IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 5 January 2023, accepted 16 January 2023, date of publication 23 January 2023, date of current version 3 February 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3238823

SURVEY

Poisoning Attacks in Federated Learning: A Survey

GEMING XIA, JIAN CHEN, CHAODONG YU, AND JUN MA
College of Computer Science and Technology, National University of Defense Technology, Changsha 410003, China
Corresponding author: Jian Chen (chenjian@nudt.edu.cn)

ABSTRACT Federated learning faces many security and privacy issues. Among them, poisoning attacks can significantly impact global models, and malicious attackers can prevent global models from converging or even manipulating the prediction results of global models. Defending against poisoning attacks is a very urgent and challenging task. However, the systematic reviews of poisoning attacks and their corresponding defense strategies from a privacy-preserving perspective still need more effort. This survey provides an in-depth and up-to-date overview of poisoning attacks and corresponding defense strategies in federated learning. We first classify the poisoning attacks according to their methods and targets. Next, we analyze the differences and connections between the various categories of poisoning attacks. In addition, we classify the defense strategies against poisoning attacks in federated learning into three categories and analyze their advantages and disadvantages. Finally, we discuss the privacy protection problem in poisoning attacks and their countermeasure and propose potential research directions from the perspective of attack and defense, respectively.

INDEX TERMS Federated learning, distributed machine learning, poisoning attacks, defense of poisoning attacks.

I. INTRODUCTION

Machine learning technology has seen a boom as people's ability to collect, and process data has improved. Artificial intelligence technologies are widely used in various fields, such as computer vision [1], natural language processing [2], and recommendation systems [3], which have significantly changed our lives. Nevertheless, most existing AI techniques rely on rich and high-quality datasets to train a highly accurate machine learning model. For example, to achieve unprecedented accuracy of the GPT-3 model, the OpenAI team used a 45 TB dataset (namely the CommonCrawl dataset) of compressed plaintexts [2].

However, in the real world, most enterprises face the problem of insufficient samples and low quality when training models, which is not enough to train a high-performance machine learning model. One possible solution is to share data among multiple enterprises and train models. However, the training dataset often contains large amounts of private data, including private or sensitive information of users, such as bank card accounts, shopping records, and medical diagnoses. Once this data is shared, the user cannot restrict the scope of data usage, which may lead to the user's private data being misused by third parties without the user's approval. Due to these concerns, many countries have enacted laws to limit the scope of use of user data, such as Health Insurance Portability and Accountability Act (HIPAA) [4] and General Data Protection Regulation (GDPR) [5]. However, a side effect of these laws is that they hinder data flow and create data silos.

To address this problem, Google developed a distributed machine learning framework called federated learning (FL) in 2016, which allows each client to jointly train a model without sharing data [6]. The difference between federated learning and other distributed machine learning frameworks is that in federated learning, each client's data is private and cannot be accessed by others. In a typical federated learning framework [6], the server first sends a global model to all selected clients. Then, these clients train the global model using their local dataset and upload the trained model to

The associate editor coordinating the review of this manuscript and approving it for publication was Mehul S. Raval.

the central server. After the server receives the models of all selected clients, the server then updates the global model by averaging the uploaded models. In the training process, only the data owner can access its local data. Therefore, federated learning is considered effective in protecting user privacy during training. However, further research shows that federated learning also faces many security and privacy risks [7], [8], [9], [10], such as poisoning attacks [11], [12], [13], [14], [15] and privacy leakage [16], [17], [18], [19], [20], [21]. This paper focuses on the risk of poisoning attacks in federated learning. Moreover, we also consider the issue of privacy protection when discussing the problem of poisoning attacks in federated learning.

Poisoning attack was proposed long before the advent of federated learning. The world's first poisoning attack was a data poisoning attack against a support vector machine (SVM) proposed by Battista et al. [22]. They create poisoned data by flipping the label of training data, which significantly decreases the performance of the trained model on the test set. Although the initial poisoning attack was performed on a single machine, the extension of Battista et al.'s approach to the distributed environment still works well [21]. Furthermore, with further research on federated learning, researchers proposed another poisoning attack method, namely the model poisoning attack. In this case, the attacker can directly manipulate the parameters of the local model and upload the crafted malicious model to the server to poison the global model. Since many clients are involved in training in federation learning, it is impossible to guarantee that all clients are reliable. Therefore, we must consider the defense strategy against poisoning attacks when deploying a large-scale federated learning system.

While several researchers have discussed poisoning attacks in their surveys, few have analyzed the privacy issues that may arise from poisoning attacks and poisoning attack defense strategies. The motivation of this survey is to fill this research gap by investigating defense strategies for poisoning attacks and poisoning attacks. We intend to cover the most representative poisoning attack methods and defense strategies. After surveying the papers published in recent years, we classify the defense strategies against poisoning attacks and poisoning attacks separately, and the classification result is shown in Fig. 1. First, we classified poisoning attacks into targeted, semi-targeted, and unargeted poisoning attacks according to the attacker's intention; and classified poisoning attacks into data poisoning attacks and model poisoning attacks according to the attack methods. Then, we classify the defense strategies against poisoning attacks into model analysis, byzantine robustness aggregation, and verification-based approaches. Finally, we discuss the impact of privacy protection in federated learning on the defense strategies against poisoning attacks and poisoning attacks.

The contribution of this paper can be summarized as follows:

- We surveyed the existing poisoning attack works and categorized them according to the methods of implementing the attack and the attackers' intention, respectively.
- We classified the defense strategies for federated learning into three categories: 1) Model analysis, 2) Byzantine robust aggregation (BRA), and 3) Verification-based. In addition, we also point out the strengths and weaknesses of each defense strategy, which will benefit future research.
- We summarize existing poisoning attack methods and defense strategies and point out their relevance to privacy protection and potential research directions.

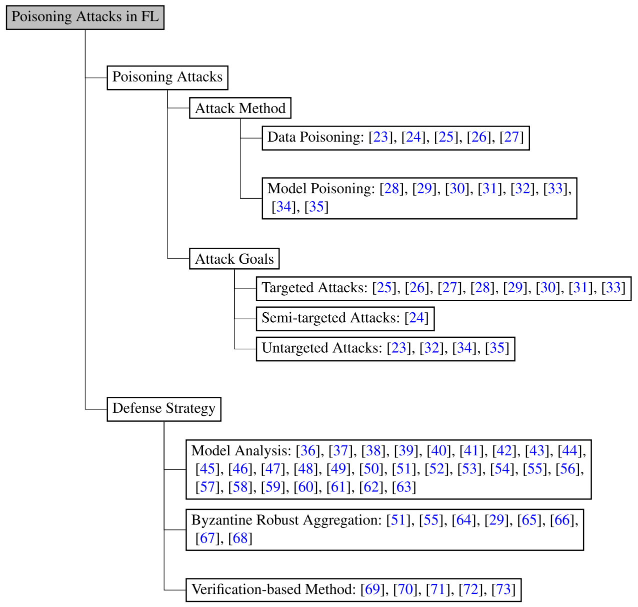

Poisoning Attacks in FL
Poisoning Attacks
Attack Method
Data Poisoning: [23], [24], [25], [26], [27]
Model Poisoning: [28], [29], [30], [31], [32], [33], [34], [35]
Attack Goals
Targeted Attacks: [25], [26], [27], [28], [29], [30], [31], [33]
Semi-targeted Attacks: [24]
Untargeted Attacks: [23], [32], [34], [35]
Defense Strategy
Model Analysis: [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63]
Byzantine Robust Aggregation: [51], [55], [64], [29], [65], [66], [67], [68]
Verification-based Method: [69], [70], [71], [72], [73]

FIGURE 1. Our taxonomy for poisoning attacks and their defense strategies in federated learning.

The rest of this paper is organized as follows. First, we compare our work with recent related reviews in Section II. Second, we introduce federated learning in Section III. Third, we describe the existing poisoning attack methods in Section IV. Fourth, we describe existing defense strategies against poisoning attacks in federated learning in Section V. Fifth, we discuss the challenges and some promising future research directions in Section VI. Finally, we summarize our work in Section VII.

II. RELATED WORK

In recent years, several surveys have discussed poisoning attacks and their defense strategies in federated learning. We select some representative works and present their main points and shortcomings in detail.

Bouacida et al. [8] provided a comprehensive survey on security vulnerabilities at each stage of federated learning and existing defense strategies and pointed out that there are no simple defenses against attacks on FL systems due to privacy restrictions. However, the poisoning attack is one of the many security vulnerabilities they discussed and is only briefly described in their paper.

TABLE 1. Summary of existing surveys.

Literature	Year	Focus
[8]	2021	Security vulnerabilities in FL
[12]	2022	Defense strategies against model poisoning attacks
[15]	2022	Attacks and Defenses of Privacy and Robustness in Federated Learning
[74]	2022	Privacy and security issues related to federated learning

Zhilin et al. [12] discussed defense strategies against model poisoning attacks in federated learning. They classified existing defense strategies into two categories: evaluation methods for local model updates and aggregation methods for the global model. Their survey described defense strategies against model poisoning attacks while ignoring defense strategies against other poisoning attack methods, such as data poisoning attacks.

