BADFL: Backdoor Attack Defense in Federated Learning From Local Model Perspective

Haiyan Zhang, Xinghua Li, *Member, IEEE*, Mengfan Xu, Ximeng Liu, *Senior Member, IEEE*, Tong Wu, Jian Weng, *Member, IEEE*, and Robert H. Deng, *Fellow, IEEE*

***Abstract*—There is substantial attention to federated learning with its ability to train a powerful global model collaboratively while protecting data privacy. Despite its many advantages, federated learning is vulnerable to backdoor attacks, where an adversary injects malicious weights into the global model, making the global model's targeted predictions incorrect. Existing defenses based on identifying and eliminating malicious weights ignore the similarity variation of the local weights during iterations in the malicious model detection and the presence of benign weights in the malicious model during the malicious local weight elimination, resulting in a poor defense and a degradation of global model accuracy. In this paper, we defend against backdoor attacks from the perspective of local models. First, a malicious model detection method based on interpretability techniques is proposed. The method appends a sampling check after clustering to identify malicious models accurately. We further design a malicious local weight elimination method based on local weight contributions. This method preserves the benign weights in the malicious model to maintain their contributions to the global model. Finally, we analyze the security of the proposed method in terms of model closeness and then verify the effectiveness of the proposed method through experiments. In comparison with existing defenses, the results show that BADFL improves the global model accuracy by 23.14% while reducing the attack success rate to 0.04% in the best case.**

***Index Terms*—Federated learning, backdoor attack, clustering, interpretability.**

I. Introduction

FEDERATED learning (FL) [1], [2], an emerging distributed machine learning framework, consists of local training and server aggregating in each iteration. Each client trains a local model and then submits local weights to the aggregation server, aggregating them to obtain a global model. Then, the aggregation server propagates the global model back to each client for the next iteration. FL allows multiple clients to train machine learning models collaboratively without uploading training data to the aggregation server and has become an infrastructure for building machine learning models in areas such as transportation, healthcare, and the Internet of Things [3], [4]. However, existing studies [5], [6], [7], [8] show that FL is vulnerable to backdoor attacks. An adversary manipulates the local models of a subset of clients joining FL, consequently injecting malicious local weights into the global model. In such an attack, the attacker aims to manipulate the global model so that the inputs with backdoors are predicted to the attacker's specified target.

Many defenses have been proposed [9], [10], [11], [12], [13] with the general idea of malicious model detection followed by further eliminating the impact of malicious local weights on the global model. During malicious model detection, existing defenses [2], [10], [12], [13], [14], [15], [16] attempt to identify malicious models by the low similarity between local weights, i.e., models incompatible with most local weights are considered as malicious. Therefore, the aggregation server first calculates the similarities between each local weight and other local weights. Then, it identifies local models whose weights are not similar to most local weights as malicious and removes them. The above detection methods cannot guarantee that none of the malicious local weights participates in the aggregation. Once the global model is contaminated, the embedded backdoor can work even if the subsequent iterations are not attacked [17]. Therefore, the existing defenses [15], [16], [18] add a malicious local weights elimination after detection. In malicious local weight elimination, the existing defenses modify the local weights associated with the backdoor task through perturbation and clipping techniques to reduce the probability of the backdoor being activated.

However, the existing defenses have two problems. First, the existing defenses consider local weights with low similarity as malicious in the malicious model detection process, ignoring the similarity variation of local weights during iterations. We find that the similarity between local weights decreases with the increase of iterations. As a result, the existing defenses cannot effectively filter malicious models and even identify clean models as malicious ones in iterations, resulting in a poor defense. Fig. 1 shows that existing defenses can erroneously remove many clean

Manuscript received 2 November 2023; revised 23 May 2024; accepted 22 June 2024. Date of publication 28 June 2024; date of current version 27 September 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62125205 and Grant U23A20303, in part by the '111 Center' under Grant B16037, the Key Research and Development Program of Shaanxi under Grant 2023KXJ-190, and in part by the Fundamental Research Funds for the Central Universities under Grant YJSJ24010. Recommended for acceptance by X. Yi. *(Corresponding author: Xinghua Li.)*

Haiyan Zhang, Xinghua Li, and Tong Wu are with the State Key Laboratory of Integrated Services Networks, and the School of Cyber Engineering, Xidian University, Xi'an 710071, China (e-mail: 21151110611@stu.xidian.edu.cn; xhli1@mail.xidian.edu.cn; tongwu562131@outlook.com).

