This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3341811

FLOW: A Robust Federated Learning Framework to Defend Against Model Poisoning Attacks in IoTs

Shukan Liu, Zhenyu Li, Qiao Sun, Lin Chen, Xianfeng Zhang, Li Duan

***Abstract*—Federated Learning (FL) is a promising distributed learning approach to enable intelligent Internet of Things (IoT) applications. However, FL is vulnerable to model poisoning attacks in which malicious clients abate the accuracy of the global model by committing crafted local model updates to the server. Existing defense methods either rely on a validation dataset or simply remove the detected malicious clients from the subsequent training process to handle attacks from a large number of malicious clients. Thus, the performance of existing methods deteriorates drastically in many scenarios where the data distributions of clients are unpredictable. To address these deficiencies, we propose a framework called FL overwatch (FLOW) to efficiently defend against model poisoning attacks taking advantages of the local model updates in current and historical training iterations. On one hand, FLOW detects malicious clients in each iteration by measuring the cosine distances between the local model updates of clients, such that malicious updates are eliminated from the current aggregation. On the other hand, FLOW gracefully punishes the previously identified malicious clients rather than removes them from the whole training process. As a result, FLOW can embrace a richer reliable set of local model updates than existing methods in aggregation. Extensive experiments on widely-used benchmark datasets show that FLOW can achieve higher success defending ratio and higher accuracy of global models over existing Byzantine-robust FL methods under typical untargeted attacks and targeted attacks. Furthermore, FLOW also shows significant effectiveness in defending against adaptive attacks tailored to FLOW.**

***Index Terms*—Federated learning (FL), model poisoning attacks, Internet of Things, model aggregation.**

I. Introduction

With the rapid development of fifth generation (5G) mobile communication networks and the Internet of Things (IoTs), a large number of sensors and devices are connected to the network. These devices generate unprecedented amounts of data, accelerating the widespread application of artificial intelligence (AI) in IoTs [1]. Many intelligent IoT applications including autonomous driving, smart healthcare, smart cities, and smart manufacturing have rapidly flourished. The diverse needs of these IoT applications have put new demands on learning methods for AI models. Some applications, such as smart medical care, involve sensitive user data. Traditional centralized learning methods are difficult to apply due to privacy issues [2]. Some other applications are time-sensitive, such as automatic driving, and need to be deployed at the edge of the network to meet the latency requirements of services [3].

To overcome these challenges, IoT researchers tend to study federated learning (FL) in recent years. Under an FL architecture, IoT devices can be employed to train local AI models and then upload them to a central server for aggregating the weights of a global model [4]–[8]. However, FL systems face cyberattacks that can severely endanger the security of applications, especially when the security capabilities of IoT devices are weak. These devices can easily be compromised by attackers and become sources of attacks for involved FL-based AI models [3]. One of the most typical and harmful types of such attacks is model poisoning attacks (MPAs). By manipulating the data output or learning processes of IoT clients, an MPA attacker can degrade the performance of the global model by generating malicious local models and sending them to the central server during the model integration process [9]–[13]. Therefore, a robust FL framework to defend against MPAs has become an urgent need in the industry.

Generally, MPAs can be classified into *untargeted attacks* [10], [11], [13] and *targeted attacks* [9], [14]. In untargeted attacks, the deteriorated global model may arbitrarily make incorrect predictions for any testing examples. In targeted attacks, the deteriorated global model makes incorrect predictions on testing examples specified by the attack, while the accuracy of the global model on other non-target testing examples remains unaffected. There are many Byzantine-robust FL methods for defending against model poisoning attacks. However, most of them can only defend against attacks from a small number of malicious clients [10] or rely on a validation dataset to identify malicious clients [15], [16]. Although FLDetector [17] can detect a large number of malicious clients, it may lower the upper bound of performance of the global model due to the loss of datasets on falsely identified benign clients and resumed malicious clients [1].

In this paper, we propose a new method called FL OverWatch (FLOW) to defense against attacks from a large number of malicious clients. Firstly, FLOW detects malicious clients by measuring the mutual distances between the local model updates of clients, rather than using an ideal validation dataset in the server. Secondly, FLOW punishes the malicious clients by lowering their trust score to preserve the benign model updates as much as possible. Different from existing methods, we conditionally allow the malicious clients detected in an iteration to join the subsequent training iterations rather than simply eliminate it. In this way, the local model updates of misjudged benign clients can still contribute to the global model.

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant No.62372073 and Grant No.62201605.

All authors are with the School of Electronic Engineering, Naval University of Engineering, Wuhan, 430033, China (e-mail: liusk@seu.edu.cn; lizhenyu1165@hotmail.com; sunqiao_zju@zju.edu.cn; forever_chenlin@126.com; dyzhbzhang@126.com; duanlidragon@126.com). *(Corresponding author: Li Duan.)*

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on January 25,2024 at 08:54:33 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3341811

Specifically, we record the long-term behaviors of clients by assigning a suspiciousness level for each client, which is raised or reduced according to the detection result of the client in each iteration. Each client has a trust score that declines exponentially with the growth of its suspiciousness level. In each iteration, FLOW aggregates the local model updates of benign clients by weighing their trust scores and suppresses the local model updates from the malicious clients identified in the current iteration. As a result, the influence of malicious clients is almost invalidated in the long run.

We evaluate the proposed FLOW on three typical benchmark datasets, including MNIST, Fashion-MNIST [18], and CIFAR-10 [19]. We compare FLOW with FedAvg [4] and five Byzantine-robust FL methods, including Trimmed-mean [13], Median [13], Krum [10], FLTrust [15], and FLDetector [17], the state-of-the-art works. We conduct three typical untargeted model poisoning attacks, including Trim attack [13], Min-Max attack and Min-Sum attack [11], and two widely-used targeted model poisoning attacks, including Scaling attack [9] and DBA [14] to contaminate all the FL methods. Our experimental results show that FLOW outperforms existing Byzantine-robust FL methods in defending against both targeted attacks and untargeted attacks. Moreover, we design an adaptive attack [11] for FLOW and evaluate the robustness of FLOW against the adaptive attack. The experimental results show that the accuracy of FLOW merely degrades 1.3-1.7% in defending against the adaptive attacks. Thus, FLOW can properly defend against the adaptive attacks.

In summary, the main contributions of the paper include:

- We present FLOW, a robust FL framework against untarget and targeted model poisoning attacks to safeguard IoT applications. This framework introduces a screening mechanism of local models to punish suspiciousness clients by lowering their participation in model aggregation and prevent potential attackers.
- We demonstrate a new approach of detecting and punishing malicious clients taking advantages of the historical behaviors of clients. This approach incorporates the trust scores of clients to construct a soft decision mechanism, which can punish suspiciousness clients while effectively avoid the model loss caused by misjudgment.
- We empirically evaluate FLOW against both three typical untarget attacks and two targeted attacks. Experimental results show that FLOW excels state-of-the-art Byzantine-robust FL methods in defending against the attacks.
- We design adaptive attacks against FLOW and evaluate the performance. Experimental results show that FLOW is robust against the adaptive attacks.

The remainder of this paper is organized as follows. Section II introduces the background and motivation. Section III defines the attack models and defense objectives of FLOW. Section IV presents the design of FLOW. Section V gives the evaluation results and Section VI concludes the paper.

II. Background and Motivation

A. Federated Learning in IoT Applications

Federated learning [2], [20], [21] provides a systematical framework with shared global model on decentralized data. Its privacy-preserving property makes FL increasingly used in Internet of Things (IoT) [22]–[24], such as industrial control [25], healthcare [26], intrusion detection [27], DDoS attacks [28], [29], and autonomous driving [30]. The vanilla FL meets the need of constructing an intelligent IoT application framework by periodically collecting and aggregating model updates from distributed IoT clients without data exchanging [31]–[33]. In the mean time, the characteristics of the global model are shared with every client to integrate the global model into their local models.

Specifically, the typical primal step of FL, such as FedAvg [4], is synchronizing the global model with clients. Then, each client fine-tune the global model using its local training dataset to train a local model. The clients feed back their local model updates to the server. Finally, the server computes a global model update via aggregating the local model updates according to certain aggregation rules. Due to its distributed nature, traditional federated learning methods could hardly distinguish the authenticity and reliability of the model updates committed by clients. Thus, it is particularly important to ensure the credibility of federated learning in IoT applications, especially in defending against poisoning attacks [34], [35].