Lingjuan et al. [15] studied poisoning attacks and their defense strategies in federated learning from the perspective of privacy protection and robustness aggregation. They proposed that secure aggregation FL for privacy purposes is more vulnerable to poisoning attacks because servers cannot check for individual updates. However, they did not specify whether a given defense strategy applies to security aggregation.

Gosselin et al. [74] enumerated the security and privacy-preserving problems in federated learning. For example, poisoning attacks, backdoor attacks, and Generative Adversarial Network (GAN)-based attacks. However, they need to go further into each problem, especially the problem of poisoning attacks in federated learning.

As shown in Table 1, many researchers have discussed poisoning attacks in federated learning. However, most studies on defense strategies against poisoning attacks have focused on discussing existing methods, ignoring the most critical privacy-preserving property of federated learning. Thus, the negative impact of existing works on privacy protection is ignored.

In addition, most current investigations into poisoning attacks are technology-driven. They divided them into model poisoning attacks and data poisoning attacks based solely on the method of poisoning attacks. These are just how the attacker implements the poisoning attack and do not describe the target of the attacker's attack well.

For these reasons, we comprehensively discuss poisoning attacks in federated learning and their defense strategies and evaluate existing defense strategies from a privacy perspective, which will benefit the development of more secure federated learning methods.

III. FEDERATED LEARNING

Federated learning is a distributed machine learning paradigm that enables clients to work together to train a shared machine learning model while keeping the training data decentralized. This innovative approach to machine learning provides a distinct advantage over traditional centralized machine learning, as it reduces the risk of privacy leakage.

A typical FL system consists of one central server and many clients. The central server coordinates the clients to complete the training and aggregate all local models to update the global model. And the clients train the local model with their local data and upload the trained models to the server.

Let m ($m \in 1, 2, 3, \cdots, M$) denote a client, where M is the total number of clients. The training dataset of the client i is denoted by D_i, and i is the client id. Depending on the FL application scenario, the datasets may be independently and identically distributed (IID) or non-independently and identically distributed (non-IID). We can summarize the implementation of FL in the following three steps:

1) Client selection. In this step, the server randomly (or selects clients based on their response latency [75], contribution [76], etc.) selects some clients to participate in this training round and distributes the initialized global model to all selected clients.

2) Local model training. In this step, the selected clients train their local model with their local data and send the trained model to the server in parallel.

3) Aggregation. After the server collects all (or most) of the models uploaded by clients, it uses these uploaded models to update the global model according to a specific aggregation algorithm.

FL is in an iterative learning process, repeating the training step 1) to 3) above until the global model converges or reaches the maximum number of iterations.

We illustrate the FL process by a typical FedAVG [6] algorithm with the pseudo-code shown in Algorithm 1. At the beginning of each training round, the server selects n^k clients to participate in federated learning. We use M_k to denote the set of clients selected at the k-th iteration, $k \in 1, 2, 3, \cdots, K$, K is the total number of training rounds. After selecting the clients, the server sends an initialized global model $\boldsymbol{w}_0^k$ to client $m \in M_k$. Then, each client m starts training the local model using its local data (As shown in lines 9 to 14). After that, each client sends the trained model $\boldsymbol{w}_m^k$ to the central server. After the server receives all the local models uploaded by clients, it uses an aggregation algorithm to update the global model (As shown in line 7).

From the above description, we can find that the direction of the global model update depends on the directions of the local models uploaded by each client. As shown in Fig. 2, if an attacker manipulates some of these clients to modify the models before uploading them to the central server by directly (for example, altering the model parameters) or indirectly (such as by flipping the label of the training data) way, it will influence the direction of the global model update and finally cause the non-convergence of the global model.

IV. POISONING ATTACK IN FEDERATED LEARNING

As shown in Fig. 3, a poisoning attack in federated learning refers to the attacker modifying the model uploaded by the client to the central server during the aggregation process of FL, either by directly or indirectly ways, and making the global model update in the wrong direction.

Researchers have proposed various methods for federated learning poisoning attacks in the past few years. We clas-

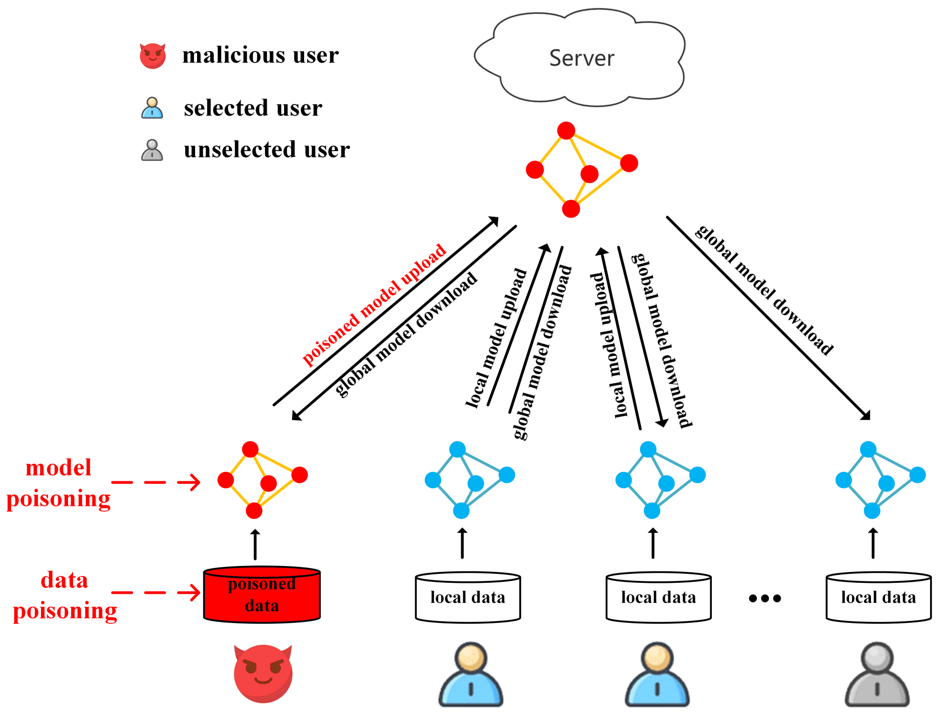


FIGURE 2. Example of poisoning attack. During the federated learning training process, an attacker can execute a poisoning attack by modifying the client's local data or directly modifying the local model parameters.

sify these methods into model poisoning attacks and data poisoning attacks according to the ways of modifying the local model parameters and into targeted, semi-targeted, and untargeted poisoning attacks according to the attacker's intention. We describe the details of each classification below.

Algorithm 1 FedAVG [6]

```
Input: client's number $M$, local batch-size $B$, local epochs
    $E$, learning rate $\eta$
Server executes:
 1: initialize $w_0^0$
 2: for each round $k \in 1, 2, 3, \cdots, K$ do
 3:     $M_k \leftarrow$ (random set of $n^k$ client)
 4:     for each client $m \in M_K$ in parallel do
 5:         $w_m^{k+1}$ = ClientUpdate(m, $w_0^k$)
 6:     end for
 7:     $w_0^{k+1} = \sum_{m \in M_k} \frac{n_m}{n} w_m^{k+1}$
 8: end for
ClientUpdate(m, $w$): // Run on client $m$
 9: $\mathfrak{B} \leftarrow$ (split $D_m$ into batched of size $B$)
10: for each local epoch $i$ from 1 to $E$ do
11:     for batch $b \in \mathfrak{B}$ do
12:         $w \leftarrow w - \eta \Delta \mathcal{L}(w; b)$
13:     end for
14: end for
```

A. CLASSIFICATION ACCORDING TO THE POISONING ATTACK METHOD

An intuitive classification criterion is to divide poisoning attacks into data poisoning attacks and model poisoning attacks based on the method by which the attacker modifies the local model parameters to generate the poisoning model.

1) DATA POISONING ATTACK

In the data poisoning attack, an attacker cannot directly modify the local model parameters and can only perform the poisoning attack by falsifying or tampering with the training data. We can divide the data poisoning attack methods into two major categories: clean label attack [24], [25], [27] and dirty label attack [23], [24]. The clean label attack assumes that the attacker cannot modify the labels of the training samples, and this assumption is based on the existence of a verification process for the class to which the samples belong [73].

a: CLEAN LABEL ATTACK

Attackers can implement a clean label attack by modifying the samples in the training set and, for example, adding noise to the training data. Moreover, an attacker can modify the samples in a specific way to inject a backdoor into the global model. For instance, Ali et al. [25] proposed the following method to modify samples:

$$\boldsymbol{p} = \arg\min_{\boldsymbol{x}} ||f(\boldsymbol{x}) - f(\boldsymbol{t})||_2^2 + \beta||\boldsymbol{x} - \boldsymbol{b}||_2^2 \quad (1)$$

