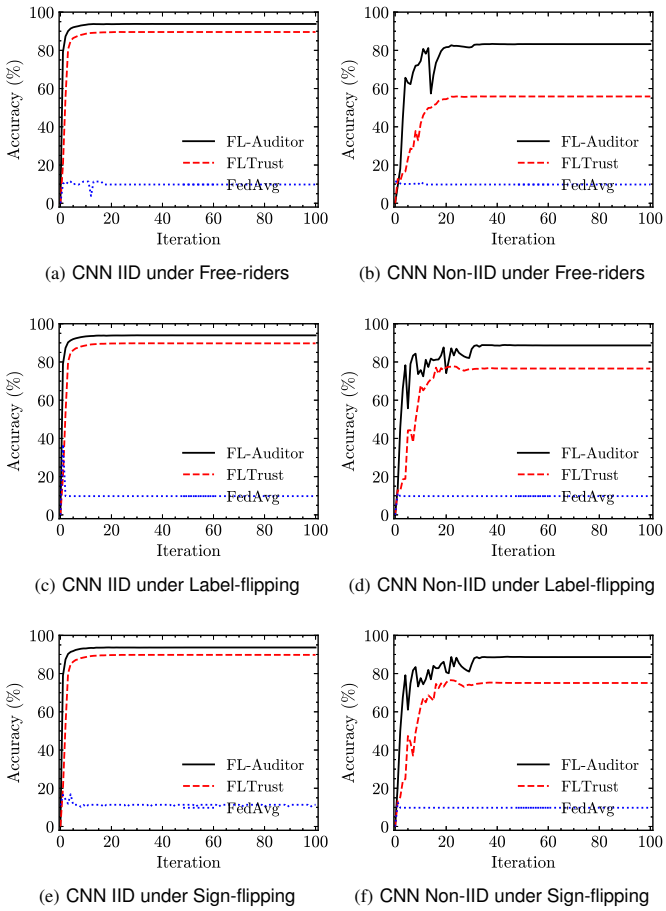


(a) MNIST IID (b) MNIST Non-IID (c) CIFAR IID

Fig. 3. The accuracy of FL-Auditor, FedAvg without Byzantine attacks. Here, we employed a CNN as the global model and evaluated its performance on the MNIST and CIFAR-10 datasets.

audit method used by FL-Auditor has a slight impact on the convergence speed of the model but does not affect the model accuracy since it can aggregate benign updates. In summary, the experimental results indicate that FL-Auditor has no effect on model accuracy. Instead, the impact on the convergence speed and the security of the measure is a trade-off. This is a reasonable sacrifice for Byzantine-robust design requirements, and is affordable for FL training. In addition, we can observe that the Non-IID dataset converges more slowly compared to the IID dataset. This is due to the uneven data distribution in the Non-IID dataset. Technically, the Non-IID dataset is unevenly distributed, leading to large differences between local models and thus reducing the usability of the global model. This demonstrates that our FL-Auditor achieves the accuracy goal.

2) Robustness against attacks: To assess the robustness of the FL-Auditor, we evaluated the prediction accuracy of the global model when exposed to Byzantine attacks on the MNIST dataset. In this experiment, we considered the cases where the global model is a CNN and an MLP, respectively. In addition, we set the sampling rate of FL-Auditor $\lambda = 0.8$, the local iteration $E = 3$, and the percentage of Byzantine participants $\rho = 0.4$, where λ and ρ denote the fraction of rounds sampled by TPA and the fraction of Byzantine participants, respectively. Similar to the accuracy evaluation, we used the IID and non-IID datasets. More specifically, we compared FL-Auditor with FLTrust and FedAvg. The results of the experiment are presented in Fig. 4 and Fig. 5. From Fig. 4 we can see that FL-Auditor has better performance than FLTrust and FedAvg under each attack (i.e., free-riders, label-flipping, sign-flipping). This indicates that the method used by FL-Auditor for Byzantine defense by selecting specific training rounds has almost no effect on Byzantine robustness performance. This is due to our design of a lazy update mechanism that maximizes the performance of the sampling audit. Similarly, the FL-Auditor has an optimal performance in Fig. 5. Furthermore, FL-Auditor exhibits a greater performance improvement, indicating its robustness across different models. Additionally, our FL-Auditor experiences a smaller performance loss under non-IID conditions compared to the other two methods, highlighting its robustness to data heterogeneity. This demonstrates that our FL-Auditor achieves the security goal.



