Neural Processing Letters (2025) 57:28
https://doi.org/10.1007/s11063-025-11724-2

FedDefense: A Defense Mechanism for Dishonest Client Attacks in Federated Learning

Gaofeng Yue[1] · **Xiaowei Han**[2]

Accepted: 6 January 2025

Abstract
Federated Learning (FL), which allows multiple participants to co-train machine learning models, enhances privacy-preserving by avoiding exposing local data. In recent years, FL has been considered a promising paradigm. However, during the FL process, individual clients may fall out on the client's side, or a particular client may engage in dishonest behavior such as uploading malicious data, thereby hindering the training of the global model. Most of the existing defense methods are considered only from the perspective of data filtering or model weighting, which have the disadvantages of poor robustness and high computational cost. Therefore, we propose a novel security FL (FedDefense) scheme based on client selection and adaptive rewards to defend against dishonest client attacks. First, to reduce the likelihood of poisoned clients participating in aggregation, we design a randomized subset method for client contribution evaluation via Kullback–Leibler (KL) divergence. Second, we reduce the server's dependence on clients through a dynamic reward strategy to ensure healthy model training. Numerical analysis and performance evaluation show that the proposed technique prevents the threat of dishonest clients during FL processing. Compared with existing methods, our approach has significant advantages in terms of efficiency and performance.

Keywords Federated learning · Dishonest client attacks · Client selection · Adaptive rewards

1 Introduction

In recent years, the topic of distributed machine learning has become popular among scholars and is widely applied in many fields [1]. The massive edge devices are producing vast amounts of data [2]. But, the thorny question is that private owners of sensitive data are reluctant to expose and share their data entirely, which forms numerous islands [3].

In the conventional Federated Learning (FL) [4] system, FL allows multiple participants to generate a global model in a privacy-preserving manner. The traditional FL scheme consists

✉ Gaofeng Yue
yuegaofeng1106@163.com

[1] School of Cyber Science and Engineering, Xi'an Jiaotong University, Taiyi Street, Xi'an 710049, Shaanxi, China

[2] Institute of Science and Technology Innovation, Shenyang University, Wanghua Street, Shenyang 110044, Liaoning, China

Published online: 07 March 2025

of three processes [5]: (1) the global server distributes the model parameters to local clients; (2) each local client trains its own model and uploads the value of the gradient change rather than raw data; (3) The trusted server will aggregate local owners' parameters to update global model. Although traditional FL is a tremendous modern shift in distributed machine learning, the trustworthiness of participants is not taken into account. It's possible that several local clients are offline for valid reasons or that local devices have compromised the tradeoff by providing unreliable or malicious models [6].

There is a widespread phenomenon that clients maliciously or unintentionally may degrade the training of the global model, thereby degrading the model's performance in the target task. This issue can have serious consequences across various fields, particularly in areas such as image classification and natural language processing, where the accuracy of models is crucial for decision-making processes.

For example, when processing image classification, a Malicious Client (MC) or Dishonest Client (DC) can re-label cars as birds, thus causing the learned model to think that a car is a bird [7]. This deliberate mislabeling can significantly distort the results of the model and lead to inaccurate predictions. Recognizing this threat, college researchers or corporate engineers have dedicated efforts to designing and proposing various security protection strategies for FL [8–11].

One such strategy is a blockchain-FL-based intrusion detection scheme [8, 11], which aims to detect and prevent malicious activities within the FL network. Additionally, a conditional variational automatic encoder known as Fedcvae [9] has been developed to detect malicious model updates and prevent them from affecting the global model. Moreover, an ensemble FL immune to MCs has also been introduced as a means of safeguarding the integrity of the training process [10].

The central idea of these approaches is to mitigate the impact of statistical anomalies in local model updates, thus ensuring the overall reliability of the global model. However, a common limitation among these strategies is the lack of emphasis on strict privacy protection and the complexity in defining thresholds for malicious behavior.

Moving forward, it will be crucial to systematically address the challenge of preventing poisoned clients from attacking the global model. This task presents an interesting and complex proposition that requires innovative solutions to effectively safeguard the integrity and performance of FL systems. By developing comprehensive security measures and prioritizing privacy protection, we can enhance the resilience of FL networks against malicious attacks and ensure the accuracy of machine learning models in diverse applications.

Our FedDefense work introduces a novel defense mechanism that is unique compared to other federated learning literature using Kullback-Leibler scatter. While existing literature [12–14] has explored various approaches to model fusion and optimization using Kullback–Leibler scatter, our work stands out by introducing a novel defense mechanism against dishonest clients in a federated learning environment. Our integration of the FLUK defense mechanism [14] into the aggregation process adds a layer of protection against malicious clients in federated learning settings, a key aspect that has not been fully explored in the existing literature. By incorporating FLUK into our FedDefense framework, we enhance the federated learning model's resistance to dishonest attacks, thereby improving the overall security and reliability of FL systems. In terms of comparing to other work in FL, FedDefense is more effective in guarding against security threats in FL systems by reducing the chances of malicious client participation and introducing an adaptive defense reward mechanism. By dynamically updating the reward list and the client subset selection step, FedDefense enables more efficient model aggregation and protection, which improves resilience against

malicious attacks. These features differentiate FedDefense from other approaches in protecting FL systems from malicious activities.

In summary, the major contributions of this article are:

1. Firstly, a method for resisting threats caused by attacks initiated by dishonest members in FL is proposed by reducing participation probability and client dependency. We introduce a novel secure FL (FedDefense) scheme with adaptive defense rewards to achieve high security within limited computational budget.
2. Secondly, a random subset client selection method is designed to reduce the likelihood of poisoned clients participating in aggregation. By dynamically updating reward lists, calculating client contributions, and evaluating the aggregation capability of client subsets, combined with reward updates and client selection steps, model aggregation and protection in FL are achieved.
3. Finally, comprehensive experiments were conducted to test the effectiveness of our defense method. We evaluated its performance on various datasets and compared the results with the accuracy and defense capabilities of existing methods. Experimental results demonstrate that our method outperforms state-of-the-art methods in terms of performance.

The structure of the rest of the paper is outlined as follows. In Sect. 2, related works are discussed. In Sect. 4, our observations regarding FL are outlined. Section 3 presents some preliminaries utilized by FedDefense. Section 6 presents the algorithm design of FedDefense that consists of client subset selection and dynamic incentive. Section 7 presents performance evaluation results. Section 8 summarizes the paper and views future work.

2 Related Work

This section presents an overview of previous and noteworthy studies regarding client attacks in the field of FL.