B. Poisoning Attacks Against FL

Poisoning attacks refer to attacks launched during the model training phase of machine learning. One category is data poisoning attacks [36], [37], which cannot directly compromise the model parameters sent to the server, but instead corrupt the global model by deleting, adding, or removing data from the local training dataset. Another powerful category of attacks is model poisoning attacks (MPAs), where the attacker directly manipulates the model updates sent to the server [36]. There are many MPAs for FL in IoT scenarios [38]–[40]. Generally, MPAs can be classified as targeted model poisoning attacks [9], [14], [41], [42] and untargeted model poisoning attacks [11], [43]–[45]. We focus on MPAs in this paper.

Untargeted model poisoning attacks. Untargeted model poisoning attacks aim to deteriorate the global model by sending arbitrary gradients or model parameters to the server such that the accuracy of the global model is low for indiscriminate testing inputs. Fang et al. [44] presents a generic framework for poisoning local models, which is usable for optimizing attacks upon any aggregation rule. The framework formulates an untargeted attack as an optimization problem that aims at maximizing the difference between the aggregated model updates before and after the attack. The framework has been applied to optimize many model poisoning attacks, including Krum [10] (called *Krum attack*), *Trim-mean* [13], and Median [13] (called *Trim attack*).

Shejwalkar et.al. [11] also propose a universal framework for model poisoning attacks in FL. In contrast to Fang [44], the framework assumes that the attack has the knowledge of benign clients and the aggregation algorithm used by the server. Based on the framework, [11] designs two untargeted attacks, including *Min-Max attack* and *Min-Sum attack*. Min-Max attack assures that the malicious gradients are close to the clique of the benign gradient. Min-Sum attack tries to limit the

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on January 25,2024 at 08:54:33 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3341811

distance between malicious gradients from benign gradients within the distances between any benign gradients.

Targeted model poisoning attacks. Targeted model poisoning attacks (also known as backdoor attacks), strive to achieve a high global accuracy while simultaneously manipulating the global model to predict a targeted label for any testing examples embedded with a trigger specified by the attacker. *Scaling attack* [9] embeds triggers with attacker-selected target labels into the training examples and adds them to the malicious client's local training data. The model parameters with the backdoor are then uploaded to the server as local model updates after scaling them to elaborate multiples. This allows the global model to be close to the backdoor model when aggregated on the server side. *Distributed Backdoor Attack (DBA)* [14] disassembles the trigger into distinct local patterns and mixes them into the training data of different groups of malicious clients.

C. Byzantine-Robust Aggregation Rules

The aggregation rule of FedAvg [4] has been widely used in FL. However, the aggregation rule of FedAvg is not robust that can even be manipulated by a single malicious client [10], [13], [42]. Thus, many Byzantine-robust aggregation rules of FL have been proposed to defend against poisoning attacks.

Trimmed-mean [13]: In Trimmed-mean, the server first separately sorts the n values of each individual parameter of the local model updates. Then, the server removes the i ($i < \frac{n}{2}$) largest updates and the i smallest updates of each model parameter. Finally, the server utilizes the average of the rest of $n-2i$ updates as the final update of the corresponding parameter of the global model.

Median [13]: Similar to Trimmed-mean, the server in Median also sorts the n values of each individual parameter of the local model updates separately. Then, the server selects the median value of each parameter as the final update of the corresponding parameter of the global model.

Krum [10]: Krum first calculates the Euclidean squared distances between the current local model updates and the most recent $n-i-2$ model updates, where n is the total number of clients and i is the number of malicious clients. Then, the local model update with the smallest distance is used as the global model update.

FLTrust [15]: FLTrust assigns scores to each client according to a validation dataset, called "root dataset", which is preserved by the server. The score consists of two components. One is the cosine similarity between the local model update and the server model update calculated using the root dataset. The other is the normalization of the local model update to the same sphere as the server model update. The lower the score is, the lower the weight of the local model update is in aggregation. However, in many real-world scenarios [1], [46], it is nontrivial for servers to preserve such a validation dataset. Moreover, it does not perform well when the validation dataset is non-independently and identically distributed with the overall training dataset.

FLDetector [17]: FLDetector proposes to detect malicious client by checking the consistency between each real local model updates and a corresponding predicted model update. The predicted model update is calculated by approximating the Hessian matrix obtained by the gradients and parameters of historical local model updates. The detected malicious clients are removed from the following trainning iterations.

D. Motivation

Basically, the fundamental idea for defending poisoning attacks on FL is to find out the malicious clients [15], [17], [47]. Among existing works, FLTrust [15] is one of the most robust solutions against a large number of malicious clients. We use FLTrust as the motivational example. FLTrust needs to preserve an ideal validation dataset in advance as the foundation of its defense. It is a pity that in many real scenarios, such as privacy-sensitive environments [46] and network attack detection [1], it is non-trivial for the server to possess such a validation dataset.

In this section, we try to measure the defense ability of FLTrust in such real scenarios. First, we configure one hundred clients, including twenty malicious clients and eighty benign clients, to run the CIFAR-10 [19] benchmark in the evaluation. Then, we set a baseline by training FedAvg [4] with all the benign clients (denoted by "FedAvg w/o attacks") , i.e., no attacks in this case [17]. Finally, we conduct trim attacks on FedAvg (denoted by "FedAvg-Trim") and FLTrust [15] without a validation dataset (denoted by "FLTrust-Trim w/o validation") using the malicious clients.

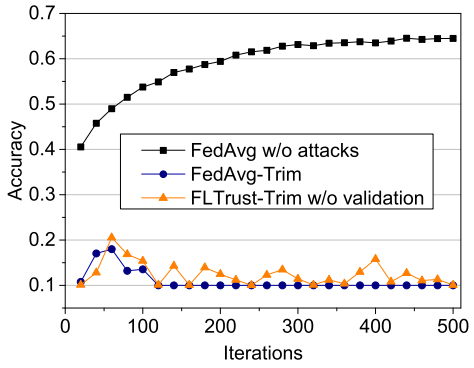


Fig. 1. Performance of FLTrust without root dataset against trim attack.

Figure 1 shows the experimental results. With the increased number of iterations, the accuracy of FedAvg without attacks, which is the ideal case, consistently grows from 0.405 to 0.645. On the contrary, the accuracy of "FLTrust-Trim w/o validation" is stuck in the range of 0.100 to 0.205, which is much lower than that of the baseline. Moreover, at the 500th iteration under trim attacks, the accuracy of FLTrust without the validation dataset even converges to 0.101, which is the almost the same as FedAvg. Such experimental results show that FLTrust loses its robustness in the case of without a preserved validation dataset. Hence, it is necessary to find a new solution for defending FL against poisoning attacks of malicious clients.

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on January 25,2024 at 08:54:33 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3341811

III. Problem Settings and Objective

Attack models. We follow the settings of attack models in previous works [9], [10], [13]–[15], [17]. We consider both untargeted and targeted model poisoning attacks in this work. Specifically, an attacker controls m malicious clients to launch attacks. Meanwhile, the server is not compromised by the attacker. Without loss of generality, we assume that there are no more than 50% clients are malicious [13], [17], [47]. The attacker has all the background knowledge about an FL system, including the local training data and model updates on the malicious clients, loss function, learning rate, and the aggregation rule. In each training iteration, the malicious clients try to corrupt the global model by transmitting manipulated local model updates to the server, while the benign clients reports the true model updates to the server. We also design an adaptive attack [11] for the proposed FLOW (Section IV-E).

Defense objectives. We aim to design a Byzantine-robust FL framework to defense against both targeted model poisoning attacks and untargeted model poisoning attacks without relying on a preserved validation dataset in the server. Generally, such an FL framework needs to fulfill the following goals: (1) *Robustness.* The framework should sustain the accuracy of the global model in the dealing with the model poisoning attacks of malicious clients. Specifically, not only the impact of arbitary model updates, but also the impact of backdoored model updates all need to be minimized in the aggregation of the global model. (2) *Efficiency.* The framework should identify benign and malicious clients without increasing the communication cost and the burden of clients since the communication and computing resources of clients in FL applications are generally highly limited.