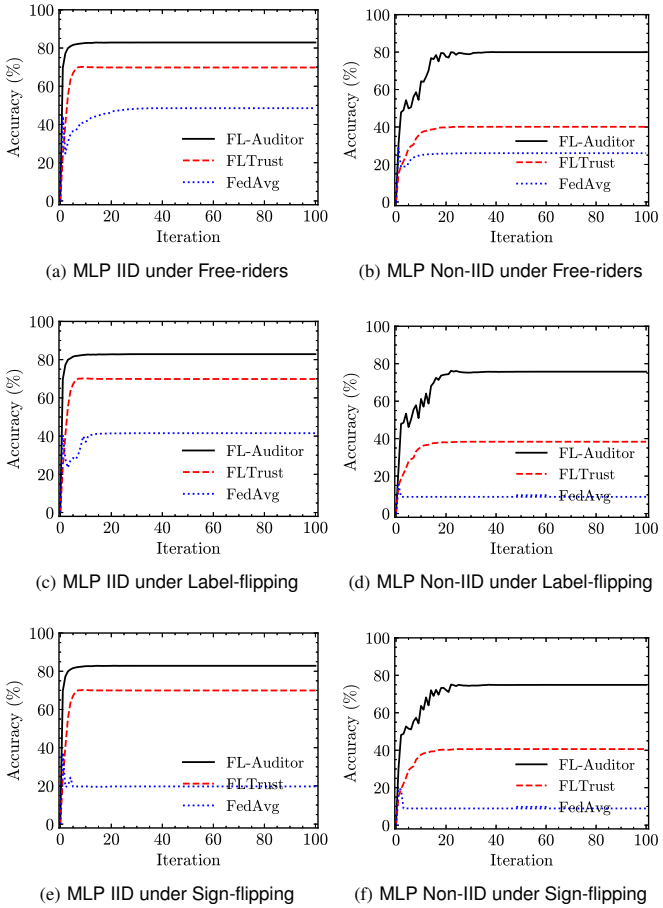
(a) CNN IID under Free-riders (b) CNN Non-IID under Free-riders

(c) CNN IID under Label-flipping (d) CNN Non-IID under Label-flipping

(e) CNN IID under Sign-flipping (f) CNN Non-IID under Sign-flipping

Fig. 4. The accuracy of FL-Auditor, FedAvg, and FLTrust under free-riders, label-flipping, and sign-flipping attacks using CNN as a global model.

Moreover, to evaluate the performance of FL-Auditor on complex datasets, we trained a global model using ResNet20 on the CIFAR-10 dataset and evaluated its accuracy. In this experiment, we set the local training learning rates (η) to be 0.01 and 0.0002, and the local iteration number E to be 5. Detailed experimental results are shown in Fig. 6. From Fig. 6, it is evident that FL-Auditor outperforms other methods, particularly maintaining good model performance at a learning rate of 0.01.