In FL, there are several types of Dishonest Client Attack(DCA) that can occur, including physical offline attacks, attacks from poisoned clients, and attacks from the model centralizer, among others. Hei et al. [8] proposed a Blockchain-FL-based Cloud Intrusion Detection System (BFL-CIDS) to address malicious attacks in a distributed environment. The solution introduces the use of a cloud computer center, which stores the training process parameters and behavior information using blockchain technology. This approach reduces the likelihood of false alarms and improves the accuracy of FL. Considering defending against targeted model poisoning attacks, the literature [9] (Gu et al., 2021) proposes a strategy called Fedcvae. In this approach, the central server employs a conditional Variational AutoEncoder (VAE) to detect and eliminate malicious models in an unsupervised FL system. The Fedcvae surmounts a few weaknesses of the VAE model but only adopts specific malicious attacks, e.g., the same-value attack. During the same year, Cao et al. [10] developed a secure FL approach to defend against MCs by utilizing a randomly selected subset of clients. This approach is robust against a limited number of NCs in certain scenarios. The authors also introduce a Monte Carlo algorithm to calculate certified security levels and achieve improved performance in defending against MCs [15]. In practical applications, Abubaker et al. (published in 2022) [11] propose a solution for detecting and removing malicious nodes in the Internet of Things (IoT) using blockchain technology based on Beyond fifth-Generation (B5G). This solution is important in building trust among all entities and the innovation of

the paper is the implementation of a combined digital signature with cascading encryption to ensure non-repudiation of both the global server and local client.

3 Preliminaries

In Sect. 3, we introduce the concept of FL [4], which is a scenario where many clients collaboratively build high-quality machine learning models, which has drawn attention recently.

Considering traditional FL [5, 16], it consists of a centralized server (i.e., global model) and n clients (i.e., local models), where the math set of clients is $N = \{1, 2, \cdots, i, \cdots, n\}$. Simultaneously, we could suppose that client-i has a local dataset $D_i = \{x_i, y_i\}$, where x_i is data input and y_i is truth label of x_i. Thus, the total samples of participating clients can be defined as $D_n = \sum_{i=1}^{n} D_i$ in FL, where the sample size is enormous. For client-i side, the local loss function $F_i(\omega_i)$ is formulated as follows:

$$F_i(\omega_i) = \frac{1}{b} \sum_{i=1}^{n} F_b(\omega_i; \xi_i), \tag{1}$$

where b represents the length of the local dataset over mini-batch ξ_i, $F_b(\omega_i; \xi_i)$ is the empirical loss function computed over mini-batch ξ_i under local model ω_i of the i-th client. The FL aims to minimize the averaged sum of loss functions among the distributed and scattered data samples and explore a set of model parameters. Thus, model training can be formally described as optimizing the following objective function [17], as Eq. (2):

$$\arg\min_{\omega_g} F_g(\omega_g) = \frac{1}{n} \sum_{i=1}^{n} F_i(\omega_i; D_i), \tag{2}$$

where ω_g represents the global model parameter of the FL networks, $F_i(\omega_i; D_i)$ represents the loss function at i-th client. The iterative optimization is used for the FL training process, which is a general approach. The ultimate result is that we get the desired global model parameters for the target task.

Figure 1 shows the fundamental procedure and framework of FL, which is more convenient for us to understand. After the central server broadcasts its stored global model to the clients (i.e., step-1 in Fig. 1), what cannot be ignored is the process of updating the local model by the clients (i.e., step-2 in Fig. 1). The model will be updated by applying the Stochastic Gradient Descent (SGD) algorithm [5, 18], which provides an effective way to optimize the loss function. For the mini-batch SGD, a gradient descent step over a mini-batch on each client is regarded as a local iteration (or a local updating). After performing one or multiple local iterations, each client uploads its own local models or gradients (i.e., step-3 in Fig. 1), and further the server aggregates these models (i.e., step-4 in Fig. 1). Such a training process is regarded as a communication round.

In detail, defining iteration set $T = \{0, 1, \cdots, t, \cdots, \tau\}$, the local client performs a partial derivation of its loss function $F_i(\omega_i)$ to get a targeted local gradient $g_i(t)$ at t round, as Eq. (3).

$$\omega_i(t) = \omega_i(t-1) - \eta g_i(t)), \tag{3}$$

where η is the local learning rate. In this way, we have accomplished the more fundamental step in FL through Eqs. (1) and (3) together.

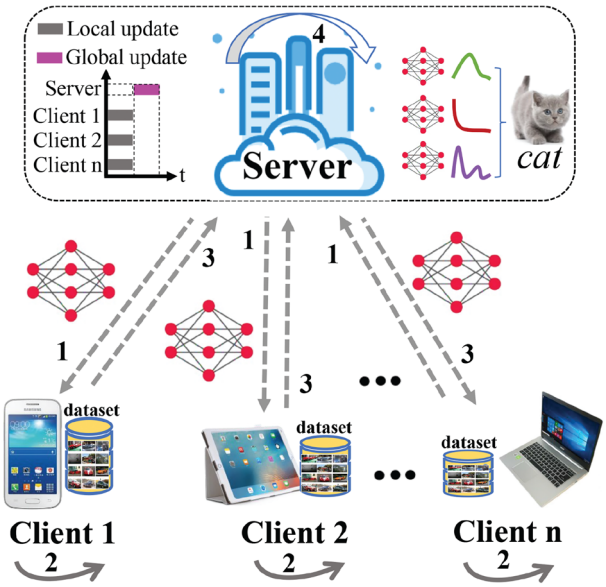


Fig. 1 Schematic representation of the FL process

Of course, for server side, there are various ways of aggregation and the simplest method is called FedAvg [16] as Eq. (4):

$$\omega_g(t) = \frac{1}{n}\sum_{i=1}^{n}\omega_i(t). \tag{4}$$

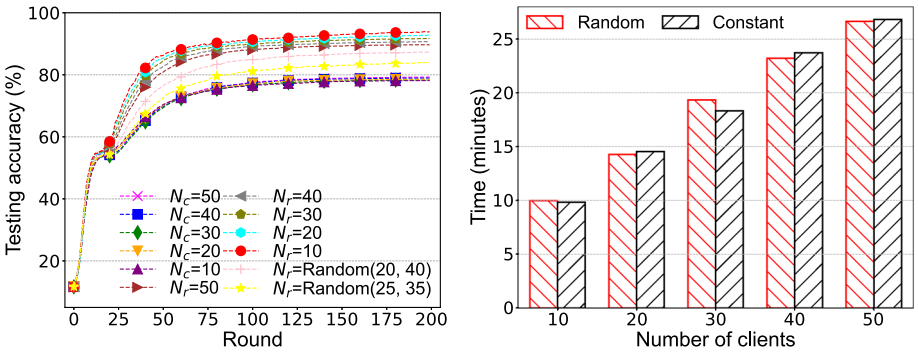
Finally, the stopping conditions for this optimization process are that the global model convergences to a sure accuracy or the iteration number exceeds the upper limit τ.

4 Observation and Motivation

In Sect. 4, we introduce some observational experiments on the thought process of innovation points. These small tests provide guidance for designing better algorithms.