IV. FLOW Overview and Design

A. FLOW Overview

In this paper, we propose Federated Learning OverWatch (FLOW) to defense against model poisoning attacks. The main idea of the proposed FLOW is to find out the malicious clients in each iteration and punish the malicious clients according to their historical behaviors. FLOW needs to handle two major issues, i.e., find out the malicious clients as much as possible and punish the malicious clients. The framework of FLOW is shown in Figure 2. In FLOW, the server is responsible for detecting and punishing malicious clients. Different from FLTrust [15], the server in FLOW does not rely on a validation dataset to detect the malicious local model updates. Hence, FLOW detects the malicious clients by measuring the differences between the updates of local models.

In general, FLOW performs three particular procedures in each iteration to defend the attacks. First, the server calculates the mutual distances of the local model updates. Second, the server divides the clients into two classes according to their mutual distances obtained in the first procedure. The server recognizes the clients in the smaller class as malicious. It is because that the attacker controls less than half of all the clients, as mentioned in Section III. Finally, the server aggregates the local model updates using the aggregation rule. Different from existing FL methods, FLOW considers both the current and the past malicious suspicion of clients in the current aggregation.

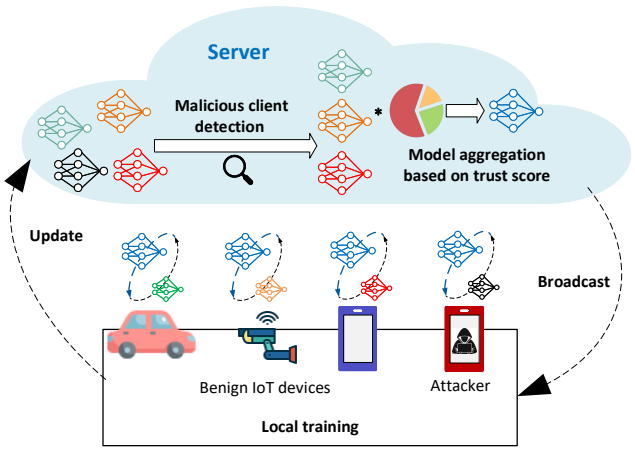


Fig. 2. Framework of FLOW.

Specifically, the server discards the local model updates of the malicious clients identified in the current iteration. For the benign clients, the server punishes them according to their past malicious suspicions. A benign client in the current iteration may be recognized as malicious client in the previous iterations in two situations. On one hand, an actual benign client may be classified as malicious client due to the statistical heterogeneity in training data or Byzantine faults. On the other hand, an actual malicious client may be recognized as a benign client in certain iterations due to clustering error, holding attacks, etc. For the clients that have been marked as malicious, we cannot either fully trust them, or simply abandon their local model updates. Hence, we propose to punish such clients by setting a *suspiciousness level* for each client in FLOW (see Section IV-C). The server aggregates the local model updates along with a set of *trust scores* calculated by the suspiciousness levels of the corresponding benign clients.

B. Malicious Client Detection

The crafted local model updates committed by the attacking malicious clients are expected to be separated from the true local model updates. Furthermore, as aforementioned, the number of malicious clients is no more than the number of benign clients. Thus, it is promised to find out the malicious clients by clustering the clients according to their local model updates and determining the smaller set of clients.

Mutual distances between clients. Before clustering the clients, we measure the variance among them. Several existing FL defense methods [47] directly use local model parameters to filter the malicious clients. Such methods may fail to detect certain carefully crafted attacks. For example, each client uses its local training dataset to train a local model (denoted as $\boldsymbol{\omega}_i$) by fine-tuning the global model. As shown in Figure 3, the cosine similarity between malicious model parameters $\boldsymbol{\omega}_3$ and the current global model $\boldsymbol{\omega}_r$ is the same with the benign model parameters $\boldsymbol{\omega}_1$. However, the trend of the local model updates of $\boldsymbol{\omega}_3$ largely deviates from those of the benign clients

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on January 25,2024 at 08:54:33 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3341811

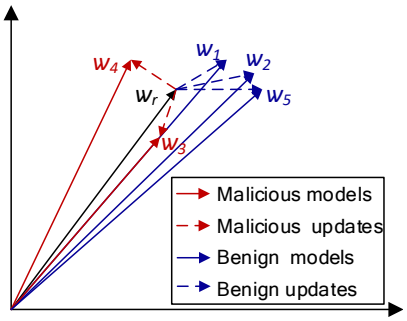


Fig. 3. Illustration of local model parameters.

$\boldsymbol{\omega}_1$, $\boldsymbol{\omega}_2$, and $\boldsymbol{\omega}_5$. Simply aggregating the local model updates of $\boldsymbol{\omega}_3$ will greatly reduce the accuracy of the global model.

Figure 3 shows that the model updates can clearly reveal the difference in model parameters between such malicious clients and benign clients. Therefore, FLOW measures the mutual distance c_{ij}^r between client i and client j in iteration r using the cosine distance between their local model updates:

$$c_{ij}^r = 1 - \frac{\boldsymbol{g}_i^r \cdot \boldsymbol{g}_j^r}{\|\boldsymbol{g}_i^r\|_2 \|\boldsymbol{g}_j^r\|_2}, \tag{1}$$

where $\boldsymbol{g}_i^r$ and $\boldsymbol{g}_j^r$ are the local model updates calculated by client i and client j in iteration r, respectively. The local model updates $\boldsymbol{g}_i^r$ of client i in iteration r are calculated by Algorithm 1. We iteratively use Equation (1) to calculate the pairwise cosine distances for the n clients in iteration r, denoted by $C^t = \{c_{11}^r, c_{12}^r, ..., c_{nn}^r\}$. $\mathcal{B}$ is the set of batches of local training data.

Algorithm 1: LocalUpdate

Input: $\boldsymbol{\omega}$ is the current global model; D is the training datasets on the client; b is the batch size; η is the learning rate; e is the number of local epochs.

Output: Model updates.

```
ω' ← ω;
B ← (split D into batches of size b);
for i = 1, 2, ..., e do
    for batch b ∈ B do
        ω' ← ω' − ηδ(ω'; b);
    end
end
return ω' − ω;
```

***k*-means based malicious client detection.** In an iteration t, once the server figures out the mutual distances C^t of all the clients, the server performs malicious client detection via the k-means algorithm [48], where C^t is the input. We use the Gap statistics [49] to analyze the number of possible clusters. If the number of clusters equals one, we treat all the clients as benign clients. Otherwise, if the possible number of clients is more than one, we divide the n clients into two sets, i.e., classifying the clients to two categories. The clients in the smaller set are marked as malicious clients.

C. Punish-based Aggregation Rule

In face of malicious clients, an instinct idea is to evict the malicious clients from the training of the subsequent iterations to eliminate their damage to the global model [17]. However, as discussed in Section IV-A, a benign client may be misjudged as malicious (false positive) or an actual malicious client may contribute real model updates for missed detection (false negative) or even causing the model to fall under the control of the attacker. Directly abandoning the malicious clients found in one iteration may result in the loss of real local model updates. In FLOW, we propose a graceful aggregation rule that considers the long-run credibility of clients rather than simply evict the malicious clients in the whole subsequent training iterations.

Punishment mechanism. Specifically, we design a weighted aggregation rule based on the trustworthiness of the clients. For the occasional exceptions, we gracefully conduct a light punishment on the local model updates of the corresponding client. For the recidivism clients, we enforce an exponentially increased punishment on their local model updates. Basically, we set a suspiciousness level l_i^r for each client i to record its accumulative malicious behaviors after r iterations. The suspiciousness level l_i^r is calculated as follows:

$$l_i^r = \begin{cases} 0, & r = 1; \\ l_i^{r-1} + 1, & \text{if } c_i \text{ is marked as malicious;} \\ l_i^{r-1} - 1, & \text{if } c_i \text{ is marked as benign.} \end{cases} \tag{2}$$

In each iteration r, if a client is classified as a malicious client, we raise its suspiciousness level by one; otherwise, we decrease its suspiciousness level by one. Based on the suspiciousness level of each client, for each client c_i in iteration r, we assign a *trust score* s_i^r as follows:

$$s_i^r = x_i^r \beta^{l_i^r}, \tag{3}$$

where x_i^r is a binary defined by

$$x_i^r = \begin{cases} 0, & \text{if } c_i \text{ is malicious in iteration } r; \\ 1, & \text{if } c_i \text{ is benign in iteration } r. \end{cases} \tag{4}$$

β in (3) is a parameter belonging to $(0, 1]$. Since l_i^r is an integer that ranges from 0 to R, the total number of training iterations, the value of the trust score is in $[0, 1]$ in each iteration. The higher the suspiciousness level of a client is, the smaller the trust score of the client. In the beginning of training, we initialize the suspiciousness level l_i^0 of client i to 0, indicating that the client is trustworthy and there is no need to punish it.