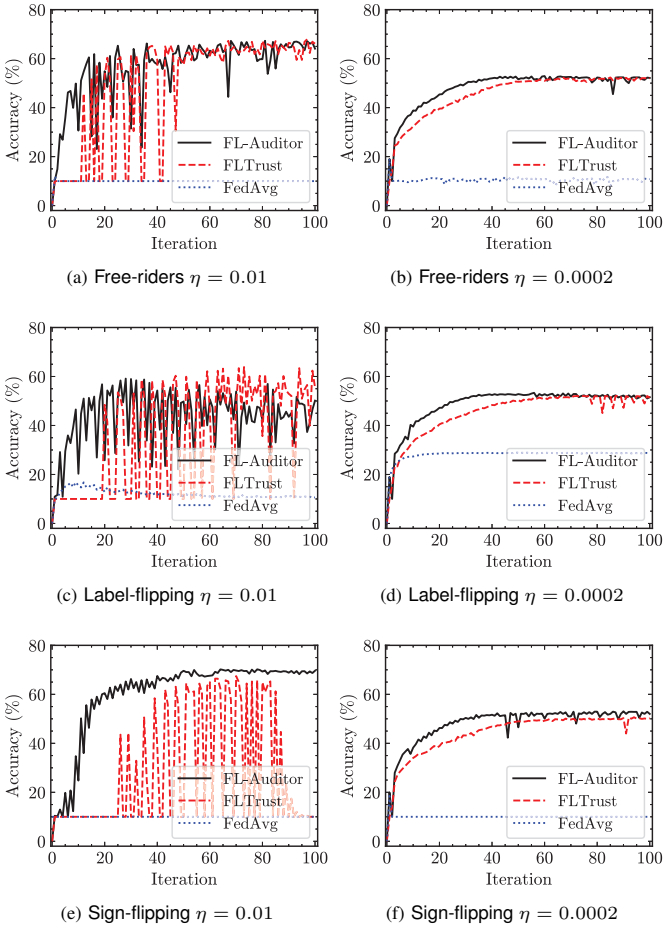


(a) MLP IID under Free-riders (b) MLP Non-IID under Free-riders

(c) MLP IID under Label-flipping (d) MLP Non-IID under Label-flipping

(e) MLP IID under Sign-flipping (f) MLP Non-IID under Sign-flipping

Fig. 5. The accuracy of FL-Auditor, FedAvg, and FLTrust under free-riders, label-flipping, and sign-flipping attacks using MLP as a global model.



(a) Free-riders $\eta = 0.01$ (b) Free-riders $\eta = 0.0002$

(c) Label-flipping $\eta = 0.01$ (d) Label-flipping $\eta = 0.0002$

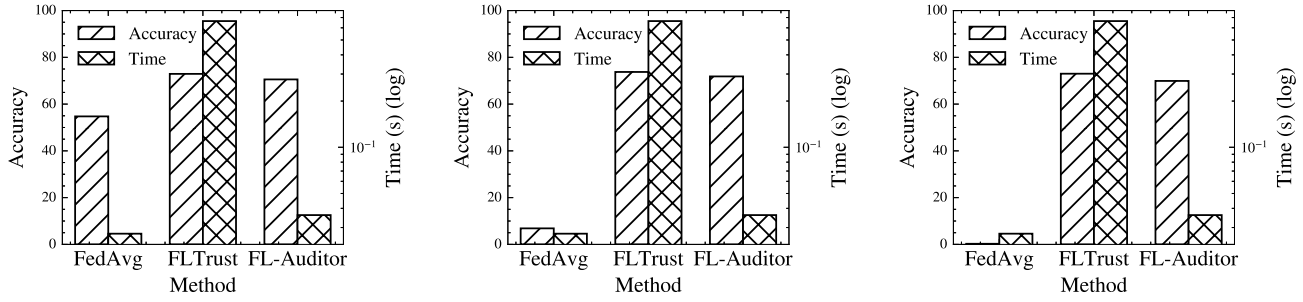
(e) Sign-flipping $\eta = 0.01$ (f) Sign-flipping $\eta = 0.0002$

Fig. 6. The accuracy of FL-Auditor, FedAvg, and FLTrust under attacks, including free-riders, label-flipping, and sign-flipping, using ResNet as the global model with the CIFAR-10 dataset.

In contrast, FLTrust fails to converge under these conditions, demonstrating the effectiveness of the public auditing protocol in FL-Auditor. Meanwhile, when $\eta = 0.0002$, both FL-Auditor and FLTrust converge successfully, but FL-Auditor maintains a performance advantage. This indicates the robustness of FL-Auditor to Byzantine attacks and its resilience to variations in the learning rate η. These findings further demonstrate the robustness of FL-Auditor against Byzantine attacks.

Furthermore, to assess the robustness of our FL-Auditor against attacks on natural Non-IID datasets, we conducted experiments comparing the global model accuracy of FedAvg, FLTrust, and FL-Auditor using the FEMNIST dataset. In this experiment, we randomly selected data from 190 participants in the FEMNIST dataset. Meanwhile, we also evaluated their computational overhead on the aggregation server. The experimental results are shown in Fig. 7. It's evident from Fig. 7 that FL-Auditor and FLTrust exhibit similar model accuracy on the FEMNIST dataset. However, FL-Auditor substantially reduces the computational burden on the aggregation server. This illustrates that our FL-Auditor mitigates the computational overhead on the aggregation server with negligible impact on the model's performance. It further validates the effectiveness of the lazy update mechanism in our FL-Auditor, emphasizing its capability to adeptly tackle the challenge of model performance degradation stemming from sampling audits. In contrast, although FLTrust attains superior model performance, it imposes a higher computational overhead on the aggregation server due to the necessity of training the root model on the server. Moreover, the model performance of our FL-Auditor is also very close to FLTrust. This demonstrates the effectiveness of FL-Auditor in using the TPA to strengthen FL against Byzantine attacks.