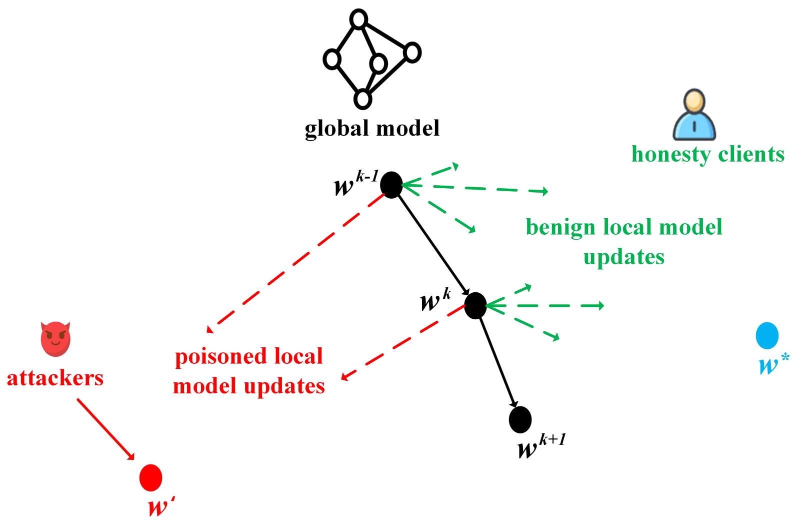


FIGURE 3. Poisoning attack in federated learning [35]. A benign model uploaded by an honest user allows the global model to be updated along the optimal direction (w^*). In contrast, a poisoned model uploaded by an attacker causes the global model to be updated in the wrong direction.

where $f(\boldsymbol{x})$ denotes the function propagating an input $\boldsymbol{x}$ through the network to the penultimate layer (before the softmax layer), and $\boldsymbol{b}$ is the base instance. Our goal is to make the poisoned instance $\boldsymbol{p}$ look like the base instance in the eyes of the classifier (β parameterizes the degree to which this is so) to poison the global model. Due to the high complexity and nonlinearity of f, it is possible to find an example $\boldsymbol{x}$ that "collides" with the target in feature space while simultaneously being close to the base instance $\boldsymbol{b}$.

In addition, an attacker can use public data to generate fake data. In the federated recommendation system, Dazhong et al. [27] use public interactions to approximate the user's feature vector. An attacker can use the approximated user feature vector to train a malicious model. However, these methods all assume that the data distribution is IID. Once the training data is non-IID, the attacker cannot execute the attack in the above way.

To solve this problem, Jiale et al. [26] use a generative adversarial network to generate samples similar to those of other clients and then use these fake samples to execute an attack. They use the global model as the discriminator D and generate the sample x_{fake} by a data generator G. If the discriminator considers x_{fake} belongs to the target class, x_{fake} is added to the malicious dataset. Otherwise, update G using the output of the discriminator D. With this approach, an attacker can execute a poisoning attack without prior knowledge.

b: DIRTY LABEL ATTACK

A typical example of a dirty label attack is label flipping. Virat et al. [23] argued that there are two kinds of label-flipping strategies, namely static label flipping (SLF) and dynamic label flipping (DLF). In SLF, an attacker can flip the label of the digit image of "1" in the MNIST dataset to "7" [22]. This approach is straightforward, and the attacker can design a label-flipping strategy according to his intentions. It is considered the default attack method in many studies of defense strategies against poisoning attacks.

However, SLF is inefficient, and some researchers have proposed DLF schemes to improve the attack's efficiency. Yuwei et al. [24] proposed the attack distance-aware attack (ADA) to enhance poisoning attacks by finding optimal target classes in the feature space. They define the attack distance as follows:

$$\boldsymbol{\mu}_c = \frac{1}{|D_{mal}(c)|} \sum_{\boldsymbol{x} \in D_{mal}(c)} \phi(\boldsymbol{x}) \tag{2}$$

$$AD(c, c') = ||\boldsymbol{\mu}_c - \boldsymbol{\mu}_{c'}||_2 \tag{3}$$

Equation (2) represents the mean of feature vectors in class c, $|D_{mal}(c)|$ is the number of samples of class c in the malicious dataset D_{mal}, and ϕ is the feature extraction function. The attacker can calculate the attack distance (AD) between all c and c' based on (3) and then use the c' with the smallest AD as the target class. The idea of the attack in this setup is that if the AD between the source class c_S and the target class c_T is small, the scale of the malicious updates required in the adversary's local model is also small. Then the attack has a greater chance of surviving the aggregation of overwhelming legitimate updates, resulting in a more significant effect.

2) MODEL POISONING ATTACK

In a model poisoning attack, the attacker uses a more direct way to manipulate the client's local model parameters. For instance, the attacker can add random noise to the local model to affect the convergence of the global model [34], [35]. Moreover, the attacker can also manipulate the update of the global model using the model replacement method

[29], [30], [31]. In model replacement, the attacker attempts to replace the global model $w_0^{(k+1)}$ with the malicious model $\widetilde{w}_0^{k+1}$. As the global model converges, $\sum_{m\in M_k}(w_m^k - w_0^k) \approx 0$. Therefore, Eugene et al. propose the following way to upload a local model $\widetilde{w}_m^k$ to replace the global model $w_0^{(k+1)}$ with $\widetilde{w}_0^{k+1}$ [30]:

$$w_m^{k+1} = w_m^k - \eta w_0^k \tag{4}$$

$$\begin{aligned}\widetilde{w}_m^k &= \frac{n}{\eta}\widetilde{w}_0^{k+1} - (\frac{n}{\eta} - 1)w_0^k - \sum_{i\in M_k, i\neq m}\Delta w_m^k \\ &\approx \frac{n}{\eta}(\widetilde{w}_0^{k+1} - w_0^k) + w_0^k\end{aligned} \tag{5}$$

To generate a malicious model $\widetilde{w}_0^{k+1}$, the attacker can use a specific dataset (e.g., different kinds of green cars labeled as birds [29]) or train the model in a benign dataset by projected gradient ascent (while the benign clients use stochastic gradient descent) algorithm [23], [67], [77]. To avoid detection, Virat et al. [23] also state that model updates should not deviate too far from benign updates to avoid detection.

Moreover, Xingchen et al. [33] proposed an optimization-based model poisoning attack method. They injected malicious neurons into the neural network's redundant space using the objective function's regularization term. The principle of this approach is that only a tiny fraction of neurons changed during the training phase, while the vast majority of neurons were close to zero. Those unchanged neurons are called the redundant space of neural networks. Therefore, the attacker can perform poisoning attacks effectively by injecting malicious neurons into the global model. Moreover, this method can resist catastrophic forgetting in neural networks. However, the biggest problem with this approach is that malicious clients need to compute the Hessian matrix during preparing an attack, which is non-trivial and time-consuming.

B. CLASSIFICATION ACCORDING TO THE INTENTION OF THE ATTACKERS

In addition to the different attack methods, a poisoning attack can also be classified according to the attacker's intention.

The attacker's goals of implementing a poisoning attack can be classified into two categories: 1) reducing the accuracy of the global model on all tasks and 2) affecting the performance of the global model on specific tasks. Therefore, according to the attacker's intention, we can classify poisoning attacks into targeted, semi-targeted, and untargeted poisoning attacks.

1) TARGETED POISONING ATTACK

In a targeted poisoning attack, the attacker's goal is to affect the performance of the global model on a specific task and not on other tasks. A typical example of a targeted poisoning attack is a backdoor attack. A backdoor attack is a way to inject a malicious model into the existing model while retaining the accuracy of the other tasks [11].

Implementing a targeted poisoning attack can be achieved by launching a poisoning attack either on the training set [24], [44] or on the local model [28], [29], [30], [31], [33]. Most of the approaches here have been described in the previous subsection and will not be repeated here. It is worth noting that many research results show that the success rate of targeted poisoning attacks depends heavily on the capacity of the neural network [13], [29], [30], [33]. In particular, the key idea of the method proposed by Xingchen et al. [33] is to inject malicious neurons into the redundant space of the neural network. Therefore, a potential defense strategy against targeted poisoning attacks is to reduce the redundant space in the neural network or clean up the redundant neurons.

2) SEMI-TARGETED POISONING ATTACK

In the semi-target attack, the attacker is assigned a class (source class), and the attacker aims to poison the global model so that samples with the source class are identified as a class other than the source class.

Unlike a targeted attack, the attacker is free to choose the target class to maximize the attack performance in a semi-target attack. This type of attack can occur in the real world in various situations. For example, an attacker sending an unauthorized advertising email aims to have the email identified as belonging to some benign category rather than spam.

We can formalize semi-targeted as follows [24]:

$$\arg\max_m f(\boldsymbol{x})_m = \begin{cases} c^*, & y = c_S \\ y, & otherwise \end{cases} \tag{6}$$

where c^* is the class that makes the attack the most effective.

3) UNTARGETED POISONING ATTACK

In an untargeted poisoning attack, the attacker's goal is to degrade the performance of the global model on all tasks or make the global model fail to converge. The most straightforward method to achieve this goal is adding random noise to the local model. For example, Xiaoyu et al. [35] proposed a way to exploit the vulnerability of user authentication in the FL system to perform a poisoning attack by uploading random noise through an injected fake client.

One possible solution to this attack is to analyze the model uploaded by the client, and the malicious model often deviates from the benign model. However, this problem becomes tricky when considering the privacy of the client. As Md Tamjid et al. [34] argued, when FL uses differential privacy techniques to protect clients' privacy, an attacker can add fake noise to the local model to perform a stealthy and persistent poisoning attack. One effective way, in this case, is using the BRA method. The defense strategy for this case can be referred to as supporting security aggregation in Table 3.

Although adding random noise is relatively simple to implement, however, there are inefficiencies in implementing attacks using this approach. To implement the attack efficiently, [23], [67], [77] studied using a gradient ascent algorithm to train a local model on a benign dataset. The

TABLE 2. Summary of poisoning attacks.