Aggregating the local model updates. In each iteration, the server aggregates the local model updates according to their historical behaviors. For the malicious clients detected in the current period, the server directly discards their local model updates and record their suspiciousness behavior. For the benign clients in the current iteration, the server aggregates their weighted local model updates to obtain the global parameters. We normalize the trust score of each benign client as its weight in aggregation. Then, we aggregate the local model updates by

$$\boldsymbol{\omega}_i^r = \boldsymbol{\omega}_i^{r-1} + \frac{1}{\sum_i^\tau s_i^r} \sum_i^\tau (s_i^r * \boldsymbol{g}_i^r), \tag{5}$$

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on January 25,2024 at 08:54:33 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3341811

where $\boldsymbol{\omega}_i^r$ and $\boldsymbol{\omega}_i^{r-1}$ are the global model parameters in the current and previous iterations, respectively. τ is the number of clients in iteration r.

Algorithm 2: FLOW

Input: n clients with local training datasets $D_i, i = 1, 2, ..., n$; $\boldsymbol{\omega}^0$ is the initial global model parameter; τ is the number of clients sampled in each iteration; R is the number of global training iterations.

Output: Global model $\boldsymbol{\omega}^R$.

1 Initialize the suspiciousness levels of all the clients to 0;
2 **for** *each global iteration* $r = 1, 2, ..., R$ **do**
3 Server sends $\boldsymbol{\omega}^{r-1}$ to the randomly selected τ clients;
4 **for** *each client* $i, i = 1, 2, ..., \tau$ **do**
5 $\boldsymbol{g}_i^r$ = LocalUpdate($\boldsymbol{\omega}^{r-1}$, D_i);
6 Send $\boldsymbol{g}_i^r$ to the server;
7 **end**
8 $(c_{11}^r, c_{12}^r, ..., c_{\tau\tau}^r) \leftarrow CosineDistance(\boldsymbol{g}_1^r, \boldsymbol{g}_2^r, ..., \boldsymbol{g}_\tau^r)$;
9 Determine the possible number k of clusters by Gap statistics;
10 **if** $k > 1$ **then**
11 $Clustering((c_{11}^r, c_{12}^r, ..., c_{\tau\tau}^r), k)$;
12 **end**
13 **if** *client* c_i *is in the largest cluster or the sole cluster* **then**
14 $l_i^r \leftarrow l_i^{r-1} - 1$;
15 **end**
16 **else**
17 $l_i^r \leftarrow l_i^{r-1} + 1$;
18 Drop client c_i from the aggregation of iteration r;
19 **end**
20 $\boldsymbol{\omega}^r = \boldsymbol{\omega}^{r-1} + \frac{1}{\sum_j^\tau s_j^r} \sum_i^\tau (s_i^r * \boldsymbol{g}_i^r)$;
21 **end**
22 **return** $\boldsymbol{\omega}^r$;

D. Complete FLOW Algorithm

Algorithm 2 shows the complete algorithm of the proposed FLOW. Generally, we follow the basic FL framework, similar to other FL methods, to conduct R iterations of training in which each iteration consists of three steps.

- **Step I (line 3):** the server distributes the current global model to a randomly selected subset of clients.
- **Step II (line 4-7)**, the selected τ clients train their local model to obtain the corresponding local model updates by using $LocalUpdate(\boldsymbol{\omega}^{r-1}, D_i)$, i.e., Algorithm 1. Then, the τ clients send their local model updates to the server.
- **Step III (line 8-20)**, the server first detects the malicious clients. The clients in the largest cluster or the sole cluster are regarded as benign clients. Then, the server updates the suspiciousness levels of the selected clients and calculate their trust score. Finally, the server aggregates the local model updates of the benign clients in the current iteration using Equation (5).

Figure 4 shows an example of the local model aggregation using the proposed FLOW algorithm. Given five local model updates $\boldsymbol{g}_1$, $\boldsymbol{g}_2$, $\boldsymbol{g}_3$, $\boldsymbol{g}_4$, and $\boldsymbol{g}_5$, we first calculates the mutual cosine distances between them. Then, we determine the proper number of clusters and classify the local updates into two clusters. The corresponding clients in the smaller cluster are regarded as malicious ones. Finally, we aggregate the benign updates using the corresponding trust scores as weights.

Time complexity analysis. The time complexity of FLOW for the server could be presumed according to the procedures in Step III. First, the pairwise cosine distance is $O(\tau^2 p)$, where n is the number of selected clients and p is the number of parameters of the model. The total complexity of Gap statistics and k-means algorithm is $O(\tau^2 BM)$, where τ is the number of sampled data volumes (i.e., the number of clients), B and M are the maximum number of sampling and number of clusters, respectively. The time complexity of the aggregation of model updates is $O(\tau p)$. Therefore, the overall time complexity of FLOW on the server is $O(\tau^2 p + \tau^2 BM + \tau p)$, which can be refined as $O(\tau^2(p + BM))$.

E. Adaptive Attacks

When an attacker knows that our FLOW is deployed to defend against attacks, the attacker can adapt its attacks to FLOW. Thus, we design a strong adaptive attack to FLOW using the methods proposed in [11], which is the state-of-the-art approach for optimizing the local model poisoning attacks on a given aggregation rule. Briefly, FLOW measures the mutual distances of clients by calculating pairwise cosine distances, generating a subset S_b of benign clients according to the mutual distances, and aggregates the local model updates of the clients in S_b with a trust score.

To maximize the damage on the global model, we design an adaptive attack for FLOW by shipping all the m malicious clients into the subset S_b of benign clients, through which the impact of malicious updates $\nabla_{\{i\in[m]\}}^m$ can be maximized in aggregation. In each iteration, the objective of the attacker is:

$$\begin{aligned} \underset{\gamma}{\operatorname{argmax}}\, m &= |\{\nabla \in \nabla_{\{i\in[m]\}}^m | \nabla \in S_b\}| \\ \nabla_{\{i\in[m]\}}^m &= \nabla^b + \gamma \nabla^p; \quad \nabla^b = f_{avg}(\nabla_{\{i\in[n]\}}), \end{aligned} \tag{6}$$

where ∇^b is a reference benign aggregate, ∇^p is the perturbation vector, and γ is the scaling coefficient of ∇^p. A possible ∇^b is the average of benign updates, denoted by f_{avg}. $\nabla_{\{i\in[n]\}}$ is the set of benign model updates that the attacker knows. n is the number of benign clients.