4.1 Environmental Settings

The local model uses a Convolutional Neural Network (CNN) consisting of an input layer, two convolutional layers, three fully connected layers, one pool layer, and an output layer, which is more representative in image classification tasks [18, 19]. In addition, the learning rate η is 0.01. The mini-batch size is 128. The number of clients is 100 in subsequent experiments. Unless otherwise noted, the experimental setup described above was kept constant in subsequent experiments. The dataset used in this Observation is MNIST dataset [15], which contains 70,000 examples divided into 10 classes. In each class, there are 7000 examples, with 6000 used for training and 1000 for testing. Each example in the dataset is a binary image of size 28×28, and is labeled according to its co*R*esponding class.



(a) Impact on training accuracy (b) Impact on training time

Fig. 2 The impact of the number of clients and being selected way on the global model

4.2 Client Selection for FL

In each iteration of FL, all local users stay online and upload their parameters to a server for aggregation. In actual scenarios, FL clients exhibit significant heterogeneity in terms of data statistics and system configurations, which can degrade FL performance if they all participate in global aggregation [20]. Thus, FL client selection (also known as participant selection or device sampling) is an emerging topic. An effective FL client selection scheme can significantly improve model accuracy, strengthen robustness [21], and reduce training overheads [22]. And cu*R*ently there are essentially two types of client selection, a constant certain selection and a random selection in all clients [20]. Therefore we tested what kind of selection method has improved FL performance. First, variables N_c and N_r are denoted as the number of clients based on two client selection methods, i.e., constant and random, respectively. Then we test the FL accuracy under multiple variants of N_c and N_r in MNIST dataset, and the results are shown in Fig. 2a. This small experiment shows that the client selection methods in different quantities and by different methods play different roles in the global model. It can be seen that the random number of clients is significantly high accurate than the constant value in Fig. 2a. The fewer randomly selected clients, the higher the accuracy of the model. A random value that varies in the interval time also does not perform well. This finding suggests that randomly selecting a diverse group of clients can help to improve the generalization ability of the global model. After this discovery, we urgently needed to understand whether these two client selection methods had any impact on training time, as FL training efficiency is something we also focus on more than anything else. We specifically compared the training time for different client numbers to achieve the same rounds by random and constant methods, and the results are shown in Fig. 2b, It is almost no difference in their training time under random and constant ways. This suggests that we can confidently go about randomly selecting clients and designing the number of clients which is selected.

The two small tests illustrate that it is not optimal for all clients to participate in FL training, but rather to choose a randomly selected group of local participants. It is important to evaluate the impact of different numbers of randomly selected clients on the FL performance and choose the optimal number for each scenario. This discovery provides novel ideas for dealing with DCA and improving model accuracy.

5 Threat Model and Problem Statement

We focus on a cross-device FL system, of which clients and the server participating in FL are honest and curious. The server will follow the protocol for FL, but is curious about individual model updates, while the clients will use the cryptographic to protect their model updates. Since the goal of our proposed method is to minimize the performance loss caused by dishonest clients to the global model, we assume that clients will not share their private data with others. This assumption is commonly adopted in systems with cryptographic protection [23–26]. The protection against malicious activities, such as cloud-client collusion, poisoning attacks, or backdoor attacks, is not the focus of this work, and we refer to existing methods for extra defense mechanisms [7, 8]. Based on the design goals and security assumptions, we formulate the research question as follows: how to keep the global model healthy for training with a sufficiently secure defense mechanism, considering the cu*R*ent means available through client selection?

6 System Design

Benefiting from the observations in Sect. 4, we designed a high-secure FL (i.e., FedDefense) to defense DCA in FL training.

6.1 Overall Workflow

DCA pose a complex and challenging problem that cannot be completely solved directly. Such attacks may be launched actively by participating nodes or passively manipulated by malicious actors. To mitigate the risk of attacks, we propose an optimized solution (i.e., FedDefense) based on randomization selection and adaptive reward in this article. The FedDefense algorithm randomly generates several client subsets and chooses one subset to reduce the probability of being attacked by dishonest clients. Then, considering the contribution of each client's model accuracy in the selected subset, we actively update the list generated by the incentive function. The FedDefense algorithm selects more secure clients by analyzing the impact of each client's model on the global model during each iteration, and uses incentives to increase the aggregation weight of high-quality and secure clients. In summary, this approach reduces the likelihood of attacks by actively defending against them and selecting high-quality clients.

Figure 3 shows the overall workflow of our proposed approach (i.e., FedDefense): The main goal of the proposed scheme is to allow multiple clients to jointly build an effective image classification network model in a high-secure way. The complete workflow can be divided into the following seven phases.

- ***System initialization:*** The initial step involves setting up a secure communication channel among all clients through the global server. Each client registers at the global server for participation in FL. The global server creates a reward list R to store the weights of all clients. Clients agree on the initial parameters ω_g of the global model, along with other key parameters such as the learning rate η, batch size bb, total communication rounds τ, and the loss function $F_g(\omega_g)$ as the common training objective. The initial global model ω_g is then distributed to each client.
- ***Local model training:*** Each client conducts model training with its local data D_i using Stochastic Gradient Descent (SGD) to obtain a local model update ω_i. The local model

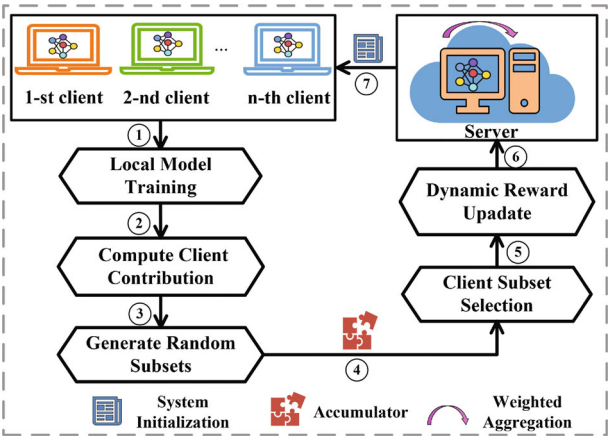


Fig. 3 System overview of the FedDefense

update information comprises the local model ω_i, local model update gradients $g_i(\omega_i)$, and a timestamp t.

- ***Compute client contribution:*** This stage involves calculating the Kullback–Leibler (KL) divergence ℓ_i for each client to evaluate the influence of their local model on the global model. By determining the client's contribution c_i, we assess the dissimilarity between each client's data distribution and the global model, aiding in identifying the clients whose data is most relevant for global model updates.
- ***Generate random subsets:*** Here, client subsets are created by randomly selecting mm clients from the overall client pool, with this process repeated d times to obtain d client subsets of size m. Ensuring equal subset sizes helps maintain data balance during model aggregation, preventing any single client from disproportionately influencing the update.
- ***Client subset selection:*** After obtaining d client subsets, the accumulator computes the total contribution of each subset. The subset with the highest contribution is selected, and its co*R*esponding client indices form the group awaiting aggregation. This method efficiently identifies the clients crucial for model updates, enhancing the selection process for model aggregation participants and improving efficiency and accuracy.
- ***Dynamic reward update:*** This component involves dynamically adjusting the reward list based on client participation. By incentivizing active involvement from healthy clients, the server boosts model performance. Continual adjustment and optimization of this process are essential for ensuring effective and stable model updates.
- ***Server aggregation:*** In this phase, the server aggregates clients based on their reward values for weighted aggregation, using the rewards stored in the list R. The parameter server executes the aggregation process to generate a new global model update. If the global model's testing e*R* or on the validation dataset meets predetermined criteria or the FL training rounds reach the set limit τ, the server ceases distribution of parameters like ω_g for further training rounds.