3) Flexibility: In general, the sampling rate and the number of malicious participants may have an impact on the performance of FL-Auditor. For this reason, we evaluated the impact of these two parameters on the global model accuracy separately. First, we tested the effect of FL-Auditor sampling rate on the robustness of the attack. Concretely, we tested the classification accuracy of the global model with setting λ to 0.5, 0.6, 0.7, 0.8, and 0.9, and the experimental results are shown in Fig. 8. From Fig. 8, we can see that the sampled fraction λ has a beneficial influence on the model accuracy. This is because as the sampling fraction increases, we can assign more fine-grained weights to different participants. In other words, TPA can compute a more accurate trust score for the participants. In this way, the aggregation server can aggregate better local models. Meanwhile, from Fig. 8, we can observe that the increase of λ does not have a particularly



(a) Free-riders (b) Label-flipping (c) Sign-flipping

Fig. 7. FL-Auditor, FedAvg, and FLTrust's performance under free-riders, label-flipping, and sign-flipping attacks. Here, we present both the global model accuracy and the time overhead on the aggregation server for these different methods.

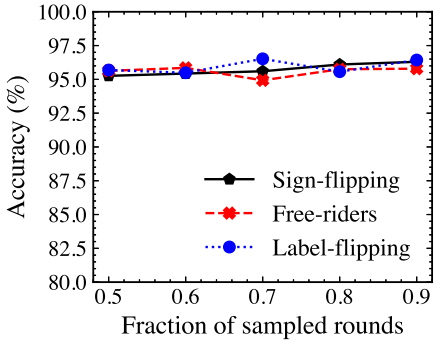


Fig. 8. The impact of the sampling ratio on the global model of FL-Auditor.

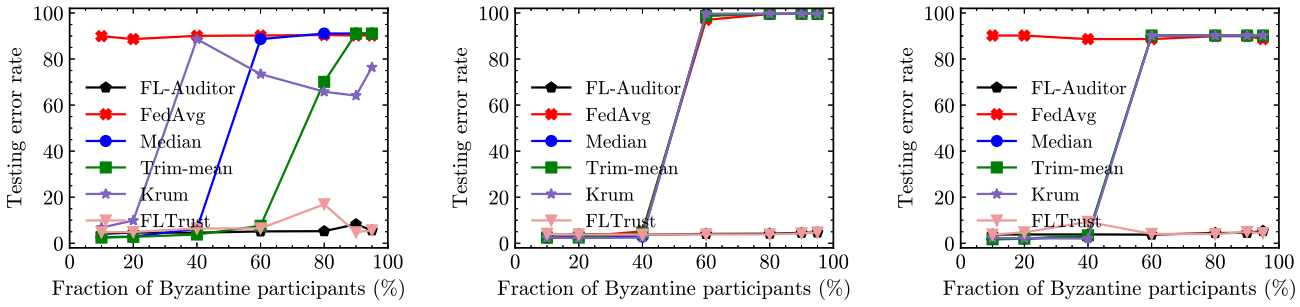
significant effect on the model accuracy. This is because we have designed a lazy update mechanism for the model that maximizes the negative impact of sampling audit on accuracy. This also demonstrates the effectiveness of our proposed public audit mechanism i.e., our proposed scheme can decrease the effect of Byzantine robustness measures on training overhead without affecting the model accuracy.