Algorithm 3 shows the optimization of γ. Specifically, we first initialize γ with a large value. Then, we use FLOW to handle the attacks and check whether all the malicious clients are successfully sneaked into the subset S_b of benign clients. If all the malicious clients are recognized as benign clients, we update γ_{succ} and scale up γ; otherwise, we shrink γ. We repeat this process until we find the most impactful γ_{succ} that satisfies the threshold θ, which is a preset parameter for controlling

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on January 25,2024 at 08:54:33 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3341811

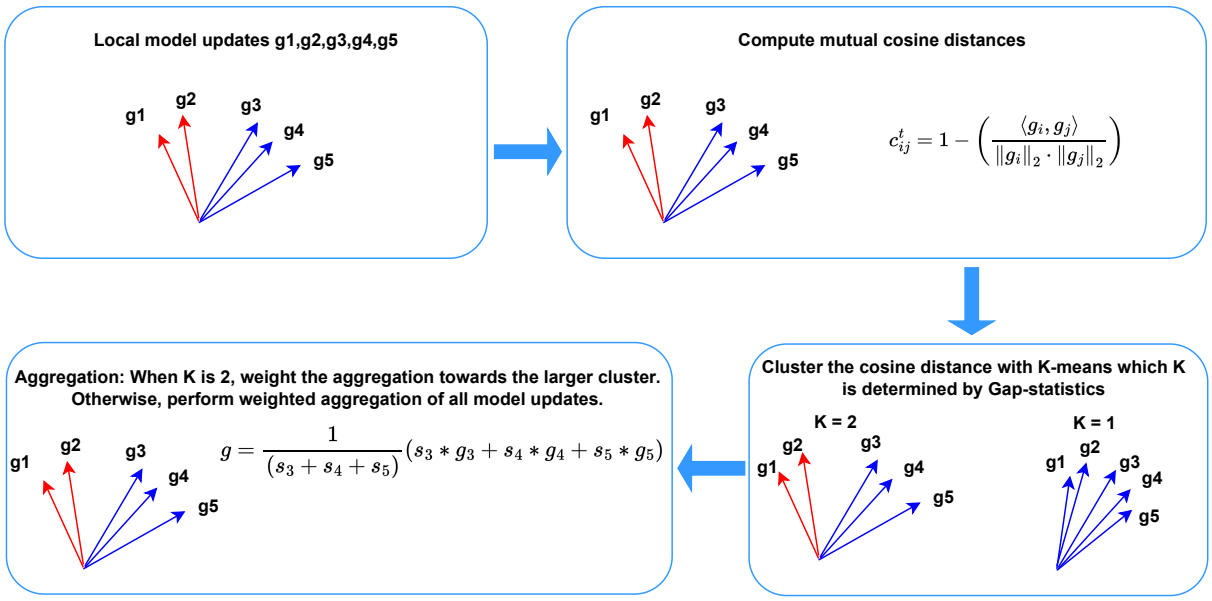


Fig. 4. Illustration of the aggregation rule of FLOW, which is applied in each iteration of training.

Algorithm 3: Algorithm to optimize γ.

Input: γ_{init} is the initial γ; θ is the threshold; O is an oracle; $\nabla_{\{i\in[n]\}}$ is the set of benign models that the attacker knows.

Output: γ_{succ}.

$step \leftarrow \gamma_{init}$, $\gamma \leftarrow \gamma_{init}$, $\gamma_{succ} \leftarrow 0$;
while $|\gamma_{init} - \gamma| > \theta$ **do**
  if $O(\nabla_{\{i\in[n]\}}, \gamma)$==*True* **then**
    $\gamma_{succ} \leftarrow \gamma$;
    $\gamma \leftarrow (\gamma + step/2)$;
  end
  else
    $\gamma \leftarrow (\gamma - step/2)$;
  end
  $step = step/2$;
end

the oscillation between the minimum and maximum γ values. It is notable that the adaptive attack cannot success in every iteration. According to Algorithm 3, if $|\gamma_{init} - \gamma|$ grows larger than the threshold θ while γ_{succ} is 0 in a certain iteration, it means that the attack in the iteration is failed.

V. Performance Evaluation

A. Experimental Setup

1) Datasets: We use the widely-used datasets MNIST, Fashion-MNIST [18], and CIFAR-10 [19] to evaluate the performance of FLOW. MNIST and Fashion-MNIST both consist of 60,000 training examples and 10,000 testing examples. CIFAR-10 includes 50,000 training examples and 10,000 testing examples. Each dataset has ten categories for images classification. In the evaluation, we distribute the training examples of the datasets to the clients, just like prior works [15], [44]. Suppose there are M classes in a dataset, we randomly divide the clients into M groups. Given a training example labelled with a, we assign it to group a with probability $q > 0$ and to any other group with probability $\frac{1-q}{M-1}$. The data are uniformly distributed to the clients within the same group. In this case, q actually determines the distribution difference of the local training data of clients. When $q = 1/M$, the data on the clients is independent and identically distributed (IID); otherwise, the data is non-IID. Moreover, the larger q is, the higher degree of non-IID among the local training data is on the clients. In the evaluation, we set the value of q to 0.5 by default [15]. We use CNN [50] as the global model for MNIST and Fashion-MNIST, and ResNet20 [51] as the global model for CIFAR-10, respectively.

TABLE I
The default parameter settings of the FL systems.

	Explanation	MNIST	Fashion-MNIST	CIFAR-10
n	# clients	100		
model	used model	CNN	CNN	ResNet20
F	# features	784	784	1024
τ	# selected clients	100		
R_l	# local iterations	1		
R	# global iterations	500		
b	# batch size	32	32	64
m	# malicious clients	20%		
η	# learning rate	0.01		
bias	Non-IId degree	0.5		

2) FL settings: We compare the proposed FLOW with six typical FL methods: FedAvg [4], Trimmed-mean [13], Median [13], Krum [10], FLTrust [15], and FLDetector [17]. We use FedAvg as the baseline to demonstrate the impact of attacks without defense. Note that the validation dataset of FLTrust is invalid in the evaluation since we are considering the application scenarios without prior knowledge of the overall data distribution.

The default FL system settings of parameters are shown in Table I. Basically, we set one hundred clients in the FL

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on January 25,2024 at 08:54:33 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3341811

TABLE II
PERFORMANCE OF CNN GLOBAL MODELS ON MNIST.

	Trim attack	Min-Max attack	Min-Sum attack	Scaling attack	DBA
FedAvg	94.3	98.0	99.0	99.2/99.96	99.2/100.0
Trimmed-mean	96.5	98.1	98.9	99.2/99.9	99.1/11.0
Median	97.4	98.6	98.8	99.0/99.5	99.0/0.4
Krum	95.0	9.8	95.2	95.2/0.8	95.3/0.6
FLTrust	92.2	90.4	91.1	97.6/0.3	97.6/0.3
FLDetector	96.8	96.7	96.8	97.2/0.4	97.4/0.4
FLOW	**99.1**	**99.1**	**99.0**	**99.1/0.1**	**99.1/0.1**

TABLE III
PERFORMANCE OF CNN GLOBAL MODELS ON FASHION-MNIST.

	Trim attack	Min-Max attack	Min-Sum attack	Scaling attack	DBA
FedAvg	74.7	85.3	88.7	90.9/99.3	90.9/99.3
Trimmed-mean	85.8	86.9	88.5	91.0/98.4	90.9/92.7
Median	87.5	87.4	88.3	90.6/97.9	90.6/84.9
Krum	84.0	81.5	77.3	83.1/1.3	83.6/2.4
FLTrust	79.3	70.6	75.0	85.8/1.7	86.1/1.4
FLDetector	85.0	84.8	83.9	85.8/5.0	84.2/3.2
FLOW	**91.0**	**90.7**	**90.8**	**90.8/1.3**	**90.7/1.5**

system. All of the clients participate in each iteration of the training process, i.e., τ is 100. For all the three datasets, the number R_l of local training iteration on each client is 1 and the number R of the whole training iterations of the global model is 500. The learning rate η of the clients is set to 0.01. In local training, for MNIST and Fashion-MNIST, the batch size b is set to 32; for CIFAR-10, the batch size is set to 64.

In the evaluation, once FLDetector detects malicious clients in a certain iteration, the server bans the malicious clients in the entire following training iterations, restarts the FL training, and repeats pre-defined number of iterations. On the other hand, if FLOW detects malicious clients, the server will remove the clients marked as malicious and increase its suspiciousness level. In future training iterations, these clients may still join the training but will be punished by the server according to their trust scores calculated by the corresponding suspiciousness levels.

3) Attack settings: We attack all the FL methods by three untargeted model poisoning attacks, including Trim attack [13], Min-Max attack [11] and Min-Sum attack [11], and two targeted model poisoning attacks, including Scaling attack [9] and Distributed Backdoor Attack (DBA) [14]. For the Min-Max and Min-Sum attacks, we set the threshold θ to the value in [11], i.e., 1×10^{-5}. For the two targeted model poisoning attacks, we use their default settings of the trigger patterns and select *label 0* as the target label. The scaling parameters of both Scaling attack and DBA are set to 1 [17], which is much more stealthy than the configuration used in [15]. In the experiments, we randomly select 20% of the clients as the malicious clients. DBA needs to equally decompose the global trigger pattern into four parts and embed them into the local training data of four malicious client groups. Thus, the number of malicious clients is 20 for all the experiments. Specifically, all attacks occur in every iteration of FL training.