6.2 Solution for DCA: FedDefense

Some existing research suggests that the biggest threat to FL comes from attacks launched by dishonest members, mainly due to the uncontrollability and dependency of the server-side

on the client [27]. To combat DCA threats, an excellent solution is to reduce the probability of participation and the dependency of the client side, ensuring that the model is not attacked. Therefore, based on observation 4.2, we have designed a client selection scheme that integrates FedDefense to defense intrusions and achieve high security while staying within a limited computation budget.

The proposed work introduces a novel approach to managing client contributions in federated learning (FL) systems. Unlike traditional FL defense mechanisms or reward systems, the proposed scheme uses Kullback-Leibler (KL) divergence to measure the contribution of each client in a more nuanced and accurate way. This distinguishes the proposed method, as it leverages a sophisticated mathematical tool to assess the quality of client updates and incentivize cooperation in the FL process.

The fundamental idea of the FedDefense scheme we have designed is to select m clients from n clients to aggregate the model and minimize the risk of DCA. For this selected subset, we can minimize the subset risk of DCA and formulate this objective as follows:

$$\underset{M_{sub}}{argmax}\ \Theta, \quad s.t.\ m = |\{M_{sub}\}| \leq n, \tag{5}$$

where m is the number of a random subset M_{sub} in n clients and Θ is the most secure level under computational budget.

First, we need to build a list $R = [R_1, R_2, \cdots, R_n]$ to store the incentive values of all clients in the training of FL, and all elements will be initialized with zero by the server when $t = 0$. In this component, we need to dynamically update the reward list. Adjusting the rewards based on client participation, the server incentivizes more healthy clients to actively participate in model updates to improve model performance.

Second, to assess the impact of the local model on the global model, we choose the KL divergence method for contribution calculation. This is exemplified by comparing $R^t_{l_1} = f(D_0, w^t_{l_1})$ and $R^t_{l_2} = f(D_0, w^t_{l_2})$, representing the knowledge probability distributions of distinct clients l_1 and l_2. In the context of discrete variables, the KL divergence is defined as:

$$KL(R^t_{l_1} \,\|\, R^t_{l_2}) = \sum R^t_{l_1} \log\left(\frac{R^t_{l_1}}{R^t_{l_2}}\right). \tag{6}$$

KL divergence is a measure of the difference between two probability distributions and is often used to measure information loss or information gain in information theory. And information entropy is a concept used to measure the uncertainty or the amount of information in information theory. By calculating the information entropy, we can understand the degree of uncertainty about a random variable. Shapley value is a method used in game theory to determine how much each player involved in a game contributes to the overall outcome. Although the Shapley value is widely used in game theory, the KL divergence method may be more appropriate in terms of contribution calculation:

- Applicability of Definition: The KL divergence is designed to measure the similarity or difference between two probability distributions. In contribution calculations, we often need to compare behavioral differences between customers, and the KL divergence can provide an objective metric to more accurately assess each customer's impact on the overall contribution.
- Mathematical properties: KL divergence has some favorable mathematical properties, such as non-negativity and symmetry, which make it more convenient and stable in practical calculations. These properties can help us better handle large-scale data and extract the really important contribution information from it.

- Informative nuance: KL divergence measures the difference in information per data point between two probability distributions and, therefore, can measure each client's contribution in a more nuanced and accurate way. In contrast, the Shapley value may be more inclined to assess each player's contribution holistically, ignoring the impact of each player's behavior on the overall outcome. In summary, the KL divergence method for contribution calculation may have been chosen because it is better suited for comparing behavioral differences between clients and can measure each client's contribution in a more precise and nuanced manner.

Thus, to abstract the most valuable information that makes the most sense to improve the accuracy of the model, so we will denote $c(t)$ to characterize the contribution degree in the accuracy impact of each candidate client of FL at t-round. We constructed the sigmod function to describe participants' contributions to the model. The advantage of the sigmod function is that it is a threshold function, and the output does not overflow as the iterations change. To prevent redundant calculations, we choose to calculate all clients at once, as in Eq. (7)

$$\begin{cases} p_i(t) = \exp(\ell_i(t))/\sum_1^n \exp(\ell_i(t)), \\ c_i(t) = 1/(1+\exp(-\vartheta \cdot p_i(t))), \end{cases} \tag{7}$$

where ϑ is a constant, $\ell_i(t)$ represents Kullback-Leibler (KL) divergence [28] between local model and global model at t-round. The $p_i(t)$ is denoted as the divergence percentage of the i-th client across all clients.

The novelty of using KL divergence lies in its ability to quantify the difference between two probability distributions, which allows for a more fine-grained evaluation of client updates. By utilizing this metric, the proposed scheme is able to better differentiate between clients who provide valuable contributions and those whose updates may be less reliable or detrimental to the overall FL model. This represents a significant advancement in FL management techniques, as it offers a more granular and precise way to assess client performance.

Next, we will randomly pick up m clients from all participants to construct a client subset M_{sub}. In the above manner, we need to iterate d times. As a result, we get d client subsets whose size is m. In the process of determining client subsets, we need to ensure that each subset is of equal size, which helps maintain data balance during model aggregation and prevents some clients from having too much impact on model updates. By randomly selecting client subsets multiple times and repeating this process, we can obtain multiple different client subsets to more comprehensively assess each client's contribution to the model.

Once the contribution capacity $c(t)$ of each client on accuracy is obtained, we can sum their contribution degree of the client subset M_{sub}. We can define $\sum c_i(t)$ as the convergence ability of the client subset. The result is that the server side will get a convergence degree sequence $C_d(t)$ of all client subsets, as Eq. (8).

$$C_d(t) = \left\{ \sum c_1(t), \sum c_2(t), \cdots, \sum c_d(t) \right\}. \tag{8}$$

Following the maximization strategy, the client's subset M_{sub} co$R$esponding to the largest in $C_d(t)$ will be chosen to aggregate the model on the server-side. Thus, we choose the subset with the largest contribution to constitute the set of local customers waiting to be aggregated.

Next, we need to update the rewards list by contribution level and client selection. At each round t, the update of the R_i is the most important part in client selection and will be calculated using Eq. (9) to get weights of clients.

$$R_i(t) = R_i(t-1) + \theta^{c_i(t-1)-1} + B, t \in T, \tag{9}$$

Algorithm 1: FedDefense: A defense mechanism for dishonest client attacks in Federated Learning.