To evaluate the robustness of FL-Auditor against the number of Byzantine participants, we compared and tested FL-Auditor with state-of-the-art solutions, including Krum [16], Median [17], Trim-mean [17], and FLTrust [30]. The test error rate was used as the metric to evaluate the performance of each method, which represents the fraction of test sample labels that are predicted incorrectly by the global model. Specifically, we tested the error rates of different schemes on the MNIST dataset for three types of attacks: free-riders, label flipping, and sign-flipping. In this experiment, we used CNN as the global model, set the fraction of Byzantine participants ρ to increment from 10%, set the maximum value of ρ to 95%, and set the number of participants to 100. The corresponding results are presented in Figure 9, which demonstrates the test error rates of different FL methods under free-riders, label-flipping, and sign-flipping attacks as the proportion of Byzantine participants increases. From Fig. 9, it can be observed that as the fraction of Byzantine participants increases, FL-Auditor can still maintain a very low testing error rate even when the number of Byzantine participants reaches 95%. However, the existing Byzantine-robust FL schemes can only tolerate a small number of Byzantine participants. For example, Median's test error rate starts to increase steeply when the fraction of Byzantine participants reaches 40% under label-flipping attacks. Meanwhile, we can also find that FLTrust, similar to our FL-Auditor, shows a lower testing error rate. In other words, FL-Auditor can achieve similar performance as FLTrust when auditing part of the training process. This also demonstrates that the sampling approach used by FL-Auditor does not have an impact on the effectiveness of Byzantine defense. Overall, our results show that FL-Auditor is robust to attacks and is almost independent of the number of Byzantine participants, even if a smaller number of training processes are evaluated by random sampling.

TABLE II
TIME AND COMMUNICATION COMPLEXITY IN FL-AUDITOR

	Aggregation server	TPA	Participant
Time complexity	$\mathcal{O}(Rn \cdot \mathrm{AGR}_c)$	$\mathcal{O}(\lambda R \cdot \mathrm{AGD}_c)$	$\mathcal{O}(R \cdot \mathrm{M}_{train})$
Communication complexity	$\mathcal{O}(2nR \cdot \lvert\mathbb{X}\rvert + \lambda n \cdot \lvert\mathbb{X}\rvert \cdot R + \lvert\varpi\rvert)$	$\mathcal{O}(\lambda n \cdot \lvert\mathbb{X}\rvert \cdot R + \lvert\varpi\rvert)$	$\mathcal{O}(2R \cdot \lvert\mathbb{X}\rvert)$

To evaluate the computational overhead of our FL-Auditor, we have analyzed the time complexity of FL-Auditor on the server, the communication complexity between the server and TPA, and the time complexity on the TPA side. Table II shows the corresponding time complexity of FL-Auditor. The time and communication complexity of the aggregation server in FL-Auditor are $\mathcal{O}(Rn \cdot \mathrm{AGR}_c)$ and $\mathcal{O}(2nR \cdot |\mathbb{X}| + \lambda n \cdot |\mathbb{X}| \cdot R + |\varpi|)$, respectively. Here, n represents the number of participants, R denotes the communication rounds, $|\mathbb{X}|$ represents the amount of parameters uploaded by a participant in one communication round, $|\varpi|$ is the additional parameters transmitted when the TPA audits the aggregation server, and $n \cdot \mathrm{AGR}_c$ is the time spent by the aggregation rule to aggregate n participants. The time and communication complexity of the TPA are denoted as $\mathcal{O}(\lambda R \cdot \mathrm{AGD}_c)$ and $\mathcal{O}(\lambda n \cdot |\mathbb{X}| \cdot R + |\varpi|)$, where AGD_c is the computational time required by the TPA to perform Byzantine attack defense, and λ is the sampling rate for all training rounds. The corresponding time and communication complexity for a participant are $\mathcal{O}(R \cdot \mathrm{M}_{train})$ and $\mathcal{O}(2R \cdot |\mathbb{X}|)$, where M_{train} represents the time a participant spends on local model training. For the aggregation server, its time complexity is significantly reduced as it only needs to perform the aggregation operation.



(a) Free-riders (b) Label-flipping (c) Sign-flipping

Fig. 9. The effect of Byzantine participant ratio on the test error rate of different FL methods (including FL-Auditor, FedAvg, Median, Trim-mean, Krum, and FLTrust).