Mengfan Xu is with the School of Computer Science, Shaanxi Normal University, Xi'an 710062, China (e-mail: cybersecurityxu@snnu.edu.cn).

Ximeng Liu is with the Key Laboratory of Information Security of Network Systems, Fuzhou University, Fuzhou 350108, China (e-mail: snbnix@gmail.com).

Jian Weng is with the College of Cyber Security, Jinan University, Guangzhou 510632, China (e-mail: cryptjweng@gmail.com).

Robert H. Deng is with the School of Information Systems, Singapore Management University, Singapore 178902 (e-mail: robertdeng@smu.edu.sg).

This article has supplementary downloadable material available at https://doi.org/10.1109/TKDE.2024.3420778, provided by the authors.

Digital Object Identifier 10.1109/TKDE.2024.3420778

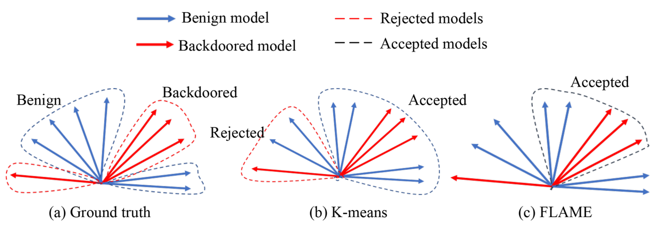

Fig. 1. Existing defenses use different clustering methods on a vector of a set of local weights. (a) It is an attack scenario involving two adversaries: one adversary manipulates one client, and the other can manipulate three clients. (b) It is the K-means method (as used in Auror [17]) which fails to separate benign and malicious models. (c) It shows the clustering of FLAME [15], which requires that N/2+1 of the clients are highly similar to the benign model. However, the similarity variations make this requirement impossible to guarantee. Hence, FLAME also unsuccessfully separates the benign from the malicious.

models, which reduces the global model's accuracy. Second, the existing defenses apply uniform clipping to local weights after detection to eliminate the impact of malicious weights, ignoring the presence of benign weights in the malicious model. The malicious model is not entirely trained with backdoor samples. Its training set also contains a high portion of clean samples [7]. The existing defenses using uniform clipping for malicious models cause the benign local weights in the malicious models to be clipped, leading to a degradation of the global model accuracy.

To overcome the limitations of existing defenses, we propose BADFL: **B**ackdoor **A**ttack **D**efense in **F**ederated **L**earning from local model perspective. Instead of relying on the similarity between local weights, we perform neuron inspection on an individual model to identify malicious models in the model detection process. In the malicious local weights elimination process, we maintain the benign weights in malicious models and clip the malicious weights. Our contributions are as follows.

- We present a malicious model detection method based on interpretability. We design the local model clustering algorithm and model neuron inspection algorithm, respectively, to cluster the local weights after outlier detection and to check the sampling of each cluster using interpretability. Our proposed method adds sampling checks after outlier detection and cluster, which can accurately identify malicious models.
- We further propose a malicious local weight elimination method based on local weight contributions. We design a local weight contribution function and a malicious local weight clipping algorithm that adaptively clips malicious local weights based on local weight contributions. The method preserves the clean local weights in malicious models so that they can continue to contribute to the global model and improve the global model accuracy.
- We analyze the security of the proposed method in terms of model closeness and then verify the effectiveness of the proposed method through experiments. In comparison with existing defenses, the results show that BADFL improves the global model accuracy by 23.14% while reducing the attack success rate to 0.04% in the best case. Meanwhile, BADFL is robust to parameters such as poisoning rate and the number of attackers.

TABLE I
SUMMARY OF EXISTING BACKDOOR ATTACK DEFENSE METHODS IN FL

Scheme	Benign paramet -er preservation	Single in -spection	Layer clipping	non-IID
Scheme [22]	✗	✗	✗	✗
Scheme [14]	✗	✗	✗	✗
Scheme [23]	✗	✗	✗	✓
Scheme [12]	✗	✗	✗	✗
Scheme [24]	✗	✗	✗	✓
Scheme [9]	✓	✓	✗	✓
Scheme [18]	✓	✗	✗	✓
Scheme [15]	✗	✗	✗	✓
Ours	✓	✓	✓	✓

Note: ✓: Contentment; ✗: Discontentment.