Input: Image classification task dataset $\{D_n\}$; Initial global model $\omega_g(0)$; Number of global training rounds τ; Number of local training rounds K; Initial the learning rate η; Initial client reward list $R(0)$; The number of client subsets is d; The mini-batch size b;
Output: The trained global model $\omega_g(t)$;

1 **for** $t = 1, 2, \ldots, \tau$ **do**
2 Server sends global model $\omega_g(t-1)$ to all the clients;
3 **for** each client $i \in N$ *in parallel* **do**
4 client i initializes local model $\omega_i(t-1) \leftarrow \omega_g(t)$;
5 **for** $k = 1, 2, \ldots, K$ **do**
6 $\omega_i(t) = \text{LocalUpdate}(\omega_i(t-1), D_i, b, \eta)$ by Eqs. (1) and (3);
7 client i upload local model $\omega_i(t)$ to server;
8 Server calculates KL divergence and client contribution of each client by Eq. (7);
9 Server randomly generate client subsets by Eq. (8) repeating d times;
10 Server performs subset selection based on $C_d(t)$;
11 Server performs dynamic reward update R by Eq. (9);
12 Server aggregates $\omega_i(t)$ for selected client and performs averaging operation by Eq. (10);

where θ is a constant and B is defined as the client's selection and abandonment, represented by "1" and "0" respectively. The motivation of parameter $c_i(t)$ is to reward the selective clients and filter out non-credible or dishonest clients. Meanwhile, setting B ensures the selected number of clients is not higher than the budget.

The final step in the FedDefense approach is that the server performs parameter aggregation as Eq. (10).

$$\omega_g(t) = \sum_{i=1}^{m} R_i(t) \cdot \omega_i(t). \tag{10}$$

The whole FedDefense algorithm will be performed with iterations of traditional FL. The FedDefense is based on random selection and active defense. After random selection and grouping, even if malicious or dishonest clients are selected, their incentive factors will be minimal over several iterations, which means that these clients will have little effect on the model. Combining learning rewards with random fading effects on the client gives FL high-level security.

To better understand the process of the proposed FedDefense, we give the algorithm pseudocode in Table 1.

At Line 2, the server sends the global model $\omega_g(t-1)$ initialized at the start of the cu*R*ent training round to all the clients. At Lines 3-7, each client initializes its local model with $\omega_g(t-1)$, updates its model $\omega_g(t)$for K consecutive local rounds by Eqs. (1) and (3). At Line 8, Server calculates KL divergence and client contribution of each client by Eq. (7). At Line 9, Server generate multiple different client subsets by Eq. (8). At Lines 10-11, the server sorts the calculated cumulative contributions and then selects the subset of clients with the highest cumulative contributions. At the same time, the server needs to dynamically update the reward list R and adjust the incentives according to the participation of the clients by Eq. (9). At Line 12, the parameter server aggregates the uploaded $\omega_i(t)$ from selected clients by Eq. (10) and performs a weight averaging operation via reward list R. Finally, the FL training system obtains the final desired global model.

Table 1 Convergence accuracy with different parameters

Dataset	MNIST			CIFAR-10			Fashion-MNIST		
Round	τ=50	τ=100	τ=150	τ=50	τ=100	τ=150	τ=50	τ=100	τ=150
(1) d=1, m=10	80.7%	88.1%	89.3%	24.6%	29.3%	30.5%	52.1%	66.3%	67.8%
(2) d=2, m=10	80.2%	87.7%	89.1%	23.3%	28.2%	29.6%	51.4%	65.8%	66.9%
(3) d=3, m=10	79.6%	86.3%	88.5%	23.0%	27.8%	29.1%	50.8%	63.7%	64.2%
(4) d=1, m=20	80.6%	88.0%	88.4%	24.2%	30.1%	31.0%	53.2%	66.5%	67.8%
(5) d=2, m=20	80.2%	88.1%	88.2%	23.2%	29.5%	31.1%	52.8%	66.1%	66.9%
(6) d=3, m=20	**81.1%**	**88.3%**	**89.5%**	**24.7%**	**30.2%**	**31.0%**	**55.4%**	**67.1%**	**68.0%**
(7) d=1, m=30	80.1%	87.0%	87.2%	23.1%	30.3%	31.1%	52.3%	66.6%	67.1%
(8) d=2, m=30	80.5%	87.2%	88.3%	23.4%	29.1%	29.9%	54.3%	67.2%	67.3%
(9) d=3, m=30	79.2%	85.3%	86.9%	24.6%	28.6%	29.1%	54.1%	66.8%	67.5%
(10) d=4, m=30	76.5%	83.3%	84.1%	23.4%	27.8%	28.9%	53.3%	65.2%	66.0%

Bold values represent that our approach compares very favorably with other approaches

6.3 Theoretical Guarantee

Suppose there are n clients, including a malicious client. The success rate of this malicious client attack is 100% under traditional FL. If only ms clients are selected in a training round, the probability of success P_s of the attack is m/n. We analyze our algorithm again. The proof of the mathematical principles underlying the proposed FedDefense scheme can be outlined as follows: To incentivize client participation and filter out non-credible clients, a reward list $R = [R_1, R_2, \cdots, R_n]$ is initialized with zeros and dynamically updated based on client contributions. The rewards are adjusted using Eq. (9). Thus, the probability of success of the attack is $m/n \cdot m/\tau n$. The contribution degree of each client in improving model accuracy is quantified using the sigmoid function in Eq. (7). The sigmoid function ensures a threshold-based output that reflects the client's contribution accurately. Multiple subsets of mm clients are randomly selected and assessed for their convergence ability, calculated as the sum of individual contribution degrees. The server chooses the subset with the highest total contribution for model aggregation. Thus, the probability of success P_s of the attack is $m/n \cdot m/\tau n \cdot 1/d$ Assuming, $n = 50, m = 20, d = 5, \tau = 20$, the success rate of our method is 0.16, i.e., $P_s = 0.16$. Compared to traditional FL, our method is theoretically well defended.

7 Performance Evaluation

This section performs some experiments on the evaluation of the performance of the algorithm. Subsection 7.2 describes the environment and parameters of the relevant tests, and subsection 7.3 implements some results on the FedDefense and gives a detailed explanation and discussion of them.

7.1 Comparison Methods

As a result, we choose the traditional FL method (i.e., FedAvg [4]) as a reference line. Of course, this case implies that DCA does not exist for the complete FL system. Meanwhile, we compare our proposed framework with four previous representative approaches, *Krum* [29], *Trimmed mean* [30], *Bulyan* [31], *DPRLDS* [32] and *HeteroFL* [14].