However, with the introduction of TPA, there is an additional communication overhead of $\lambda n \cdot |\mathbb{X}| \cdot R + |\varpi|$. Nevertheless, FL-Auditor adopts a sampling approach for auditing, and with an appropriate sampling rate λ, the introduced communication overhead is something the aggregation server can afford. Here, $|\varpi|$ represents the proof and verification results transmitted between TPA and the aggregation server, and these parameters are negligible compared to the model parameters. Therefore, the communication overhead between the aggregation server and TPA is affordable for FL-Auditor.

In addition, we compared it with existing methods in terms of computational cost on the aggregation server. In this experiment, we used CNN model as global model and used MNIST, FEMNIST,and CIFAR-10 datasets respectively. For the MNIST and CIFAR-10 datasets we set the number of participants to 10, while for the FEMNIST dataset we randomly selected 190 participants i.e. the number of participants is 190. The experimental results are shown in Table III. All results were computed eight times and then averaged. From Table III, we can observe that our FL-Auditor exhibits the lowest computational cost, except for FedAvg, which lacks any security measures. Compared to the FLTrust, our FL-Auditor reduces the time overhead on the aggregation server by $8\times$ to $17\times$. This is because our FL-Auditor does not change the original aggregation rules of FL. In contrast, FLTrust requires the root model to be trained on an aggregation server, which leads to higher computational costs. If the aggregation server does not have a GPU, the time overhead resulting from training the root model on the aggregation server will be even more pronounced. Similarly, other methods also modify the aggregation rules, resulting in higher time overhead compared to FedAvg. This demonstrates the advantage of our FL-Auditor, which significantly reduces the time overhead of the aggregation server without altering the original aggregation rules.

TABLE III
THE TIME OVERHEAD (IN SECOND) ON THE AGGREGATION SERVER FOR VARIOUS FL METHODS ON THE MNIST, FEMNIST, AND CIFAR-10 DATASETS

Dataset		FL-Auditor	FLTrust	FedAvg	Median	Trim-mean	Krum
MNIST	IID	**0.00260**	0.02395	0.00223	0.00764	0.00622	0.00748
	Non-IID	**0.00330**	0.02972	0.00235	0.00811	0.00659	0.00782
FEMNIST	Non-IID	**0.03692**	0.66282	0.02732	0.67833	1.17527	6.71043
CIFAR-10	IID	**0.00404**	0.03390	0.00280	0.01483	0.01178	0.01098

VI. CONCLUSION

In this paper, we propose FL-Auditor, an efficient Byzantine-robust FL framework based on public auditing. FL-Auditor conducts auditing of a sample of FL training processes using a trusted third-party auditor (TPA) to mitigate the impact of malicious participants. Specifically, the TPA first initiates an audit challenge for the aggregation server to receive the information for the audit. Then, the TPA evaluates the parameters of each participant and returns the results to the server. Finally, the server aggregates the local models based on the results to obtain the correct global model. Compared to existing FL schemes, FL-Auditor reduces the computational overhead of the aggregation server and Byzantine robust algorithm by introducing a TPA to audit a sample of the FL training process. This approach also greatly avoids direct modifications to the original FL workflow, making it very user-friendly for FL participants. We also conducted an exhaustive experimental evaluation of FL-Auditor, and the results show that FL-Auditor achieves resistance to Byzantine attacks while ensuring efficiency. In the future, we aim to further enhance the capabilities of TPA, enabling it to not only achieve robust model aggregation but also ensure the integrity of the FL training process. In addition, we will take precautions to mitigate potential privacy leakage risks associated with the introduction of TPA.

ACKNOWLEDGMENTS

The authors would like to thank the Editor-in-Chief, the Associate Editor, and the reviewers for their insightful comments and suggestions.

REFERENCES

[1] Y. Zeng, B. Song, Y. Chen, X. Du, and M. Guizani, "Few-shot scale-insensitive object detection for edge computing platform," *IEEE Transactions on Sustainable Computing*, vol. 7, no. 4, pp. 726–735, 2022.

[2] X. Wang and Y. Lu, "Sustainable and efficient fog-assisted iot cloud based data collection and delivery for smart cities," *IEEE Transactions on Sustainable Computing*, vol. 7, no. 4, pp. 950–957, 2022.

[3] X. Pang, Z. Wang, D. Liu, J. C. Lui, Q. Wang, and J. Ren, "Towards personalized privacy-preserving truth discovery over crowdsourced data streams," *IEEE/ACM Transactions on Networking*, vol. 30, no. 1, pp. 327–340, 2021.