The rest of this paper is organized as follows. In Section II, we discuss the related work. In Section III, we introduce some preliminaries. We state the problem in Section IV. In Section V, BADFL is presented in detail. Then, we analyze the security of BADFL in Section VI. The experiment analysis is given in Section VII. Finally, we conclude this paper in Section VIII.

II. Related Work

Many defenses against FL backdoor attacks have been proposed [9], [10], [11], [12], [13] with the general idea of malicious model detection followed by further eliminating the impact of malicious local weights on the global model. During malicious model detection, PEFL [12] applies the Pearson Coefficient to measure the similarity. FLAME [15] applies cosine similarity to measure the similarity and then uses HDBSCAN's idea to identify malicious models. All the above defenses are designed with different detection schemes based on the criterion of low similarity of malicious models with other models. However, they ignore that the similarity between local weights varies from iteration to iteration. Therefore, it is not reasonable to rely solely on the low similarity criterion to identify malicious models.

Based on the existing defenses, we add random sampling after clustering and further employ neuron checking to identify malicious models. Existing defenses [2], [18], [19], [20], [21] focus on reducing the impacts of malicious local weights that escape during the malicious model detection on the global model by using clipping and perturbation. CRFL [18] and FLAME [15] are uniformly clipped to reduce the impacts of malicious weights on the global model. However, they ignore that the local local weights have different contributions to the global model. Thus the strategy of uniform clipping affects the contribution of benign weights to the global model. Based on the existing work, we design the local weight contribution function to dynamically adjust the weight of different local weights to the global model and further reduce the impacts of malicious weights on the global model.

Finally, we summarize the existing backdoor attack defense schemes in terms of different scenarios and requirements, as shown in the following Table I.

III. Preliminaries

A. Local Outliers Factor Anomaly Detection

The local outliers factor (LOF) is an unsupervised anomaly detection algorithm that achieves anomaly detection by calculating the local density deviation of a given data point concerning its neighborhood. The LOF algorithm determines whether each data point p is an anomaly by comparing the density of that point with that of its neighbors. A low-density point p is more likely to be considered an anomaly. The distance between points can be used to estimate the local density. In general, the farther the distance between points, the lower the local density, as there are fewer points nearby. Conversely, when points are close to each other, the local density is higher because there are more data points in their vicinity. Therefore, the point density in the LOF algorithm is calculated by the k-neighborhood of points, not by comparing all points. It provides anomaly calculation for sets with drastically different density dispersions. We regard the similarity between models as a property of the models. However, the similarity between models can vary due to the training of local models on different datasets. Therefore, we use the LOF algorithm to divide the local models into clusters based on the reachable density distribution of the local models.

B. Interpretability Technique

The interpretability technique is critical to understanding how Deep Neural Networks (DNN) make decisions and interpreting the output of a DNN based on its inputs. In the domain of computer vision, interpretability describes the visualization of a DNN's representation. Deconvolution of DNN feature representations can provide a solution for the interpretation of the DNN output. A gradient-based saliency map is an interpretability technique that can calculate the gradient of the output prediction labels compared to the DNN input for evaluating the significance of features. With a given image x and an output class y, the output prediction for the class can be expressed as $f_y = (w; x)$. The output can be estimated as a linear function and denoted as:

$$f_y = (w; x) \approx w_y * x + b_y. \tag{1}$$

Consequently, we can calculate the gradient of the output category relativized to the input image as:

$$g = \left.\frac{\partial f_y(w; x)}{\partial x}\right|_x. \tag{2}$$

This formula describes the gradient g of a function f at input x with respect to parameter w. Here, $f_y(w; x)$ represents the output of the function f, where y typically denotes the category. $\frac{\partial}{\partial x}$ denotes the partial derivative with respect to x, and $|_x$ indicates the value at x. This gradient reveals it is how the single pixel of the input image has an impact on the output prediction of a single class.