4) Evaluation metrics: For untargeted attacks, i.e., Trim attack, Min-Max attack, and Min-Sum attack, we show the accuracy of the global model in percentile that a larger value means a higher accuracy. For targeted attacks, i.e., Scaling attack and DBA, we consider two metrics for evaluating the effectiveness of targeted attack and defense methods as follows: (1) *Main Task Accuracy (MTA)*, which indicates the accuracy of a model in its main (benign) tasks [47]. MTA denotes the fraction of benign inputs for which the trained global model provides correct predictions. (2) *Attack Success Rate (ASR)*, which is the fraction of the trigger set for which the trained global model provides the attacker-chosen outputs. On one hand, the attacker aims to minimize the impact on MTA to reduce the chance of being detected. The defense system should also not negatively influence MTA. On the other hand, the attacker aims at maximizing ASR while an effective defense prevents the attacker from increasing it. In summary, for an FL defense approach, a higher MTA and a lower ASR means a better performance in defending against targeted attacks.

B. Performance of Global Models under Attacks

Table II, III, IV show the experimental results of different FL methods under different attacks on MNIST, Fashion-MNIST and CIFAR-10 datasets, respectively. For Scaling attack and DBA, the experimental results are denoted in the manner of "MTA/ASR". All the experimental results are represented in percentile.

The experimental results show that our FLOW achieves the robustness goal. FLOW not only achieves the highest model accuracy against the three typical untargeted model poisoning attacks, but also reaches least ASR against the two targeted model poisoning attacks on almost all the cases. The only exception is DBA on Fashion-MNIST, where the ASR of FLOW is 0.1% higher than that of FLTrust. However, FLOW actually removes all the malicious local model updates in each training process, just the same as FLTrust. Several existing techniques, including Median, Krum, FLTrust, and FLDetector show poor predictions on benign inputs under targeted attacks, i.e., Scaling attack and DBA. It is because

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on January 25,2024 at 08:54:33 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3341811

TABLE IV
PERFORMANCE OF RESNET20 GLOBAL MODELS ON CIFAR-10.

	Trim attack	Min-Max attack	Min-Sum attack	Scaling attack	DBA
FedAvg	10.0	13.4	38.2	65.3/94.0	64.1/95.7
Trimmed-mean	38.7	13.1	10.0	64.9/95.5	67.0/95.9
Median	35.2	10.0	10.0	59.1/98.1	52.1/94.3
Krum	31.7	10.0	10.0	29.5/23.0	29.3/26.7
FLTrust	39.4	23.5	23.8	49.0/7.1	46.9/2.5
FLDetector	53.8	47.8	54.0	53.2/21.8	54.0/16.9
FLOW	**64.6**	**65.9**	**64.9**	**65.8/2.5**	**65.9/1.8**

that their aggregation rules are misled by malicious model updates during training. Meanwhile, the MTA of FLOW is 4.6% higher than that of FLTrust, meaning that FLOW has a smaller negative impact on benign tasks.

In particular, on the CIFAR-10 dataset, the accuracy of FLOW is 25.2% higher than FLTrust (Trim attack in Table IV) and the ASR of FLOW is 19.3% lower than FLDetector (Scaling attack in Table IV). Moreover, under Trim attack, the model accuracy of different FL algorithms decreases to varying degrees, while the proposed FLOW maintains the model accuracy. Specifically, on the CIFAR-10 dataset, the model accuracy for FLOW is 64.6%, while the highest model accuracy for the other FL methods is 53.8%. Trim attack contaminates the global model by sending arbitrary gradients, which puts forward high requirements for identifying malicious clients. The reason why FLOW can well defend against Trim attack is that FLOW clusters the cosine distances among the local model updates to sift out malicious clients, such that the malicious clients do not participate in the aggregation.

As a baseline without defense, FedAvg is expected to show the worst performance under attacks. However, the experimental results show that FedAvg actually outperforms existing defensive FL methods on several cases. An instance is Krum under Min-Max attack and Min-Sum attack on CIFAR-10, where the accuracy of the global model is 3.4% and 28.2% lower than these of FedAvg, respectively. Krum selects a local model whose parameters are most similar to those of other local models, and uses the parameters of this local model as global parameters. Unfortunately, both Min-Max attack and Min-Sum attack aim to ensure that the malicious parameters stick closely to the centre of the benign parameters. As a result, Krum constantly chooses the malicious local model as the global model in the training iterations. An extreme instance is Krum under Min-Max attack on MNIST, where the accuracy of the global model is merely 9.8%. By tracing the experiment, we find that Krum precisely pick the local model updates of a malicious client to update the global model all the time.

Trimmed-mean and Median show similar appearance in several cases. Trimmed-mean obtains the global model parameters by removing some of the maximum and minimum values. Median uses the median as the updated global model parameter. While both of them can reduce the impact of abnormal value, none of them can prevent malicious clients from participating in the training. Therefore, both Trimmed-mean and Median have high ASR in targeted model poisoning attacks. For example, on both Fashion-MNIST and CIFAR-10 datasets, the ASRs of the Trimmed-mean and Median are 84.9% to 95.9% under targeted model poisoning attacks.

Compared with FLTrust and FLDetector, which are the most advanced defense methods, FLOW still has considerable advantages in performance. Specifically, FLTrust is based on the root dataset to identify malicious clients. However, this assumption is difficult to realize in reality. FLDetector identifies malicious clients by judging the consistency of model updates, while FLDetector only uses the distance between predicted updates and actual updates as the clustering criterion, which may fail to find out malicious clients after tens of iterations and misjudge the benign clients. Although FLTrust and FLDetector can effectively reduce the ASR, they suffer a large loss in model accuracy. For instance, under Scaling attack and DBA, the model accuracies of FLTrust and FLDetector are not as good as that of FedAvg.

C. Robustness Analysis

1) Resistance performance: Figure 5 shows the model accuracy and ASR of our proposed FLOW under different attacks as the proportion m of malicious clients increases from 10% to 40%. Under untargeted attack, the model accuracy shows a decreasing trend as the number of malicious clients increases, as Figure 5(a) shows. With the increase of the number of malicious clients, the proportion of malicious gradients is increasing, which greatly increases the difficulty of identifying malicious gradients. Although the model accuracy of our framework shows a decreasing trend under untargeted attacks, the loss of model accuracy under different proportions of malicious clients is less than 10%.

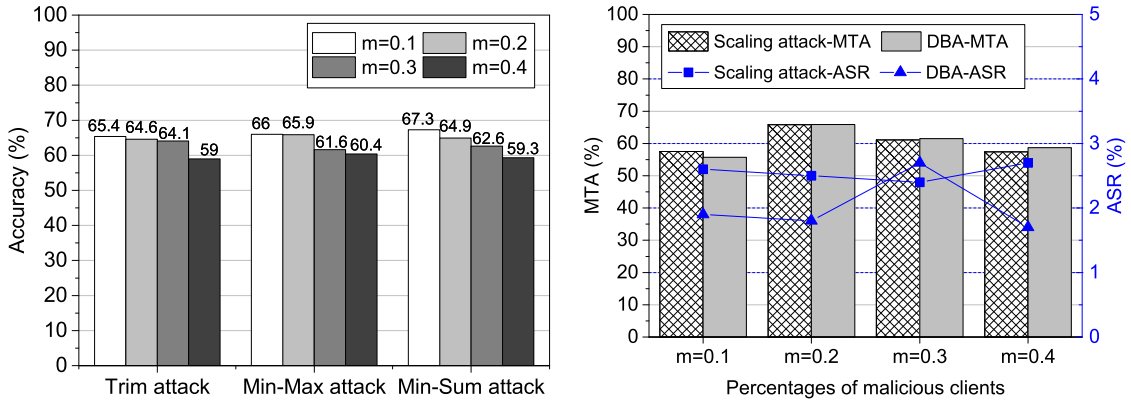
Experimental results in Figure 5(b) show that FLOW is robust to targeted attacks even if a large proportion of clients are malicious. Specifically, FLOW can maintain similar ASRs, all lower than 3%, even when the proportion m of malicious clients increases to 40%. It is because FLOW can accurately identify and reject malicious inputs, and radically punish the real malicious clients in the long run.