[4] Z. Zhang, L. Wu, D. He, Q. Wang, D. Wu, X. Shi, and C. Ma, "G-VCFL: grouped verifiable chained privacy-preserving federated learning," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4219–4231, 2022.

[5] X. Pang, Z. Wang, Z. He, P. Sun, M. Luo, J. Ren, and K. Ren, "Towards class-balanced privacy preserving heterogeneous model aggregation," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 3, pp. 2421–2432, 2023.

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[7] Z. Lian, W. Wang, Z. Han, and C. Su, "Blockchain-based personalized federated learning for internet of medical things," *IEEE Transactions on Sustainable Computing*, 2023.

[8] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[9] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the tails: Yes, you really can backdoor federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 070–16 084, 2020.

[10] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2938–2948.

[11] X. Cao and N. Z. Gong, "MPAF: Model poisoning attacks to federated learning based on fake clients," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3396–3404.

[12] Z. Zhang, A. Panda, L. Song, Y. Yang, M. W. Mahoney, J. E. Gonzalez, K. Ramchandran, and P. Mittal, "Neurotoxin: Durable backdoors in federated learning," *arXiv preprint arXiv:2206.10341*, 2022.

[13] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1142–1154, 2022.

[14] X. Pan, M. Zhang, D. Wu, Q. Xiao, S. Ji, and Z. Yang, "Justinian's GAAvernor: Robust distributed learning with gradient aggregation agent," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1641–1658.

[15] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "DBA: Distributed backdoor attacks against federated learning," in *International Conference on Learning Representations*, 2019.

[16] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: byzantine tolerant gradient descent," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 118–128.

[17] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5650–5659.

[18] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in byzantium," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3521–3530.

[19] C. Fung, C. J. 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.

[20] C. Xie, S. Koyejo, and I. Gupta, "Zeno++: Robust fully asynchronous sgd," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 495–10 503.

[21] X. Ma, X. Sun, Y. Wu, Z. Liu, X. Chen, and C. Dong, "Differentially private byzantine-robust federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 3690–3701, 2022.

[22] X. Li, Z. Qu, S. Zhao, B. Tang, Z. Lu, and Y. Liu, "Lomar: A local defense against poisoning attack on federated learning," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[23] B. Zhao, P. Sun, T. Wang, and K. Jiang, "FedInv: Byzantine-robust federated learning by inversing local model updates," in *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, 2022, pp. 9171–9179.

[24] C. Zhu, S. Roos, and L. Y. Chen, "Leadfl: Client self-defense against model poisoning in federated learning," in *International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 202. PMLR, 2023, pp. 43 158–43 180.

[25] S. Park, S. Han, F. Wu, S. Kim, B. Zhu, X. Xie, and M. Cha, "Feddefender: Client-side attack-tolerant federated learning," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 1850–1861.

[26] M. Fang, M. Sun, Q. Li, N. Z. Gong, J. Tian, and J. Liu, "Data poisoning attacks and defenses to crowdsourcing systems," in *Proceedings of the Web Conference 2021*, 2021, pp. 969–980.

[27] C. Xie, S. Koyejo, and I. Gupta, "Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6893–6901.

[28] S. Andreina, G. A. Marson, H. Möllering, and G. Karame, "BaFFLe: Backdoor detection via feedback-based federated learning," in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2021, pp. 852–863.

[29] Z. Zhang, L. Wu, C. Ma, J. Li, J. Wang, Q. Wang, and S. Yu, "LSFL: A lightweight and secure federated learning scheme for edge computing," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 365–379, 2023.

[30] 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*. The Internet Society, 2021, pp. 1–18.

[31] O. Ibitoye, M. O. Shafiq, and A. Matrawy, "Differentially private self-normalizing neural networks for adversarial robustness in federated learning," *Computers & Security*, vol. 116, p. 102631, 2022.

[32] C. Wang, X. Wu, G. Liu, T. Deng, K. Peng, and S. Wan, "Safeguarding cross-silo federated learning with local differential privacy," *Digital Communications and Networks*, vol. 8, no. 4, pp. 446–454, 2022.