TABLE II
SYMBOLS AND NOTATIONS USED IN THE SCHEME

Notation	Description
$A(\cdot)$	Aggregation algorithm
z_j^i	j-th data sample on i-th client
z'^{i}_{j}	j-th data poisoned sample on i-th client
δ_j	The backdoor added by the attacker on the j-th sample
D	The union of clean dataset of N clients
D'	Total samples poisoned by R attackers on a clean dataset
$g_i(w)$	The local gradients of client i with clean dataset
$g'_i(w)$	The local gradients of client i with backdoor dataset
w_s^i	The local model weights of client i at local round s
w_t	t-th global model weights
h_s	Smooth classication
E	Local training rounds
$\frac{q_i}{bn_i}$	The poisoning ratio of the i-th client
ξ_s^k	A random small batch sample in local training

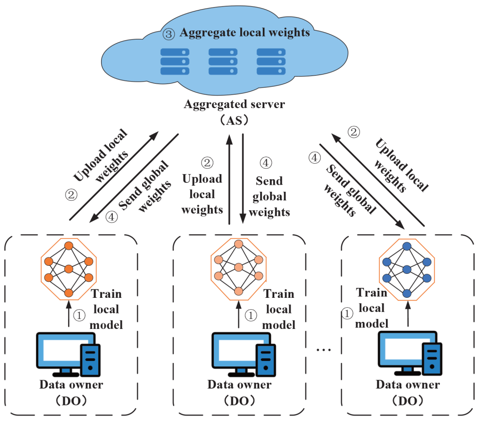

Fig. 2. System model.

IV. Problem Formulation

A. Notation

To facilitate the description, we first define some symbols and give the corresponding description in Table II.

B. System Model

As depicted in Fig. 2, there are two basic entities in our system:

- *Data owners (DOs):* All data owners, also known as clients, collaboratively train a global model under the coordination of an aggregation server. For privacy reasons, each client trains a local model (see ①) on a local dataset and then uploads the local weights to the aggregation server (see ②). In addition, we believe that the distributions of data held by DOs can be different.
- *Aggregation Server (AS): AS* is responsible for receiving all the local weights and aggregating them to obtain an optimized global model (see ③). Then, the AS sends a global model to DOs (see ④). It also needs to detect backdoor attacks launched by potential malicious DOs.

We adopt FedAvg [25], which is the typical FL aggregation method. Formally, assuming the model weights $w \in \mathbb{R}^d$, the clients attempt to minimize the mean of their loss functions.

$$\min_{w\in\mathbb{R}^d} F(w) = \sum_{k=1}^{N} p^{(k)} F^{(k)}(w), \tag{3}$$

where $F^{(k)}$ and $p^{(k)}$ are the loss function and aggregation weights of the i-th client, respectively. The N is the number of clients. Assuming that the i-th client has n_k training data in the local dataset $S^{(k)} = \{z_1^{(k)}, z_2^{(k)}, \ldots, z_{n_k}^{(k)}\}$, let the local model $F^{(k)}(\cdot)$ be defined as a typical loss function under empirical risk minimization, such as cross-entropy.

$$F^{(k)}(w) = \frac{1}{n_k}\sum_{j=1}^{n_k} L\left(z_j^{(k)}; w\right), \tag{4}$$

where $L(\cdot;\cdot)$ is defined as the loss function.

Specifically, the FedAvg protocol is implemented as follows: in round t, the AS sends the current global model w_{t-1} to the clients; then, each client initializes the local model $w_t^{(k)} = w_{t-1}$ with w_{t-1} and trains for t iterations; we use stochastic gradient descent (SGD) to optimize the model, so $w_s^{(k)} \leftarrow w_{s-1}^{(k)} - \eta_k g_k\left(\xi_{s-1}^{(k)}; w_{s-1}^{(k)}\right), s = \{1, 2, \ldots, \tau_k\}$, where η_k is the learning rate, $\xi_{s-1}^{(k)} \subset S^{(k)}$ is a random small batch sample with size bn_k, and $g_k(\xi_k; w) = \frac{1}{bn_k}\sum_{z_j^{(k)}} \nabla L\left(z_j^{(k)}; w\right)$ denotes the stochastic gradient. After that, the clients compute the local weights $w_t^{(k)} - w_{t-1}$ and send it back to the AS. Eventually, the AS aggregates the local weights and aggregates a global model $w_t \leftarrow w_{t-1} + \sum_{k=1}^{N} p_k\left(w_t^{(k)} - w_{t-1}\right)$.