Literature	Category	Goal	Data distribution	Persistence	Year	Attack strategy	Stealthiness strategy
[28]	MP	targeted	IID	N	2019	boosting malicious updates	alternating minimization
[29]		targeted	non-IID	N	2019	boosting malicious updates	bound the norm of updates
[30]		targeted	non-IID	Y	2020	boosting malicious updates	add an anomaly detection term to attacker's loss function
[31]		targeted	IID	Y	2020	boosting malicious updates	decompose global trigger pattern
[32]		untargeted	IID/non-IID	N	2022	use RL to select optimal attack policy	adjust attack policy to bypass defense
[33]		targeted	non-IID	Y	2021	inject adversarial neurons	alternating minimization
[34]		untargeted	non-IID	Y	2021	add fake noise	none
[35]		untargeted	non-IID	N	2022	add fake noise	use fake clients to execute the attack
[23]	DP	untargeted	non-IID	Y	2022	dirty label attack	bound the norm of updates
[24]		semi-targeted	IID	N	2022	dirty label attack	bound the norm of updates
[25]		targeted	IID	N	2018	clean label attack	none
[26]		targeted	non-IID	Y	2020	clean label attack	execute a single-round attack
[27]		targeted	IID	N	2022	clean label attack	reduce side effects on recommendation accuracy

MP: model poisoning attack, DP: data poisoning attack

model trained in this way is in the opposite direction of the model update method trained by the honest client (which uses the gradient descent algorithm) and therefore reduces the performance of the global model.

Another efficient attack method was proposed by Henger et al. [32]. In their method, they use a gradient inversion-based inference attack to approximate the distribution of the clients' aggregated data. Then the learned distribution is used to build a simulator of the FL environment, which is utilized to learn an adaptive attack policy through reinforcement learning. Training an optimal poisoning attack strategy by reinforcement learning achieves outperforming performance even in the non-IID setting (when the data distribution of other clients cannot be accurately approximated). Moreover, Henger et al.'s approach can learn strong attacks automatically when the server uses a robust aggregation policy.

To avoid detection, Virat et al. [23] propose a method that projects the malicious gradients on a ball of radius r around the origin, i.e., bound the scaled malicious gradients within a norm $||\Delta \boldsymbol{w}_m^k|| \leq r$, where r is the average of norms of the benign gradients. When r is large, it increases the risk of being detected, and when r is small, it reduces the effectiveness of the attack. Therefore, the optimal strategy is to choose the appropriate r according to the detection strength.

C. SUMMARY

Table 2 lists some representative poisoning attack methods. In addition to the classification methods mentioned above, we analyze these methods' data distribution, persistence, and stealthiness strategy as follows.

1) DATA AND MODEL POISONING ATTACK

Both data poisoning attacks and model poisoning attacks are ways to modify the client's local model. However, model poisoning attacks tend to work better in practice because they can directly modify local model parameters [11], [23], [33], [40]. For example, the experimental results of Eugene et al. [30] showed that in a given scenario, an attacker controlling 1% of clients to perform a model poisoning attack could achieve the same (or even better) effect as controlling 20% of clients to perform a data poisoning attack.

There are two possible reasons that make model poisoning attacks more effective than data poisoning attacks. First, Xingchen et al. [33] found that only a tiny fraction of neurons were changed during the training phase, while most neurons were close to zero. They can implement a more stealthy and persistent poisoning attack using these redundant neurons, which requires the ability to modify the model parameters directly. Second, model poisoning attacks often multiply the local gradient by a factor before uploading it to the server to amplify the impact of the malicious gradient [28], [29], [30], [31]. This approach increases the risk of being detected by the server, but the right choice of amplification factor can effectively improve the attack's efficiency.

However, data poisoning attacks also have their advantages. For example, data poisoning attacks are more intuitive, and attackers can implement label flipping according to their intentions. Moreover, attackers can often combine data poisoning attacks with model poisoning attacks. For example, the label-flipping strategy is first used to train the poisoned model, and then the model poisoning attack is used to maximize the effect of the attack [30].

2) TARGETED, SEMI-TARGETED, AND UNTARGETED POISONING ATTACK

Targeted poisoning attacks, semi-targeted poisoning attacks, and untargeted poisoning attacks are distinguished according to the attacker's intention. The attacker can choose which attack to make according to his intention. For example, an attacker who wishes to make the global model non-convergence can perform an untargeted poisoning attack. Since untargeted poisoning attacks affect all tasks, they are more easily detected by the server.

On the other hand, targeted poisoning attacks are highly stealthy and more challenging to detect, especially when the data is non-IID. However, the performance of targeted poisoning attacks is usually class-sensitive and varies with the target classes. Usually, the effectiveness of the attack decreases when the attack is shifted to a different target class. The semi-targeted attack can improve the performance of the attack by fixing the source class and choosing the optimal target class.

3) DATA DISTRIBUTION AND STEALTHINESS STRATEGY

In a poisoning attack, the data distribution can not only affect the effectiveness of the attack but also determine the success or failure of the attack.

First, a server is more likely to detect an attack when the data is IID. However, IID data facilitates the execution of data poisoning attacks (especially clean label attacks). Second, in the non-IID setting, both the attacker and the server faced significant challenges but affected the server even more. In general, the execution of poisoning attacks in the IID setting is more challenging.

In order to prevent attacks from being detected by the server, various stealth strategies have been proposed. A widely used strategy is alternating minimization [28], [33]. The alternating minimization approach decouples the poisoning attack target from the stealthiness target. An attacker can train a malicious model and then use measures such as bounding the norm of the malicious gradient to provide greater stealth.

Data distribution also affects the stealthiness of poisoning attacks. Typically, non-IID data distribution provides a good cover for the attack [78]. This is because when the data is Non-IID it can bias the model more, especially when trained with SGD [79]. In addition, many attackers avoid detection by bounding the norm of their malicious gradients, the range of which is strongly influenced by the data distribution.

4) PERSISTENCE OF POISONING ATTACK

Another critical problem is the persistence of the poisoning attack. However, this is a very challenging task. Due to the catastrophic forgetting property of neural networks, an attacker's malicious updates may be forgotten during subsequent training.

One of the simplest methods to solve this problem is to perform poisoning attacks continuously. However, the increased number of attacks increases the risk of being detected. Although Thien et al. [80] propose that it can mix a small number of malicious updates with numerous benign updates to avoid server detection, the server can still identify the attack by comparing historical updates from the client.

Moreover, Xingchen et al. [33] propose a feasible solution to implement a persistent poisoning attack: finding redundant spaces in the neural network (neurons that remain unchanged during training). And then embed malicious updates into the redundant spaces to avoid updates from benign clients overwriting malicious updates.

V. DEFENSE STRATEGIES OF POISONING ATTACK IN FEDERATED LEARNING

In this section, we analyze the existing defense strategies against the poisoning attack. Although detection and defense of poisoning attacks are two different phases, this paper considers them the same phase because they usually work together to protect FL. We classify defense strategies for poisoning attacks into three categories: 1) Model analysis, 2) Byzantine robust aggregation, and 3) Verification-based method.

A. MODEL ANALYSIS

Model analysis methods are based on the assumption that there are significant differences between the poisoned and benign models and that there is some way to distinguish these differences. This assumption is always valid because benign models and malicious models are inherently different from each other. There are many model analysis methods for defending against poisoning attacks. We will describe most of them in the following few paragraphs.

1) SIMILARITY-BASED METHOD

The first one is a similarity-based malicious model detection algorithm, which considers that statistical methods (such as euclidean distance [40], [52], cosine similarity [37], [38], k-means [46], Pearson correlation coefficient [49], [51], etc.) can be used to measure the differences between benign and malicious models [36], [80]. In [52] and [40], they cluster all clients by calculating the euclidean distance between the local models uploaded by each client. Di et al. [52]. argued that if two models are close enough, both models are likely to belong to the same category (malicious or benign).

Furthermore, Siping et al. [54] propose an alternative way of analyzing whether a local model is malicious. They developed a lightweight, unsupervised anomaly detection method based on SVM. The main idea of their method is to estimate a nonlinear decision boundary to find anomalies using an appropriate kernel function and soft margins.

The biggest problem with the above approach is that it assumpts that the attacker is in the minority of all clients. To alleviate this problem, Xiaoyu et al. [40] proposed FLTrust, where the server trains a server model on the dataset collected by itself. In particular, FLTrust assigns a trust score to a local model update based on its direction similarity with the server model. They believe that the lower the trust score of a client, the higher the probability that the client is malicious.

In addition to clustering the user-uploaded models, we can also directly analyze the user-uploaded models to detect poisoning attacks. This approach relies heavily on human-discovered patterns to defend against poisoning attacks. For example, Clement et al. [53] found that the gradients uploaded by honest clients were more diverse than those uploaded by malicious clients.

The difference between malicious and benign model parameters can identify malicious models. For example, Wenxin et al. [43] proposed a method based on the isolated forest algorithm to see whether a model is malicious. They randomly select a value in the model and divide the set of parameters into two subsets presented in two leaf nodes based on the selected values. Since the number of outliers is always smaller and different from the normal data points, the outliers are always closer to the root. Similar to [45], [53], and [50] argues that the diversity of output logits of the attacker's local model differs significantly from

TABLE 3. The defense strategies for difference attack methods.