2) Generalization performance: In order to investigate the impact of different non-IID cases, we change the value of parameter q to verify the generalization performance of FLOW. The larger q is, the higher degree of non-IID is. The experimental results are shown in Figure 6. It can be observed that FLOW can still maintain the model accuracy and ASR similar to that in the case of IID (See $q = 0.1$) when the non-IID degree gradually increases to 0.7.

When the degree of non-IID q gradually increases from 0.5 to 0.9, it means that the probability of each label in the dataset being monopolized by a group of clients is greatly

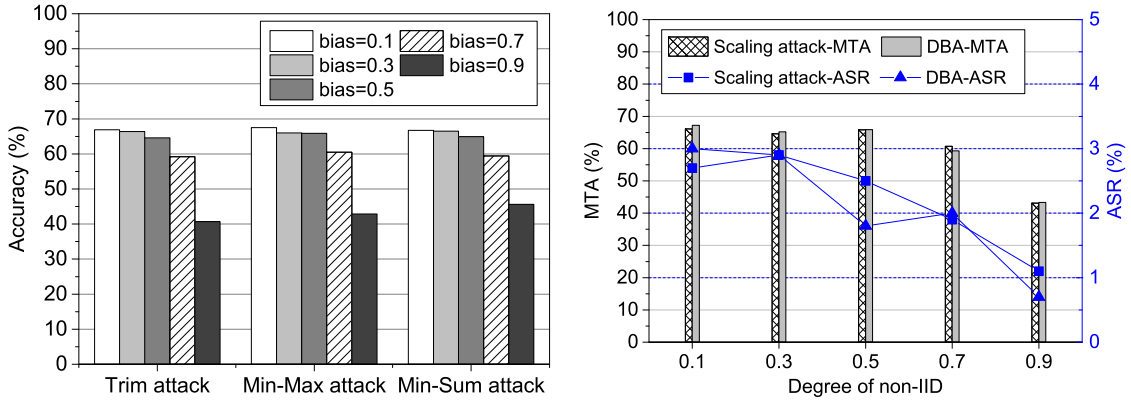
Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on January 25,2024 at 08:54:33 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3341811



(a) Accuracy under untargeted attacks. (b) MTAs and ASRs under targeted attacks.

Fig. 5. FLOW under untargeted attacks and targeted attacks with different proportions of malicious clients.



(a) Accuracy under untargeted attacks. (b) MTAs and ASRs under targeted attacks.

Fig. 6. FLOW under untargeted attacks and targeted attacks with different degrees of non-IID.

increased. We use MNIST, which has 10 labels of samples, as an example. Suppose the degree of non-IID q equals 0.9, it means that the data samples of a certain label i have a 90% probability of being allocated to group i, while the probabilities of the data samples of label i being allocated to the remaining groups are 1.1%. Such a data allocation will lead to huge data distribution deviation. When the clients in group i cannot participate in the training, the accuracy of the global model will be greatly reduced [36], [52].

Due to the huge variation between the data distribution of clients, it is more likely to wrongly recognize benign clients as malicious ones. Benign clients that are mistaken for malicious clients will be screened out and will not be able to participate in the model training, making it highly likely that the model will not learn enough about a certain label. That's why the MTA of FLOW decreases with the deepened of the degree of non-IID. Additionally, as the non-IID degree deepens, the ASR of targeted model poisoning attacks decreases. The diversity in data distribution among clients also makes it challenging for attackers to devise a universally effective attack method, which greatly reduces the ASR.

3) Effectiveness of punishment mechanism: We assess the proposed FLOW in terms of trust mechanism on CIFAR-10 dataset. FLOW without punishment refers to using FedAvg's averaging aggregation rule to perform aggregation. FLOW refers to use the trust score as the aggregation weight for each selected client. The experimental results are shown in Table V.

TABLE V
EFFECTIVENESS OF PUNISHMENT MECHANISM.

	FLOW	FLOW without punishment
Trim attack	64.6	64.6
Min-Max attack	65.9	65.9
Min-Sum attack	64.9	64.9
Scaling attack	65.8/2.5	64.5/3.4
DBA	65.9/1.8	63.6/3.1

For untargeted attacks, i.e., Trim attack, Min-Max attack, and Min-Sum attack, FLOW uses a clustering algorithm to filter out malicious clients in advance to make untargeted attacks invalid. Therefore, there is no difference in performance between FLOW and FLOW without punishment. For targeted model poisoning attacks, FLOW can achieve a higher model accuracy and a lower ASR. This is because FLOW can reduce the aggregate weight of malicious clients, which greatly reduces their impact on the global model. In addition, FLOW removes some malicious clients in advance through clustering algorithm, so it can maintain high performance even if FLOW without punishment.

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on January 25,2024 at 08:54:33 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3341811

D. Defence Efficiency under Adaptive Attacks

We use three types of perturbation vectors [11] to evaluate defence efficiency of FLOW under the adaptive attacks, including: 1) Inverse unit vector $\nabla^p_{uv} = -\frac{\nabla^b}{||\nabla^b||_2}$; 2) Inverse standard deviation $\nabla^p_{std} = -std(\nabla_{i\in[n]})$; 3) Inverse sign $\nabla^p_{sgn} = -sign(\nabla^b)$. In the perturbation vectors, ∇^b equals $f_{avg}(\nabla_{\{i\in[n]\}})$, where $\nabla_{\{i\in[n]\}}$ is the set of benign model updates. We measure the effectiveness of the adaptive attacks via three metrics: 1) Attack success times (AST); 2) Cosine similarity between the average of benign updates μ^b and malicious updates μ^m; 3) The accuracy of the global model after training.

TABLE VI
EFFECT OF ADAPTIVE ATTACKS.

	AST	Cosine similarity	Accuracy
∇^p_{uv}	66	0.83	64.3
∇^p_{std}	43	0.43	64.6
∇^p_{sgn}	1	0.87	64.7
No attack	-	-	66.0

We conduct adaptive attacks on FLOW with CIFAR-10. We use FedAvg under no attack, denoted by "No attack", as the baseline. Table VI shows the effectiveness of FLOW against the adaptive attacks. For all the three types of perturbation vectors, the accuracies of the global models under adaptive attacks are at most 1.7% lower than that of FedAvg under no attacks. It reveals that FLOW is still robust against the adaptive attacks. Moreover, even though the adaptive attacks may reach high cosine similarity between the benign updates and the malicious updates, such as 0.83 and 0.87 for ∇^p_{uv} and ∇^p_{sgn} respectively, the malicious updates merely bring puny damage to the global model. It is because FLOW can prevent the adaptive attacks in most of training iterations. For instance, the ASTs of the adaptive attacks are at most 66, which is only 13.2% of the total training iterations.

VI. CONCLUSION

In this paper, we proposed and evaluated a new method called FLOW as a punishment-based federated learning framework to achieve Byzantine-robustness against both untargeted and targeted model poisoning attacks. We exploited the historical behaviors of the clients to adjust the weights of the corresponding local model updates in aggregating the global model. This design can not only reduce the negative effect of local models generated by malicious clients on the global model, but also avoid the model loss caused by misjudging benign clients as malicious. The evaluation results on different datasets show that FLOW outperforms the state-of-the-art FL Frameworks in defending against typical model poisoning attacks from a large number of malicious clients. Moreover, FLOW also shows Byzantine-robustness in defending against the dedicated adaptive attacks.

REFERENCES

[1] A. Belenguer, J. Navaridas, and J. A. Pascual, "A review of federated learning in intrusion detection systems for iot," *arXiv preprint arXiv:2204.12443*, 2022.

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

[3] B. Ghimire and D. B. Rawat, "Recent advances on federated learning for cybersecurity and cybersecurity for federated learning for internet of things," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8229–8249, 2022.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Statistics (AISTATS)*, Fort Lauderdale, FL, USA, 2017, pp. 1273–1282.

[5] M. Duan, D. Liu, X. Ji, Y. Wu, L. Liang, X. Chen, Y. Tan, and A. Ren, "Flexible clustered federated learning for client-level data distribution shift," *IEEE Trans. Parallel. Distrib. Syst.*, vol. 33, no. 11, pp. 2661–2674, 2022.