Specifically, *FedAvg* [4] utilizes the aggregated model parameters of clients, which are weight-averaged by the clients' data quantity, to update the model of each client. *Krum* [29] propose an aggregation rule that satisfies our resilience property, arguing that it is the first provable Byzantine elasticity algorithm for distributed SGD. *Trimmed mean* [30] develop distributed optimization algorithms that are provably robust against Byzantine failures-arbitrary and potentially adversarial behavior, in distributed computing systems, with a focus on achieving optimal statistical performance. A main result of this work is a sharp analysis of two robust distributed gradient descent algorithms based on median and trimmed mean operations, respectively. *Bulyan* [31] provided raised concerns about the vulnerability of SGD and proposed a FL defense solution that enhances the resilience of SGD. *DPRLDS* [32] analyze this adversarial learning process in the FL setting and develop a new reinforcement learning-based defense strategy against model poisoning attacks. It also minimizes the attack vectors and facilitates attack disclosure *FLUK* [33] is (protecting FL Utilizing Kullback–Leibler divergence), a detection framework against poisoning attacks in FL-IoV setting by detecting malicious clients. Our key insight is that existing attacks produce malicious local updates that

derive from those of benign ones, resulting in a different distribution among these updates. The difference can be reflected by the Kullback–Leibler divergence between client updates in a single round and also between rounds.

7.2 Experimental Setting

FL Datasets. The experiments are driven by the datasets introduced in Sect. 4. The experimental datasets in this paper are MNIST [15], CIFAR-10 [19], and Fashion-MNIST [15], which are three classic datasets built for image classification. The three datasets used in these studies are:

- CIFAR-10 dataset [19], comprises 60,000 examples spread across 10 classes. Within each class, there are 6,000 examples, with 5,000 used for training and 1,000 for testing. Each example in the dataset is a color image of dimension 32×32 and is associated with a label reflecting its specific class.
- MNIST dataset [15], contains 70,000 examples divided into 10 classes. In each class, there are 7,000 examples, with 6,000 used for training and 1,000 for testing. Each example in the dataset is a binary image of size 28×28, and is labeled according to its co*R*esponding class.
- Fashion-MNIST [15] contains 70,000 examples divided into 10 classes. In each class, there are 7,000 examples, with 6,000 used for training and 1,000 for testing. Each example in the dataset is a binary image of size 28×28, and is labeled according to its co*R*esponding class. Following the common practice, we randomly take 50,000 samples as the training dataset and 10,000 as the test dataset for all 10 classes.

Models: The local model employs a CNN consisting of an input layer, two convolutional layers, three fully connected layers, one pool layer, and an output layer, which is more representative in image classification tasks [18, 19]. The learning rate is 0.01. The mini-batch size is 128. The number of clients is 100. In our experiments, without losing generality, the local epochs K is 5 [26] in our experiment.

Implementation: We conduct evaluations on a Linux server with 2 GeForce RTX 3090 GPUs, 1 Intel Xeon CPU, and 256 GB memory. We implement our framework with PyTorch [34], and all experiments were implemented in Python [34].

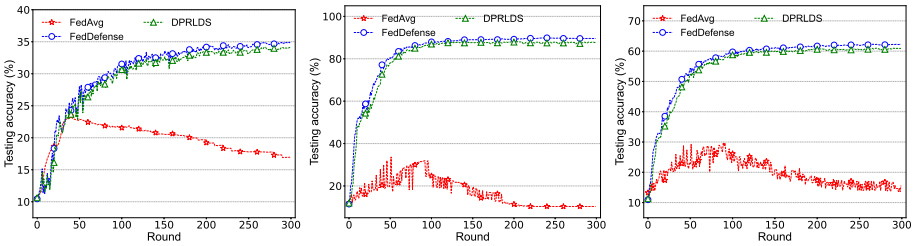
Evaluation Metrics: The metrics we measured are accuracy of image classification. Accuracy is the most commonly used metric and refers to the number of co*R*ectly classified samples as a percentage of the total number of samples, with the following formula:

$$Accuracy(\%) = \frac{TP + TN}{TP + FN + FP + TN}, \tag{11}$$

where TP (True Positive), Number of positive samples and classified as positive; FN (Flase Negative):The number of samples labeled positive and classified as negative, FP (Flase Positive):Number of samples whose label is negative and classified as positive, TN (True Negative): Number of samples labeled negative and classified as negative.

7.3 Experimental Evaluation

(1) *Convergence of FedDefense* In a real-world application of FL, clients may generate different dishonest data for various reasons. Therefore, DCA in this paper contains any local type of attacks without making any assumptions. For example, DCA contains physical offline



(a) Dataset MNIST (b) Dataset CIFAR-10 (c) Dataset Fashion-MNIST

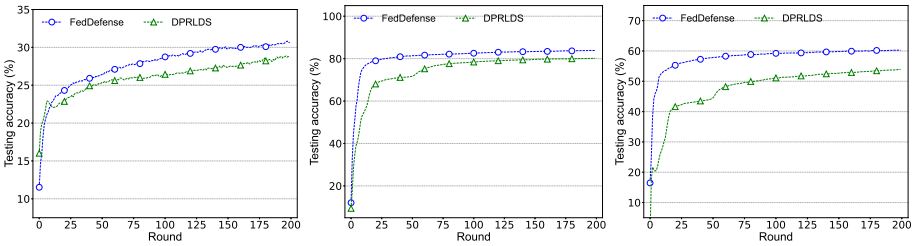
Fig. 4 The anti-attack capability of FedDefense

attacks, data manipulation attacks, and data cleansing attacks. In this experiment, we will test the convergence ability of DPRLDS [32] and FedDefense under continuous DCA with FedAvg as the reference line. The anti-attack capability of FedDefense and FedDefense is tested with a total of 100 clients. The number of malicious or dishonest clients is 5 (10% of total). Parameter dd is 3 and mm is 20, respectively. The entire attack is present from the first round and the malicious client stays five. The test results on CIFER-10, MNIST and Fashion-MNIST datasets are shown in Fig. 4.

From the experimental results, we can easily see that the traditional FL has lost a lot of accuracy in the case of one-time DCA. When there are successive DCAs during the training process, the unprotected FedAvd is co*R*upted, showing non-convergent results. In contrast, both the FedDefense method and the FedDefense method are greatly superior to FedAvg in terms of attack resistance in the case of successive attacks. Meanwhile, from the image curves of FedDefense, the FedDefense method has an extremely strong accuracy preservation ability, which is mainly due to the fact that the FedDefense method is equipped with a reward tool. Overall, the FedDefense method is convergent under various tests, which fully demonstrates that the design of the method in this paper is co*R*ect. Our method does not lose much model accuracy and has perfect anti-DCA capability.

(2) *Evaluating FedDefense's Performance in Heterogeneous Environments* The complexity in defining thresholds for malicious behavior, as highlighted by ours in the introduction, remains a significant challenge in the field of cybersecurity. While KL divergence in FedDefense may not directly resolve this issue, it can play a crucial role in addressing it by providing a quantitative measure of the differences between normal and malicious behavior patterns across federated datasets. The suggestion to evaluate FedDefense's performance in highly heterogeneous settings is valuable as it can provide insights into its ability to distinguish outliers from malicious clients in federated learning environments. To address this suggestion, the we could consider conducting experiments with datasets that exhibit varying degrees of heterogeneity in terms of data distributions, client populations, and communication patterns. The test results on CIFER-10, MNIST and Fashion-MNIST of Non-IID datasets are shown in Fig. 5.