Literature	Category	Attack	Data distribution	SA support	Year	Method
[36]	MA	MP/DP	non-IID	non-SA	2020	similarity-based
[37]		MP	non-IID	non-SA	2021	similarity-based
[38]		MP	non-IID	SA	2022	similarity-based
[39]		MP	non-IID	non-SA	2020	similarity-based
[40]		MP	non-IID	non-SA	2020	similarity-based
[41]		MP	non-IID	non-SA	2022	similarity-based
[42]		DP	non-IID	non-SA	2021	similarity-based
[43]		DP	IID	non-SA	2021	similarity-based
[44]		DP	non-IID	non-SA	2020	similarity-based
[45]		DP	IID	non-SA	2020	similarity-based
[46]		DP	IID	non-SA	2021	similarity-based
[47]		DP	non-IID	non-SA	2022	similarity-based
[48]		DP	IID	non-SA	2021	similarity-based
[49]		DP	IID	SA	2020	similarity-based
[50]		DP	IID	non-SA	2022	similarity-based
[52]		DP	IID	non-SA	2019	similarity-based
[53]		DP	non-IID	non-SA	2018	similarity-based
[40]		DP	non-IID	non-SA	2020	similarity-based
[54]		DP	non-IID	non-SA	2022	similarity-based
[56]		MP	non-IID	SA	2021	performance evaluation
[57]		MP	IID	non-SA	2020	performance evaluation
[58]		MP	IID	non-SA	2022	performance evaluation
[59]		MP/DP	non-IID	non-SA	2022	performance evaluation
[42]		DP	non-IID	non-SA	2021	performance evaluation
[58]		DP	non-IID	non-SA	2020	performance evaluation
[60]		DP	IID	non-SA	2021	performance evaluation
[61]		DP	IID	non-SA	2019	performance evaluation
[62]		DP	IID	non-SA	2021	performance evaluation
[63]		MP	non-IID	SA	2022	performance evaluation
[51]	MA/BRA	DP	IID	SA	2021	similarity-based/remove extreme values
[55]		MP	non-IID	non-SA	2020	performance evaluation/remove extreme values
[64]	BRA	DP	non-IID	non-SA	2018	gradient clipping
[29]		MP	non-IID	non-SA/SA	2019	gradient clipping/differential privacy
[65]		MP	non-IID	non-SA	2022	gradient clipping
[66]		DP	IID	SA	2020	differential privacy
[67]		MP	IID	non-SA	2018	remove extreme values
[68]		DP	non-IID	SA	2022	remove extreme values
[69]	VB	MP	IID	SA	2020	trusted execution environment
[70]		MP	IID	non-SA	2020	user verification
[71]		DP	IID	non-SA	2020	data verification
[72]		DP	IID	non-SA	2020	data verification
[73]		DP	IID	SA	2020	data verification

MA: model analysis, BRA: byzantine robust aggregation, VB: verification-based method
MP: model poisoning attacks, DP: data poisoning attacks
SA: secure aggregation

that of the normal model. Therefore, the server can execute an additional logit-based prediction model on the server side to predict whether a given logit belongs to a malicious client.

Moreover, Zaixi et al. [41] proposed the FLDetector framework to detect malicious clients by evaluating the consistency of the update direction of the clients' local models over multiple iterations. Roughly speaking, the server predicts the client's model updates in each iteration based on the historical model updates. If the model updates received from the client and the predicted model updates are inconsistent (calculated based on euclidean distance) in multiple iterations, the client is marked as malicious. Furthermore, the malicious updates will be removed.

It is worth noting that if the server can access the client's local model directly, there is a great risk of privacy leakage because the server may recover the original training data from the client's gradients [16], [17], [18], [19]. Therefore, many FL frameworks use secure aggregation [81], [82], [83], [84] algorithms in the aggregation process, at which point most of the poisoning attack defense methods mentioned above will be invalid.

To solve this problem, [38], [49], [51] used homomorphic encryption to evaluate the local model securely.

Zhuoran et al. [38] designed a privacy-preserving defense strategy called ShieldFL, which uses two-trapdoor homomorphic encryption to defend against poisoning attacks. They used two servers in ShieldFL to compute the cosine similarity of the encrypted local models. One server S_a has the public key for homomorphic encryption, and the other server S_b has both public and private keys. ShieldFL is implemented in the following way: First, S_a adds a random number to the ciphertext as a mask and sends the masked model to S_b. When S_b receives the model, it first decrypts the encrypted model with the private key, then encrypts the calculated cosine similarity value and sends it back to S_a. Finally, S_a removes the noise in the cosine similarity to get the final result. The method of [49], [51] is similar to [38]. The difference is that [49] and [51] calculate the Pearson correlation coefficient between the models. Table 3 shows whether the existing defense strategy supports security aggregation.

2) PERFORMANCE EVALUATION

We can divide methods for detecting poisoning attacks based on evaluating model performance into two categories: evaluating local models [55], [57], [59], [62] and evaluating global models [56], [60], [63].

The approach to evaluating local models is to use a validation dataset to evaluate the local models uploaded by the clients. For example, Minghong et al. [55] removed the local model that significantly negatively impacts the accuracy and loss of the global model. Yuao et al. [57] updated the global model using only the local model that performed well on the test set and marked the client that uploaded a low-accuracy model as a malicious client. Furthermore, Mallah et al. [59] defend the poisoning attack by monitoring: 1) the convergence of the local model during training, 2) the angular distance of successive local model updates, and 3) removing local model updates from clients whose performance does not improve to defend against poisoning attacks. In addition, Yi et al. [62] proposed a method to automatically verify the validity of model updates through smart contracts in the blockchain, one of which is to test the performance of local models uploaded by users.

Moreover, we can also evaluate the updated global model to detect whether it contains malicious updates in this iteration. As a single global model is vulnerable to attacks, Xiaoyu et al. [63] propose a new defense strategy called FLCert. FLCert divides all clients into N groups and trains a global model for each group. Xiaoyu et al. propose two grouping strategies, FLCert-P, which uses random grouping, and FLCert-D, which uses hashing based on client ids. Finally, FLCert performs majority voting among the N global models to predict the labels of the test inputs. Although FLCert can resist poisoning attacks well, it cannot find the attacker.

Trust evaluation for the clients can be a good solution to this problem. Aashma et al. [60] proposed a novel framework that updates the global model after receiving each local model uploaded by a client and tests the performance of the updated model on a test set, updating the client's trust value based on the test results. The methods mentioned above for testing global/local models require the server to collect a test set, and the quality of this test set dramatically affects the effectiveness of the defense.

To solve this problem, Sebastien et al. [56] propose a new approach. They send the updated global model to the validation clients instead of testing it directly on the server. And these validation clients evaluate the global model using their local data. The server will remove this update when most validation clients vote that the global model is poisoned.

B. BYZANTINE ROBUST AGGREGATION

BRA is a passive defense strategy that mitigates the damage of poisoning attacks by changing the aggregation method of the global model. We can roughly classify BRA into three classes: 1) gradient clipping [29], [65], 2) removing extreme values [51], [64], [67], [68], [85], [86], and 3) differential privacy [29], [66].

1) GRADIENT CLIPPING

In a typical gradient clipping approach, we can set a threshold for the model parameters and discard all parameters that exceed the threshold [29], [65]. For example, Ziteng et al. [29] argue that boosted attacks are likely to produce updates with large norms. Therefore the aggregator can update the model in the following way to eliminate the effect of malicious gradients:

$$\boldsymbol{w}_0^{k+1} = \boldsymbol{w}_0^k + \frac{\eta}{n} \sum_{m \in M_k} \frac{\Delta \boldsymbol{w}_m^k}{max(1, ||\Delta \boldsymbol{w}_m^k||_2/T)} \tag{7}$$

where the threshold T can sometimes be chosen randomly. Similarly, Ashwinee et al. [65] clipped the gradient using the following way:

$$\Delta \boldsymbol{w}_m^k = \Delta \boldsymbol{w}_m^k \cdot min(1, \frac{L}{|\Delta \boldsymbol{w}_m^k|_2}) \tag{8}$$

where L is the l_2 clipping threshold.

2) REMOVE EXTREME VALUES

We can also remove extreme values from the model parameters to defend against the poisoning attack. For example, *Krum* [86] uses the one model with the smallest euclidean distance to all other models to update the global model, and the other models are discarded. In *trimmed mean* and *median*, for each parameter p in the model, the server uses the median value of each local model to update the global model [85]. In *trimmed mean*, for each parameter p, the largest b parameters and the smallest b parameters are removed from all m local models, and the mean of the remaining $m - 2b$ parameters is used to update the global model. In *median*, for each parameter p, the median of all m local models (if m is even, the mean of its middle two parameters) is used to update the global model.

3) DIFFERENTIAL PRIVACY

Besides, differential privacy was initially used for privacy protection in FL [7]. However, some researchers have also explored the application of differential privacy in poisoning attack protection [29], [66]. For example, Ziteng et al. [29] found that adding a small amount of Gaussian noise can further mitigate the attacker's impact. Further, Mohammad et al. [66] investigated whether and how differential privacy can defend against attacks against robustness and privacy in FL. Specifically, they experimentally evaluated the feasibility and effectiveness of local/central differential privacy (LDP/CDP) for poisoning attack defense for the first time. The experimental results show that both LDP and CDP are effective against poisoning attacks, and CDP is more effective than LDP against poisoning attacks. Interestingly, at the same time, some attackers exploit the differential privacy mechanism in FL to perform stealthy and persistent model poisoning attacks by adding specific fake noise [34].