[33] V. Valadi, X. Qiu, P. P. B. de Gusmão, N. D. Lane, and M. Alibeigi, "Fedval: Different good or different bad in federated learning," in *32nd USENIX Security Symposium*. USENIX Association, 2023, pp. 6365–6380.

[34] L. Muñoz-González, K. T. Co, and E. C. Lupu, "Byzantine-robust federated machine learning through adaptive model averaging," *arXiv preprint arXiv:1909.05125*, 2019.

[35] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *European Symposium on Research in Computer Security*. Springer, 2020, pp. 480–501.

[36] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage, "Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1354–1371.

[37] J. Park, D.-J. Han, M. Choi, and J. Moon, "Sageflow: Robust federated learning against both stragglers and adversaries," *Advances in Neural Information Processing Systems*, vol. 34, pp. 840–851, 2021.

[38] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Defending against saddle point attack in byzantine-robust distributed learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7074–7084.

[39] Y. Deng, M. M. Kamani, and M. Mahdavi, "Distributionally robust federated averaging," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15 111–15 122, 2020.

[40] Z. Zhang, L. Wu, D. He, J. Li, S. Cao, and X. Wu, "Communication-efficient and byzantine-robust federated learning for mobile edge computing networks," *IEEE Network*, vol. 37, no. 4, pp. 112–119, 2023.

[41] S. Shen, S. Tople, and P. Saxena, "Auror: Defending against poisoning attacks in collaborative deep learning systems," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, 2016, pp. 508–519.

[42] V. Shejwalkar and A. Houmansadr, "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning," in *NDSS*, 2021.

[43] 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.

[44] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" *arXiv preprint arXiv:1911.07963*, 2019.

[45] T. D. Nguyen, P. Rieger, H. Chen, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, S. Zeitouni, F. Koushanfar, A. Sadeghi, and T. Schneider, "FLAME: Taming backdoors in federated learning," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 1415–1432.

[46] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.

[47] 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*, 2016, pp. 770–778.

Zhuangzhuang Zhang received the B.S. degree in software engineering from Taiyuan University of Technology, Taiyuan, China, in 2017, the M.S. degree from the College of Information and Computer, Taiyuan University of Technology, Taiyuan, China, in 2020. He is currently pursuing the Ph.D. degree in the School of Cyber Science and Engineering, Wuhan University, China. His main research interests include AI security and data security.

Libing Wu received the B.S. and M.S. degrees in computer science from Central China Normal University, Wuhan, China, in 1994 and 2001, respectively, and the Ph.D. degree from Wuhan University, Wuhan, in 2006. He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University. He is also a Researcher with Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen, China. His research interests include network security, Internet of Things, machine learning and data security.

Debiao He received his Ph.D. degree in applied mathematics from School of Mathematics and Statistics, Wuhan University, Wuhan, China in 2009. He is currently a professor of the School of Cyber Science and Engineering, Wuhan University, Wuhan, China. His main research interests include cryptography and information security, in particular, cryptographic protocols. He has published over 100 research papers in refereed international journals and conferences, such as IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Information Security and Forensic, and Usenix Security Symposium. He is the recipient of the 2018 IEEE Sysems Journal Best Paper Award and the 2019 IET Information Security Best Paper Award. His work has been cited more than 10000 times at Google Scholar. He is in the Editorial Board of several international journals, such as Journal of Information Security and Applications, Frontiers of Computer Science, and Human-centric Computing & Information Sciences.

Jianxin Li received the PhD degree in computer science from the Swinburne University of Technology, Australia, in 2009. He is an Associate Professor in Data Science and the director of the Smart Networks Lab at the School of Information Technology, Deakin University. His research interests include graph database query processing & optimization, social network analytics & computing, complex network data representation learning traffic, and personalized online learning analytics.

Na Lu received the B.S. degree in computer science from Wuhan University of Science and Technology in 2005, the M.S. degree in computer application technology from Wuhan University of Science and Technology in 2008. She is currently a lecturer with the School of Artificial Intelligence, Wuhan Technology and Business University. Her research interests include artificial intelligence and neural network applications.

Xuejiang Wei worked with the School of Logistics, Wuhan Technology and Business University as an Associate Professor. His current research interests include the Internet of things and smart logistics.