It is obvious that both our method and the state-of-the-art method have degraded performance on the Non-IID dataset, but our method still outperforms DPRLDS. Incorporating this evaluation would indeed strengthen the work by showcasing the versatility and adaptability of FedDefense in diverse federated learning settings. Additionally, it could provide a more comprehensive understanding of the method's performance under different conditions, thereby enhancing the practical relevance of the proposed approach.



(a) Dataset MNIST (b) Dataset CIFAR-10 (c) Dataset Fashion-MNIST

Fig. 5 FedDefense's performance in heterogeneous environments

(3) *Ablation Study of Parameters* Since the novel algorithm proposed in this paper involves in two significant parameters, namely, Number of subsets d, and size of subsets m clients, we need to explore the mode in which these two parameters are mixed, and analyze their influence on the accuracy of the algorithm. Here we take the final convergence accuracy as the evaluation index and set the number of the client is 10 which would launch various attacks. The testing is conducting in three public datasets, i.e., MNIST, CIFAR-10, Fashion-MNIST. The detailed experimental results are shown in the following Table 1.

Based on the convergence accuracy data given in the table for different parameter settings, some of the main conclusions can be observed as follows: for the MNIST dataset, the highest convergence accuracy of 89.5% is obtained for d=3, m=20 at τ=150. For the CIFAR-10 dataset, the highest convergence accuracy is 31.0% for d=3, m=20 at τ=150. Different values of dd and mm have a large impact on the convergence accuracy. For Fashion-MNIST dataset, the highest convergence accuracy of 68.0% is obtained for d=3, m=20 at τ=150. Hence we can choose group-(6), i.e., d=3, m=20, which is a set of parameters to ensure the complete high-secure function of the algorithm and make the experiment achieve the desirable performance.

(4) *Comparison with State-of-the-Art Methods* To validate the effectiveness of the method in this paper, we compare the method in this paper with existing methods under the same setting. The comparison methods selected are several attack-resistant methods recently proposed in the machine learning community (e.g., *Krum* [29], *Trimmed mean* [30] and *Bulyan* [31]), which aim to be robust to Byzantine faults. Also we selected DPRLDS [32] and FLUK [33] as comparison methods which are also anti-attacking method. Thus, the FedDefense methods in this paper will be tested based on the above four methods in three public datasets using two local models, CNN and Multilayer Perceptron (MLP). In this section, the basic experimental parameters are set as follows: d=3, m=20, the testing is conducted in three public datasets, i.e., MNIST, CIFAR-10, Fashion-MNIST (F-MNIST). And here the number N_p of poisoned clients is 5, 10, and 15, respectively. Tables 2 and 3 are the testing results of maximum of convergence accuracy.

According to Tables 2 and 3, we can observe that the accuracy of the FedDefense model is generally higher than that of other aggregation algorithms under different numbers of malicious clients on the MNIST, CIFAR-10 and Fashion-MNIST datasets. The FedDefense model demonstrates higher accuracy compared to other aggregation algorithms under varying numbers of malicious clients across the MNIST, CIFAR-10, and Fashion-MNIST datasets. The model exhibits strong robustness, with only a slight decrease in accuracy as the number of malicious clients increases. In MLP network experiments, all models experience a decrease in accuracy as the number of malicious clients increases, while CNN network experiments

Table 2 Performance comparison via MLP networks

Dataset	MNIST			CIFAR-10			Fashion-MNIST		
Number of poisoned clients	N_p=5	N_p=10	N_p=15	N_p=5	N_p=10	N_p=15	N_p=5	N_p=10	N_p=15
Krum [29]	67.5%	65.4%	59.1%	28.0%	27.3%	25.1%	58.7%	55.0%	52.3%
Trimmed mean [30]	57.8%	49.0%	47.5%	25.8%	24.2%	23.3%	47.2%	36.8%	27.5%
Bulyan [31]	62.9%	64.0%	61.6%	26.8%	26.4%	25.1%	56.3%	54.2%	47.3%
DPRLDS [32]	60.3%	60.1%	60.2%	27.3%	26.3%	25.4%	56.3%	54.6%	54.6%
HeteroFL [14]	61.5%	61.1%	60.2%	27.3%	26.9%	25.29%	57.1%	57.4%	56.4%
FedDefense	**61.5%**	**61.3%**	**60.5%**	**27.6%**	**26.3%**	**27.1%**	**57.5%**	**56.1%**	**55.3%**

Bold values represent that our approach compares very favorably with other approaches

Table 3 Performance comparison via CNN networks

Dataset	MNIST			CIFAR-10			Fashion-MNIST		
Number of poisoned clients	N_p=5	N_p=10	N_p=15	N_p=5	N_p=10	N_p=15	N_p=5	N_p=10	N_p=15
Krum [29]	85.1%	84.6%	81.5%	30.4%	28.5%	27.2%	68.3%	67.6%	65.3%
Trimmed mean [30]	68.3%	65.2%	63.9%	27.3%	24.5%	22.3%	60.3%	59.3%	58.2%
Bulyan [31]	75.7%	70.9%	66.8%	26.5%	25.3%	25.6%	62.2%	60.0%	57.6%
DPRLDS [32]	87.2%	85.3%	84.6%	30.2%	30.1%	28.3%	69.2%	67.3%	66.7%
FLUK [14]	88.3%	84.1%	85.3%	31.5%	30.3%	29.4%	70.1%	68.3%	67.9%
FedDefense	**88.3%**	**85.6%**	**85.0%**	**31.1%**	**30.5%**	**29.3%**	**70.1%**	**69.2%**	**68.3%**

Bold values represent that our approach compares very favorably with other approaches

show overall higher accuracy with smaller differences, highlighting the advantage of CNN networks on image datasets.

The superior performance of FedDefense under CNN networks can be attributed to its more adaptive defense mechanism to the CNN structure, effective model integration strategy, and utilization of CNN network features. This allows FedDefense to better utilize learned features to improve robustness against adversarial attacks. In contrast, the performance of FedDefense with MLP structures may be weaker due to the complexity of MLP networks, which contain multiple fully connected layers and a larger number of parameters. By comparing with literature [29] and literature [31], our approach highlights the effectiveness and performance advantages in dealing with dishonest client attacks. Our approach not only focuses on traditional aspects such as data filtering or model weighting, but also focuses more on the innovative application of client selection and reward strategies, which improves the robustness and efficiency of the system.

Overall, FedDefense demonstrates stable and superior performance across multiple datasets and varying numbers of malicious clients, essential for ensuring security and privacy in federated learning models. Our approach effectively guards against dishonest clients and outperforms existing methods in terms of efficiency and performance, particularly in handling dishonest client attacks. Future considerations should include evaluating training time and computational resource consumption to comprehensively assess the feasibility and efficiency of different aggregation algorithms. While the FedDefense model shows promise in federated learning, practical applications should consider factors like efficiency, security, and performance for enhanced FL model protection and advancement.