C. VERIFICATION-BASED METHOD

In addition to the methods mentioned above, there are still many methods for poisoning attack defense. For example, Yu et al. [69] proposed a trusted execution environment (TEE)-based FL training protocol that can use digital signature techniques to verify that the gradient uploaded by the client is computed by standard SGD (or other optimization methods). The drawback of this approach is that it cannot cope with attackers' training models using fake or tampered data (data poisoning attacks). References [71] and [72] trains a GAN server-side to reconstruct the prototypical samples of clients' training data for auditing the accuracy of each client's model. However, this approach may compromise the user's privacy. The method proposed by Ronald et al. [73] is to train an SVM on the server to distinguish benign data from malicious data and then classify the local data on the client by facilitator (can be considered as a trusted third party). The biggest problem with this approach is that it requires thorough training of an SVM on poisoned and non-poisoned datasets that are relatively similar to the client's data source and thus may not apply to scenarios where the data is Non-IID. In addition, Jingjing et al. [70] devised a proof generation method called ADFL for users to generate proofs to verify whether it is malicious or not. The server sends a challenge to all clients at each iteration and determines whether the user is malicious by the client's response to the challenge. This approach also cannot cope with data poisoning attacks.

D. SUMMARY

We list some representative defense strategies against poisoning attacks in Table 3. We classify the existing defense strategies into three categories of models and show in the table the data distribution assumptions of the existing literature and whether they support security aggregation. Based on these data, we can draw the following conclusions.

1) MODEL ANALYSIS, BRA AND VERIFICATION-BASED METHOD

Poisoning attacks are executed through malicious models, so an intuitive defense strategy is to analyze the model. Typically, the server can analyze either a local model uploaded by the client or an aggregated global model. The biggest challenge of the former is that the server cannot access individual client models in a secure aggregation setting. However, some researchers have proposed methods to analyze models in encrypted mode [38], [49], [51]. The biggest problem with the latter is that current research is based on the predictive accuracy of the global model on the validation dataset to determine whether an attack has been poisoned, and targeted poisoning attacks may evade this defense.

BRA can be divided into gradient clipping, removing extreme values, and differential privacy. First, gradient clipping is used to defend against poisoning attacks by limiting the norm of parameters of gradients uploaded by the client. This approach prevents attackers from scaling up malicious gradients and swamping benign updates. Nevertheless, this strategy cannot work well in the non-IID setting.

Second, removing the extreme values discards the boundary values in a batch of updates. This approach also prevents the attacker from amplifying the malicious gradients, but it is more like a median aggregation than an average aggregation.

Finally, differential privacy improves robustness with a theoretical guarantee by clipping the gradient norm and injecting perturbations to the gradients. Differential privacy is a low-cost, privacy-preserving defense strategy against poisoning attacks. The reason that limits differential privacy's widespread use is that it reduces the accuracy of the global model.

The verification-based defense strategy prevents attackers from forging data/models by adding a verification step. This makes it more difficult for an attacker to execute an attack, but this restriction is possible to bypass.

At the same time, data distribution significantly impacts the defense against poisoning attacks. No matter which defense strategy is used, non-IID data will make it more difficult for servers to defend against poisoning attacks. This is mainly because, in the non-IID setting, the local models trained by individual clients are inherently different, which makes it difficult for the server to distinguish malicious updates from benign ones. We note that much work has been done on this problem, but more work is needed.

2) SECURE AGGREGATION SUPPORT

One of the most remarkable features of federated learning is that it protects the data privacy of the clients involved in training. However, many efforts to defend against poisoning attacks will likely lead to client data privacy leakage. While these efforts follow the fundamental principle that the client's original data is not released to the public, the intermediate data in federated learning may cause privacy leaks. Many improved federated learning aggregation strategies have been proposed to protect privacy in federated learning. We call these improved aggregation policies as secure aggregation policies in this paper.

The security aggregation mechanism in federated learning is not the focus of this paper, we only focus on its restriction of server access to the client's local model. Based on this, we counted the existing defense policies' support for security aggregation, and the results are shown in Table 3.

Among the model analysis methods, we found that in addition to the direct analysis of the global model that supports secure aggregation, the analysis of the encrypted local model can defend against poisoning attacks and protect clients' data privacy. For example, after encrypting a local model using homomorphic encryption, information such as the similarity of two models can be computed by performing operations on the ciphertext to detect malicious models [26].

In BRA, gradient clipping cannot support secure aggregation because it requires operations on individual gradients and can only be done by the server. Access to the client's local model is also usually required in the extreme value removal strategy. However, some researchers [68] have proposed an iterative median approximation method that can approximate the median of all models without accessing the local model and thus can also support the secure aggregation strategy. Finally, differential privacy is a secure aggregation strategy that can also resist poisoning attacks, but its biggest problem is that it reduces the accuracy of the global model. Moreover, Yuzhe et al. [87] show that differential privacy is adequate when the adversary can only poison a small number of samples. However, as the adversary poisons more data, this protection degrades.

In a verification-based approach, the support for secure aggregation depends on its verification policy. For example, verifying the integrity of the training process can prevent an attacker from injecting false training results. This process does not require access to the local model and thus can support secure aggregation. In addition, directly verifying the client's local data is not always allowed, but introducing a trusted third party can solve this problem. Although trusted third parties are difficult to guarantee in real-world settings, this approach also provides lessons for subsequent research.

VI. CHALLENGES AND FUTURE DIRECTIONS

Research on implementing poisoning attacks and defending against poisoning attacks has made significant progress in recent years, but some issues still need further research. We summarize the main past work and build on it to explore promising future research directions.

A. POISONING ATTACK

For poisoning attacks, the attacker is no longer satisfied with simply breaking the convergence of the global model. For attackers, it becomes a new goal to improve the attack's stealthiness, efficiency, and persistence.

1) STEALTHINESS, EFFICIENCY, AND PERSISTENCE OF POISONING ATTACK

The stealthiness of poisoning attacks can be improved in two ways. First, the attacker can learn the characteristics of benign clients, such as controlling the Euclidean distance between the malicious and benign models [28] or controlling the poisoning data rate (PDR) [80] to reduce the risk of detection by the server. However, these methods often reduce the attack's efficiency, requiring the attacker to make a trade-off between stealth and efficiency.

Second, attackers can improve the design of triggers. For example, the attacker can decompose a global trigger pattern into separate local patterns and embed them into the training set of different adversarial parties respectively [31]. This way, the stealthiness of the poisoning attack can be effectively improved because individual local triggers are more difficult to detect than global ones.

Moreover, neural networks have catastrophic forgetting properties, where neural networks may quickly forget previously acquired concepts. Florian et al. [88] pointed out that an attacker with sufficiently long continuous learning cycles can achieve high accuracy on backdoor samples. However, the attacker faces the next two difficulties in realistic scenarios. First, each training round of FL selects some clients randomly. The attacker cannot guarantee that the clients under his control will be selected to participate in the training every time, especially if many clients are participating in the training. Second, frequent attacks increase the risk of exposure. Therefore, a possible future research direction is the study of the persistence of poisoning attacks.

2) POISONING ATTACKS COMBINED WITH PRIVACY INFERENCE ATTACKS

In privacy inference attacks, attackers use local/global models to infer the data privacy of other clients. Exploiting the data privacy of other clients can help attackers improve the efficiency of poisoning attacks. For example, an attacker can use GAN to reconstruct other clients' datasets so that the attacker can execute the attack by generating fake data, even if the attacker lacks a specific class of sample data [26].

Furthermore, in the approach proposed by Henger et al. [32], the malicious client first uses a gradient inversion-based inference attack to learn approximations of other client data distributions from model updates received by the server. Then, the attacker uses his local data and the learned distributions to construct a simulator of the FL environment. Finally, the attacker uses reinforcement learning to learn an adaptive attack strategy.

Therefore, one future research direction is to combine the latest research on privacy inference attacks with poisoning attacks to design a stronger attack framework.

B. STRATEGIES FOR DEFENDING AGAINST POISONING ATTACKS

1) MORE POWERFUL ADVERSARIES

Most current defense strategies limit the attacker's capabilities. For example, many studies assume that the attacker only uses label-flipping attacks. However, the existing attacks are constantly enriched with increasingly sophisticated attack methods being proposed. Attackers have used methods such as reinforcement learning to improve the efficiency of their attacks. Most existing defense strategies are static, and cannot be dynamically adjusted to the attackers' attack methods. At the same time, many researchers assume that the adversary in their experiments undisguisedly performs the attack. Thus, they are vulnerable to more sophisticated poisoning attacks, despite the positive results they have achieved in their experiments.

Another problem with existing defense strategies is that they are primarily targeted at specific attacks. They can only defend against known attack methods. Once an attacker knows these defense strategies exist, it is easy to bypass them. In general, defenses against poisoning attacks are at a relative disadvantage. Improving the effectiveness of countermeasures should start with releasing assumptions about the adversary, such as the assumption that the attacker can perform more sophisticated and stealthy attacks.