[6] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, and L. Liang, "Self-balancing federated learning with global imbalanced data in mobile systems," *IEEE Trans. Parallel. Distrib. Syst.*, vol. 32, no. 1, pp. 59–71, 2021.

[7] Y. Zhang, D. Liu, M. Duan, L. Li, X. Chen, A. Ren, Y. Tan, and C. Wang, "Fedmds: An efficient model discrepancy-aware semi-asynchronous clustered federated learning framework," *IEEE Trans. Parallel. Distrib. Syst.*, vol. 34, no. 3, pp. 1007–1019, 2023.

[8] M. Duan, D. Liu, X. Chen, Y. Tan, J. Ren, L. Qiao, and L. Liang, "Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications," in *Proc. IEEE 37th Int. Conf. Computer Design (ICCD)*, Abu Dhabi, United Arab Emirates, 2019, pp. 246–254.

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

[10] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Adv. Neural Inf. Process.*, vol. 30, 2017.

[11] V. Shejwalkar and A. Houmansadr, "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, Virtual event, 2021.

[12] X. Chen, L. Xiao, W. Feng, N. Ge, and X. Wang, "Ddos defense for iot: A stackelberg game model-enabled collaborative framework," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9659–9674, 2022.

[13] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Stockholm, Sweden, 2018, pp. 5650–5659.

[14] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, New Orleans, LA, USA, 2019.

[15] X. Cao, M. Fang, J. Liu, and N. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, Virtual event, 2021.

[16] S. Li, Y. Cheng, W. Wang, Y. Liu, and T. Chen, "Learning to detect malicious clients for robust federated learning," *arXiv preprint arXiv:2002.00211*, 2020.

[17] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Min. (SIGKDD)*, Washington, DC, USA, 2022, pp. 2545–2555.

[18] 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, 1998.

[19] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[20] L. Li, D. Liu, M. Duan, Y. Zhang, A. Ren, X. Chen, Y. Tan, and C. Wang, "Federated learning with workload-aware client scheduling in heterogeneous systems," *Neural Netw.*, vol. 154, pp. 560–573, 2022.

[21] M. Duan, D. Liu, X. Ji, R. Liu, L. Liang, X. Chen, and Y. Tan, "Fedgroup: Efficient federated learning via decomposed similarity-based clustering," in *Proc. IEEE Int. Conf. Parallel Distrib. Process. Applications (ISPA)*, New York City, NY, USA, 2021, pp. 228–237.

[22] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained iot devices," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 1–24, 2022.

[23] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in iot," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5986–5994, 2020.

[24] T. Zhang, C. He, T. Ma, L. Gao, M. Ma, and S. Avestimehr, "Federated learning for internet of things," in *Proc. 19th ACM Conf. on Embedded Networked Sensor Systems (SenSys)*, New York, NY, USA, 2021, pp. 413–419.

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on January 25,2024 at 08:54:33 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3341811

[25] W. Y. B. Lim, Z. Xiong, J. Kang, D. Niyato, C. Leung, C. Miao, and X. Shen, "When information freshness meets service latency in federated learning: A task-aware incentive scheme for smart industries," *IEEE Trans Industr. Inform.*, vol. 18, no. 1, pp. 457–466, 2020.

[26] R. S. Antunes, C. André da Costa, A. Küderle, I. A. Yari, and B. Eskofier, "Federated learning for healthcare: Systematic review and architecture proposal," *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 4, may 2022.

[27] E. M. Campos, P. F. Saura, A. González-Vidal, J. L. Hernández-Ramos, J. B. Bernabé, G. Baldini, and A. Skarmeta, "Evaluating federated learning for intrusion detection in internet of things: Review and challenges," *Comput. Netw.*, vol. 203, p. 108661, 2022.

[28] J. Li, L. Lyu, X. Liu, X. Zhang, and X. Lyu, "Fleam: A federated learning empoweblue architecture to mitigate ddos in industrial iot," *IEEE Trans. Ind. Informat.*, vol. 18, no. 6, pp. 4059–4068, 2022.

[29] X. Chen, W. Feng, N. Ge, and Y. Zhang, "Zero trust architecture for 6g security," *IEEE Netw.*, 2023.

[30] Y. Li, X. Tao, X. Zhang, J. Liu, and J. Xu, "Privacy-preserved federated learning for autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 8423–8434, 2022.

[31] T. Liu, J. Xia, Z. Ling, X. Fu, S. Yu, and M. Chen, "Efficient federated learning for aiot applications using knowledge distillation," *IEEE Internet Things J.*, vol. 10, no. 8, pp. 7229–7243, 2023.

[32] W. Ni, J. Zheng, and H. Tian, "Semi-federated learning for collaborative intelligence in massive iot networks," *IEEE Internet Things J.*, vol. 10, no. 13, pp. 11 942–11 943, 2023.

[33] T. Zhang, L. Gao, C. He, M. Zhang, B. Krishnamachari, and A. S. Avestimehr, "Federated learning for the internet of things: Applications, challenges, and opportunities," *IEEE Internet Things Mag.*, vol. 5, no. 1, pp. 24–29, 2022.

[34] 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. (DISS)*, San Diego, CA, USA, 2020, pp. 1–7.

[35] J. Zhang, B. Chen, X. Cheng, H. T. T. Binh, and S. Yu, "Poisongan: Generative poisoning attacks against federated learning in edge computing systems," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3310–3322, 2020.

[36] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, no. 1–2, pp. 1–210, 2021.

[37] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Proc. 25th Eur. Symp. Research Comput. Security (ESORICS)*, Guildford, UK, 2020, pp. 480–501.

[38] J. Zhang, C. Ge, F. Hu, and B. Chen, "Robustfl: Robust federated learning against poisoning attacks in industrial iot systems," *IEEE Trans. Ind. Informat.*, vol. 18, no. 9, pp. 6388–6397, 2022.

[39] X. Ma, Q. Jiang, M. Shojafar, M. Alazab, S. Kumar, and S. Kumari, "Disbezant: Secure and robust federated learning against byzantine attack in iot-enabled mts," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 2, pp. 2492–2502, 2023.

[40] Z. Ma, J. Ma, Y. Miao, Y. Li, and R. H. Deng, "Shieldfl: Mitigating model poisoning attacks in privacy-preserving federated learning," *IEEE Trans. Inf. Forensics Secur.*, vol. 17, pp. 1639–1654, 2022.

[41] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," *Adv. Neural Inf. Process.*, vol. 32, 2019.

[42] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Long Beach, California, USA, 2019, pp. 634–643.

[43] C. Xie, O. Koyejo, and I. Gupta, "Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation," in *Uncertainty Artif. Intell.* PMLR, 2020, pp. 261–270.

[44] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *Proc. 29th USENIX Secur. Symp. (USENIX Security)*, Virtual event, 2020, pp. 1605–1622.

[45] X. Cao and N. Z. Gong, "Mpaf: Model poisoning attacks to federated learning based on fake clients," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, New Orleans, LA, USA, 2022, pp. 3396–3404.

[46] Y. Huang, L. Chu, Z. Zhou, L. Wang, J. Liu, J. Pei, and Y. Zhang, "Personalized cross-silo federated learning on non-iid data," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, vol. 35, no. 9, Virtual event, 2021, pp. 7865–7873.

[47] T. D. Nguyen, P. Rieger, R. De Viti, H. Chen, B. B. Brandenburg, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen *et al.*, "Flame: Taming backdoors in federated learning," in *Proc. 31st USENIX Secur. Symp. (USENIX Security*, Boston, MA, USA, 2022, pp. 1415–1432.

[48] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An efficient k-means clustering algorithm: analysis and implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 881–892, 2002.

[49] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *J. R. Stat. Soc., Ser. B, Methodol.*, vol. 63, no. 2, pp. 411–423, 2001.

[50] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognit.*, vol. 77, pp. 354–377, 2018.

[51] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770–778.

[52] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," in *Proc. IEEE 38th Int. Conf. Data Eng. (ICDE)*, Kuala Lumpur, Malaysia, 2022, pp. 965–978.

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on January 25,2024 at 08:54:33 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.