8 Conclusion

Although recent works explored the anti-attack ability of server in FL, such methods are fatal to the accuracy or computational cost. To tackle this challenge, we have refined FL by designing client selection and dynamic reward. We have innovated FL with high security. Our novel approach to FL incorporates several key components, including client subset aggregation, reward updates, client selection steps, and model aggregation, all of which work together to enhance the protection and optimize the performance of the FL system. Through rigorous experimentation and evaluation, we have shown that our proposed defense method not only boosts the overall accuracy of FL models but also significantly improves their defense capabilities against various attacks compared to state-of-the-art techniques.

Author Contributions All authors reviewed the manuscript.

Data Availibility Statement No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no Conflict of interest.

References

1. Duan Y, Fu X, Luo B, Wang Z, Shi J, Du X (2015) Detective: automatically identify and analyze malware processes in forensic scenarios via dlls. In: Proc of ICC
2. Liu T, Di B, An P, Song L (2021) Privacy-preserving incentive mechanism design for federated cloud-edge learning. IEEE TNSE 8(3)
3. Mills J, Hu J, Min G (2022) Multi-task federated learning for personalised deep neural networks in edge computing. IEEE TPDS 33(3)
4. McMahan B, Moore E, Ramage D, Hampson S, Arcas BA (2017) Communication-efficient learning of deep networks from decentralized data. In: Proc of AISTATS
5. Liu B, Cai Y, Zhang Z, Li Y, Wang L, Li D, Guo Y, Chen X (2021) Distfl: distribution-aware federated learning for mobile scenarios. ACM IMWUT 5(4)
6. Wang N, Yang W, Guan Z, Du X, Guizani M (2021) BPFL: a blockchain based privacy-preserving federated learning scheme. In: Proc of GLOBECOM
7. Bagdasaryan E, Veit A, Hua Y, Estrin D, Shmatikov V (2020) How to backdoor federated learning. In: Proc of AISTATS
8. Hei X, Yin X, Wang Y, Ren J, Zhu L (2020) A trusted feature aggregator federated learning for distributed malicious attack detection. Comput Secur 99
9. Gu Z, Yang Y (2021) Detecting malicious model updates from federated learning on conditional variational autoencoder. In: Proc of IPDPS
10. Cao X, Jia J, Gong NZ (2021) Provably secure federated learning against malicious clients. In: Proc of AAAI
11. Abubaker Z, Javaid N, Almogren A, Akbar M, Zuair M, Ben-Othman J (2022) Blockchained service provisioning and malicious node detection via federated learning in scalable internet of sensor things networks. Comput Netw 204:108691
12. Yu P, Liu Y (2019) Federated object detection: optimizing object detection model with federated learning. In: Proc of ICVISP, pp 7–176. ACM https://doi.org/10.1145/3387168.3387181
13. Claici S, Yurochkin M, Ghosh S, Solomon J (2020) Model fusion with kullback-leibler divergence. In: Proceedings of ICML. Proceedings of machine learning research, vol 119, pp 2038–2047. PMLR http://proceedings.mlr.press/v119/claici20a.html
14. Zhu M, Ning W, Qi Q, Wang J, Zhuang Z, Sun H, Huang J, Liao J (2024) Fluk: protecting federated learning against malicious clients for internet of vehicles. In: Euro-Par 2024: parallel processing, pp 454–469. Springer, Cham
15. Yadav C, Bottou L (2019) Cold case: the lost MNIST digits. In: Wallach HM, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox EB, Garnett R (eds) Proc of NeurIPS
16. Zhong Z, Zhou Y, Wu D, Chen X, Chen M, Li C, Sheng QZ (2021) P-fedavg: parallelizing federated learning with theoretical guarantees. In: Proc of INFOCOM
17. Liao Y, Xu Y, Xu H, Wang L, Qian C (2023) Adaptive configuration for heterogeneous participants in decentralized federated learning. In: Proc of INFOCOM
18. Geiping J, Bauermeister H, Dröge H, Moeller M (2020) Inverting gradients - how easy is it to break privacy in federated learning? In: Proc of NeurIPS
19. Fu C, Zhang X, Shouling Ji ea (2022) Label inference attacks against vertical federated learning. In: Proc of USENIX Security
20. Fu L, Zhang H, Gao G, Zhang M, Liu X (2023) Client selection in federated learning: principles, challenges, and opportunities. IEEE IoT-JI 10(24)
21. Nguyen HT, Sehwag V, Hosseinalipour S, Brinton CG, Chiang M, Poor HV (2021) Fast-convergent federated learning. IEEE JSAC 39(1)
22. Li C, Zeng X, Zhang M, Cao Z (2022) Pyramidfl: a fine-grained client selection framework for efficient federated learning. In: Proc of ACM MobiCom
23. Dwork C, Roth A (2014) The algorithmic foundations of differential privacy. Found Trends Theor Comput Sci 9(3–4):211–407
24. Shi L, Shu J, Zhang W, Liu Y (2021) HFL-DP: hierarchical federated learning with differential privacy. In: Proc of GLOBECOM
25. Hu R, Gong Y, Guo Y (2022) Federated learning with sparsified model perturbation: improving accuracy under client-level differential privacy. CoRR abs/2202.07178

26. Stacey T, Ling L, Ka-Ho C, Mehmet-Emre G, Wenqi W (2020) Ldp-fed: federated learning with local differential privacy. In: Proc of EdgeSys
27. Cheng Y, Lu J, Niyato D, Lyu B, Kang J, Zhu S (2022) Federated transfer learning with client selection for intrusion detection in mobile edge computing. IEEE Commun Lett 26(3)
28. Jinkyu Kim BH, Geeho Kim (2022) Multi-level branched regularization for federated learning. In: Proc of ICML
29. Blanchard P, El Mhamdi EM, Guerraoui R, Stainer J (2017) Machine learning with adversaries: byzantine tolerant gradient descent. In: Proc of NIPS
30. Yin D, Chen Y, Kannan R, Bartlett P (2018) Byzantine-robust distributed learning: towards optimal statistical rates. In: Proc of ICML
31. El Mhamdi EM, Guerraoui R, Rouault S (2018) The hidden vulnerability of distributed learning in Byzantium. In: Proc of ICML
32. Hossain MT, Islam S, Badsha S, Shen H (2021) Desmp: Differential privacy-exploited stealthy model poisoning attacks in federated learning. In: Proc of MSN
33. Lu J, Hu S, Wan W, Li M, Zhang LY, Xue L, Jin H (2024) Depriving the survival space of adversaries against poisoned gradients in federated learning. IEEE TIFS 19
34. PySyft. https://github.com/OpenMined/PySyft. Accessed in June, 2023 (2023)