2) PRIVACY PROTECTION

Many existing studies assume the server has access to the client's local model, which poses a significant risk of privacy leakage. Worse, these defense strategies cannot be directly integrated with existing privacy protection strategies. Therefore, it is necessary to study defense strategies for poisoning attacks that support secure aggregation.

Fortunately, more and more researchers have realized this problem. One existing trend is to design defense strategies against poisoning attacks in the cryptographic mode with existing privacy-preserving cryptographic algorithms. However, there are relatively few studies in this direction, mainly focusing on homomorphic encryption and differential privacy. Therefore, future research can combine other encryption algorithms commonly used for privacy protection (e.g., function encryption, secret sharing, etc.) to research defense strategies against poisoning attacks.

VII. SUMMARY

This survey aims to provide a comprehensive and up-to-date overview of poisoning attacks and countermeasures in federated learning. We investigate existing poisoning attacks and defense strategies against poisoning attacks from the perspective of privacy protection. Moreover, we classify poisoning attacks from two aspects: attack methods and the attacker's intention. Based on this, we analyze the differences and connections between the different categories of poisoning attacks. Our observations show that poisoning attacks are becoming more effective, stealthy, and powerful. On the other hand, the defense against poisoning attacks, although well studied, still has many issues to be addressed.

We have pointed out potential research directions for poisoning attacks. Attackers have already solved the feasibility problem. As a result, they may be more concerned with the trade-off between efficiency and stealth. In poisoning attacks, multiple attacks may be combined (e.g., with privacy inference attacks). At the same time, the challenge for defenders is much tougher, as effectively preventing or detecting poisoning attacks remains an open problem. Moreover, defending against poisoning attacks while protecting privacy is the focus of future research.

REFERENCES

[1] A. Esteva, K. Chou, S. Yeung, N. Naik, A. Madani, A. Mottaghi, Y. Liu, E. Topol, J. Dean, and R. Socher, "Deep learning-enabled medical computer vision," *NPJ Digit. Med.*, vol. 4, no. 1, pp. 1–9, Jan. 2021.

[2] T. Brown et al., "Language models are few-shot learners," 2020, *arXiv:2005.14165*.

[3] I. Portugal, P. Alencar, and D. Cowan, "The use of machine learning algorithms in recommender systems: A systematic review," *Expert Syst. Appl.*, vol. 97, pp. 205–227, May 2018.

[4] A. Act, "Health insurance portability and accountability act of 1996," *Public Law*, vol. 104, p. 191, Aug. 1996.

[5] P. Voigt and A. V. D. Bussche, *The EU General Data Protection Regulation (GDPR): A Practical Guide*, 1st ed. Cham, Switzerland: Springer, 2017.

[6] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Conf. Artif. Intell. Statist.*, vol. 54, Apr. 2017, pp. 1273–1282.

[7] P. Kairouz, "Advances and open problems in federated learning," 2019, *arXiv:1912.04977*.

[8] N. Bouacida and P. Mohapatra, "Vulnerabilities in federated learning," *IEEE Access*, vol. 9, pp. 63229–63249, 2021.

[9] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, 3rd Quart., 2020.

[10] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," 2020, *arXiv:2003.02133*.

[11] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Gener. Comput. Syst.*, vol. 115, pp. 619–640, Feb. 2021.

[12] Z. Wang, Q. Kang, X. Zhang, and Q. Hu, "Defense strategies toward model poisoning attacks in federated learning: A survey," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2022, pp. 548–553.

[13] P. M. Mammen, "Federated learning: Opportunities and challenges," 2021, *arXiv:2101.05428*.

[14] J. Liu, J. Huang, Y. Zhou, X. Li, S. Ji, H. Xiong, and D. Dou, "From distributed machine learning to federated learning: A survey," *Knowl. Inf. Syst.*, vol. 64, no. 4, pp. 885–917, Apr. 2022.

[15] L. Lyu, H. Yu, X. Ma, C. Chen, L. Sun, J. Zhao, Q. Yang, and P. S. Yu, "Privacy and robustness in federated learning: Attacks and defenses," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Nov. 10, 2022, doi: 10.1109/TNNLS.2022.3216981.

[16] C. Fu, X. Zhang, S. Ji, J. Chen, J. Wu, S. Guo, J. Zhou, A. X. Liu, and T. Wang, "Label inference attacks against vertical federated learning," in *Proc. 31st USENIX Secur. Symp.* Boston, MA, USA, Aug. 2022, pp. 1–15.

[17] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients—How easy is it to break privacy in federated learning?" in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, Sep. 2020, pp. 16937–16947.

[18] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. D. Alché-Buc, E. Fox, and R. Garnett, Eds., 2019, pp. 14774–14784.

[19] B. Zhao, K. R. Mopuri, and H. Bilen, "IDLG: Improved deep leakage from gradients," 2020, *arXiv:2001.02610*.

[20] Z. Ying, Y. Zhang, and X. Liu, "Privacy-preserving in defending against membership inference attacks," in *Proc. Workshop Privacy-Preserving Mach. Learn. Pract.*, Nov. 2020, pp. 61–63.

[21] Z. Li, V. Sharma, and S. P. Mohanty, "Preserving data privacy via federated learning: Challenges and solutions," *IEEE Consum. Electron. Mag.*, vol. 9, no. 3, pp. 8–16, Apr. 2020.

[22] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," 2012, *arXiv:1206.6389*.

[23] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage, "Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning," 2021, *arXiv:2108.10241*.

[24] Y. Sun, H. Ochiai, and J. Sakuma, "Semi-targeted model poisoning attack on federated learning via backward error analysis," in *Proc. Int. Joint Conf. Neur. Netw. (IJCNN)*, 2022, pp. 1–8.

[25] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! Targeted clean-label poisoning attacks on neural networks," in *Proc. 32nd Int. Neural Inf. Process. Syst.*, 2018, pp. 6106–6116.

[26] 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, Mar. 2021.

[27] D. Rong, S. Ye, R. Zhao, H. N. Yuen, J. Chen, and Q. He, "FedRecAttack: Model poisoning attack to federated recommendation," in *Proc. IEEE 38th Int. Conf. Data Eng. (ICDE)*, May 2022, pp. 2643–2655.

[28] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, Jun. 2019, pp. 634–643.

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

[30] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, vol. 108, Aug. 2020, pp. 2938–2948.

[31] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "DBA: Distributed backdoor attacks against federated learning," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–9.

[32] H. Li, X. Sun, and Z. Zheng, "Learning to attack federated learning: A model-based reinforcement learning attack framework," in *Proc. Adv. Neural Inf. Process. Syst.*, Oct. 2022, pp. 1–15. [Online]. Available: https://openreview.net/forum?id=4OHRr7gmhd4

[33] X. Zhou, M. Xu, Y. Wu, and N. Zheng, "Deep model poisoning attack on federated learning," *Future Internet*, vol. 13, no. 3, p. 73, Mar. 2021.

[34] M. T. Hossain, S. Islam, S. Badsha, and H. Shen, "DeSMP: Differential privacy-exploited stealthy model poisoning attacks in federated learning," in *Proc. 17th Int. Conf. Mobility, Sens. Netw. (MSN)*, Dec. 2021, pp. 167–174.

[35] X. Cao and N. Z. Gong, "MPAF: Model poisoning attacks to federated learning based on fake clients," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2022, pp. 3396–3404.

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

[37] Y. Mao, X. Yuan, X. Zhao, and S. Zhong, "Romoa: Robust model aggregation for the resistance of federated learning to model poisoning attacks," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, Sep. 2021, pp. 476–496.

[38] 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 Security*, vol. 17, pp. 1639–1654, 2022.

[39] V. Shejwalkar and A. Houmansadr, "Manipulating the Byzantine: Optimizing model poisoning attacks and defenses for federated learning," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2021, pp. 1–19.

[40] X. Cao, M. Fang, J. Liu, and N. Zhenqiang Gong, "FLTrust: Byzantine-robust federated learning via trust bootstrapping," 2020, *arXiv:2012.13995*.

[41] 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. Discovery Data Mining*, New York, NY, USA, 2022, pp. 2545–2555, doi: 10.1145/3534678.3539231.

[42] S. Awan, B. Luo, and F. Li, "CONTRA: Defending against poisoning attacks in federated learning," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, Sep. 2021, pp. 455–475.

[43] W. Liu, H. Lin, X. Wang, J. Hu, G. Kaddoum, M. J. Piran, and A. Alamri, "D2MIF: A malicious model detection mechanism for federated learning empowered artificial intelligence of things," *IEEE Internet Things J.*, early access, May 18, 2021, doi: 10.1109/JIOT.2021.3081606.

[44] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Proc. Eur. Symp. Comput. Secur.* Cham, Switzerland: Springer, Sep. 2020, pp. 480–501.

[45] J. Zhang, D. Wu, C. Liu, and B. Chen, "Defending poisoning attacks in federated learning via adversarial training method," in *Proc. Int. Conf. Frontiers Cyber Secur.* Singapore: Springer, Nov. 2020, pp. 83–94.

[46] D. Li, W. E. Wong, W. Wang, Y. Yao, and M. Chau, "Detection and mitigation of label-flipping attacks in federated learning systems with KPCA and K-means," in *Proc. 8th Int. Conf. Dependable Syst. Appl. (DSA)*, Aug. 2021, pp. 551–559.

[47] X. Li, Z. Qu, S. Zhao, B. Tang, Z. Lu, and Y. Liu, "LoMar: A local defense against poisoning attack on federated learning," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 1, pp. 437–450, Jan. 2023.

[48] K.-H. Chow and L. Liu, "Perception poisoning attacks in federated learning," in *Proc. 3rd IEEE Int. Conf. Trust, Privacy Secur. Intell. Syst. Appl. (TPS-ISA)*, Dec. 2021, pp. 146–155.

[49] Z. Ma, J. Ma, Y. Miao, X. Liu, K.-K.-R. Choo, and R. H. Deng, "Pocket diagnosis: Secure federated learning against poisoning attack in the cloud," *IEEE Trans. Services Comput.*, vol. 15, no. 6, pp. 3429–3442, Nov. 2022.

[50] 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, Sep. 2022.

[51] X. Liu, H. Li, G. Xu, Z. Chen, X. Huang, and R. Lu, "Privacy-enhanced federated learning against poisoning adversaries," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 4574–4588, 2021.

[52] D. Cao, S. Chang, Z. Lin, G. Liu, and D. Sun, "Understanding distributed poisoning attack in federated learning," in *Proc. IEEE 25th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2019, pp. 233–239.

[53] C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," 2018, *arXiv:1808.04866*.

[54] S. Shi, C. Hu, D. Wang, Y. Zhu, and Z. Han, "Federated anomaly analytics for local model poisoning attack," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 2, pp. 596–610, Feb. 2022.

[55] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to Byzantine-robust federated learning," in *Proc. USENIX Secur.*, Aug. 2020, pp. 1605–1622.

[56] S. Andreina, G. A. Marson, H. Mollering, and G. Karame, "BaFFLe: Backdoor detection via feedback-based federated learning," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2021, pp. 852–863.

[57] Y. Wang, T. Zhu, W. Chang, S. Shen, and W. Ren, "Model poisoning defense on federated learning: A validation based approach," in *Proc. Int. Conf. Netw. Sys. Secur.* Cham, Switzerland: Springer, Dec. 2020, pp. 207–223.

[58] J. Tan, Y. C. Liang, N. C. Luong, and D. Niyato, "Toward smart security enhancement of federated learning networks," *IEEE Netw.*, vol. 35, no. 1, pp. 340–347, Jan./Feb. 2020.

[59] R. A. Mallah, D. Lopez, G. B. Marfo, and B. Farooq, "Untargeted poisoning attack detection in federated learning via behavior attestation," 2021, *arXiv:2101.10904*.

[60] A. Uprety and D. B. Rawat, "Mitigating poisoning attack in federated learning," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2021, pp. 1–7.

[61] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *IEEE Wireless Commun.*, vol. 27, no. 2, pp. 72–80, Feb. 2020.

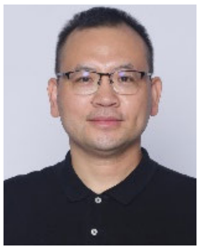
[62] Y. Liu, J. Peng, J. Kang, A. M. Iliyasu, D. Niyato, and A. A. A. El-Latif, "A secure federated learning framework for 5G networks," *IEEE Wireless Commun.*, vol. 27, no. 4, pp. 24–31, Aug. 2020.

[63] X. Cao, Z. Zhang, J. Jia, and N. Z. Gong, "FLCert: Provably secure federated learning against poisoning attacks," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 3691–3705, 2022.

[64] M. Shayan, C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Biscotti: A ledger for private and secure peer-to-peer machine learning," 2018, *arXiv:1811.09904*.

[65] A. Panda, S. Mahloujifar, A. N. Bhagoji, S. Chakraborty, and P. Mittal, "SparseFed: Mitigating model poisoning attacks in federated learning with sparsification," in *Proc. 25th Int. Conf. Artif. Intell. Statist.*, 2022, pp. 7587–7624.

[66] M. Naseri, J. Hayes, and E. De Cristofaro, "Local and central differential privacy for robustness and privacy in federated learning," 2020, *arXiv:2009.03561*.
[67] R. Guerraoui and S. Rouault, "The hidden vulnerability of distributed learning in Byzantium," in *Proc. 35th Int. Conf. Mach. Learn.*, Jul. 2018, pp. 3521–3530.
[68] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," 2019, *arXiv:1912.13445*.
[69] Y. Chen, F. Luo, T. Li, T. Xiang, Z. Liu, and J. Li, "A training-integrity privacy-preserving federated learning scheme with trusted execution environment," *Inf. Sci.*, vol. 522, pp. 69–79, Jun. 2020.
[70] J. Guo, H. Li, F. Huang, Z. Liu, Y. Peng, X. Li, J. Ma, V. G. Menon, and K. K. Igorevich, "ADFL: A poisoning attack defense framework for horizontal federated learning," *IEEE Trans. Ind. Informat.*, vol. 18, no. 10, pp. 6526–6536, Oct. 2022.
[71] Y. Zhao, J. Chen, J. Zhang, D. Wu, J. Teng, and S. Yu, "PDGAN: A novel poisoning defense method in federated learning using generative adversarial network," in *Proc. Int. Conf. Algorithms Archit. Parallel Process.* Cham, Switzerland: Springer, Jun. 2019, pp. 595–609.
[72] Y. Zhao, J. Chen, J. Zhang, D. Wu, M. Blumenstein, and S. Yu, "Detecting and mitigating poisoning attacks in federated learning using generative adversarial networks," *Concurrency Comput., Pract. Exper.*, vol. 34, no. 7, p. e5906, Mar. 2022.
[73] R. Doku and D. B. Rawat, "Mitigating data poisoning attacks on a federated learning-edge computing network," in *Proc. IEEE 18th Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2021, pp. 1–6.
[74] R. Gosselin, L. Vieu, F. Loukil, and A. Benoit, "Privacy and security in federated learning: A survey," *Appl. Sci.*, vol. 12, no. 19, p. 9901, Oct. 2022.
[75] Z. Chai, A. Ali, S. Zawad, S. Truex, A. Anwar, N. Baracaldo, Y. Zhou, H. Ludwig, F. Yan, and Y. Cheng, "TiFL: A tier-based federated learning system," in *Proc. 29th Int. Symp. High-Perform. Parallel Distrib. Comput.*, Jun. 2020, pp. 125–136.
[76] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient federated learning via guided participant selection," 2020, *arXiv:2010.06081*.
[77] G. Sun, Y. Cong, J. Dong, Q. Wang, L. Lyu, and J. Liu, "Data poisoning attacks on federated machine learning," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 11365–11375, Nov. 2021.
[78] A. Manna, H. Kasyap, and S. Tripathy, "*Moat*: Model agnostic defense against targeted poisoning attacks in federated learning," in *Proc. Int. Conf. Inf. Commun. Secur.* Cham, Switzerland: Springer, 2021, pp. 38–55.
[79] S. A. Rahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani, "A survey on federated learning: The journey from centralized to distributed on-site learning and beyond," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5476–5497, Apr. 2020.
[80] 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.*, 2020, pp. 1–7.
[81] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, p. 1175.
[82] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning," in *Proc. Annu. Tech. Conf.*, Jul. 2020, pp. 493–506.
[83] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers, "Protection against reconstruction and its applications in private federated learning," 2018, *arXiv:1812.00984*.
[84] Z. Chuanxin, S. Yi, and W. Degang, "Federated learning with Gaussian differential privacy," in *Proc. 2nd Int. Conf. Robot., Intell. Control Artif. Intell.*, Oct. 2020, p. 296.
[85] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine robust distributed learning: Towards optimal statistical rates," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, vol. 80, Jul. 2018, pp. 5650–5659.
[86] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, Dec. 2017, pp. 1–15.
[87] Y. Ma, X. Zhu, and J. Hsu, "Data poisoning against differentially-private learners: Attacks and defenses," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 4732–4738.
[88] F. Nuding and R. Mayer, "Data poisoning in sequential and parallel federated learning," in *Proc. ACM Int. Workshop Secur. Privacy Anal.*, Apr. 2022, pp. 24–34.



GEMING XIA received the B.S., M.S., and Ph.D. degrees in computing science from the National University of Defense Technology (NUDT), Changsha, China, in 1994, 2000, and 2008, respectively. He is currently a Full Professor with the College of Computers, NUDT. His research interests include edge computing, the IoT, computing power networks, and artificial intelligence.

JIAN CHEN received the B.S. degree in the IoT engineering from the Wuhan University of Technology, Wuhan, China, in 2020. He is currently pursuing the M.S. degree in electronic information with the College of Computer Science and Technology, National University of Defense Technology, Changsha, China. His research interests include edge computing, federated learning, and swarm intelligence.

CHAODONG YU received the B.S. degree in communication engineering from the National University of Defense Technology, Changsha, China, in 2020, where he is currently pursuing the M.S. degree in electronic information with the College of Computer Science and Technology. His research interests include edge computing, trust evaluation, and swarm intelligence.

JUN MA received the B.S. degree in electronic information science and technology from Hunan Normal University, in 2016. He is currently pursuing the M.S. degree in electronic information with the College of Computer Science and Technology, National University of Defense Technology, Changsha, China. His main research interests include edge computing, decentralized computing, the IoT, vehicle-mounted, and UAV self-organizing networks.