The process iterates until the number of communication rounds reaches the specified value T or until the performance requirements are met

C. Attacker Model

The common ways of backdoor attacks include data poisoning and model poisoning. In data poisoning, adversary $\mathcal{A}$ can modify some local data by flipping data labels or adding triggers simultaneously. In model poisoning, it is required that $\mathcal{A}$ has complete control over the client. $\mathcal{A}$ not only poisons the training dataset but also manipulates the local training process by modifying and scaling the local weights. Scaling up the weights of the malicious model increases the impact of the attack (e.g., model replacement attack [26] or projected gradient descent (PGD) attack using model replacement [27]. To escape anomaly detection, $\mathcal{A}$ limits the scaling (e.g., training and scaling [26]), or reduces the deviation of the malicious model from the benign model.

Attacker Capabilities: Just like in the real world, we have no prior knowledge about $\mathcal{A}$. We only make some basic requirements that the number of compromised clients is less than half the number of clients in the FL system. In addition, $\mathcal{A}$ cannot control any process executing on the AS, nor can it manipulate honest clients. Therefore, DO is either honest or malicious in our system, while AS is honest. Then, we give some formal definitions related to attackers.

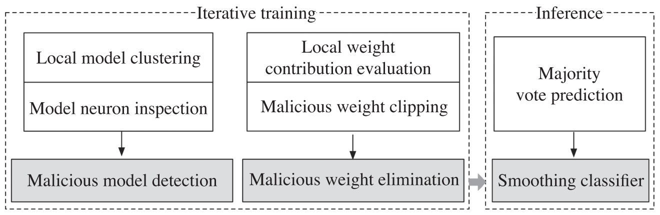

Fig. 3. Our system structure.

Definition 1 (Backdoor Sample): If a malicious DO modifies the local training sample x to x' and its corresponding label y to y', we call the modified sample a backdoor sample, denoted as:

$$x' = Tr\left(x, m_L, p\right) = x * (1 - m_L) + p * m_L, \tag{5}$$

where m_L denotes the shape and position of the trigger, p indicates the color of the trigger, and x is the matrix of the input samples.

Definition 2 (Malicious Client): A malicious DO employing a backdoor sample to train a local model is defined as a malicious client, also known as a compromised client.

Definition 3 (Adversary): The adversary is the leader who directs compromised clients to launch backdoor attacks. An adversary can control multiple compromised clients. In addition, multiple adversaries can exist in the system. The set of adversaries $\mathcal{A}^* = \{\mathcal{A}_1^*, \mathcal{A}_2^*, \ldots\}$. The size of the set $\mathcal{A}^*$ is unknown to us, but its range is an integer belonging to $[0, N/2 - 1]$.

Definition 4 (Malicious Model): A local model trained by a malicious client using backdoor samples is called a malicious model, and its weights are denoted as w'.

Noting that the complete set of local clean datasets as $D = \{S^{(1)}, S^{(2)}, \ldots, S^{(N)}\}$, the data samples in S_i are expressed as $z_j^{(i)} = \{x_j^{(i)}, y_j^{(i)}\}$ and the backdoor samples as $z'^{(i)}_j = \{x_j^{(i)} + \delta_{ix}, y_j^{(i)} + \delta_{iy}\}$. In case, $\mathcal{A}_i^*$ makes S'^{i} backdoor samples of size q_i on a local dataset, the set with backdoor datasets is $D' = \{S'^{(1)}, \ldots, S'^{(R)}, S^{(R+1)}, \ldots, S^{(N)}\}$. We assume that the poisoning ratio of the dataset is $\frac{q_i}{bn_i}$.

V. Our Proposed Scheme: BADFL

Overview: BADFL comprises three primary components: malicious model detection, malicious weight elimination, and a smoothing classifier, as depicted in Fig. 3. Specifically, the first two components are involved in the iterative training aggregation process, while the smoothing classifier is applied during the inference phase. The primary role of the first component is to accurately identify malicious models within the system, which then serves as input for the second component responsible for eliminating malicious parameters. This process involves local model clustering and model neuron inspection. The second component facilitates the contribution of benign parameters from malicious models to the global model, which includes evaluating local weight contributions and clipping malicious model weights. Finally, the last component ensures more robust classification results through majority voting prediction, acting as a final defense against potential misclassifications from the first two components. The overview of BADFL can be summarized as follows:

- Initially, for the exact identification of malicious models and to minimize processing time, we cluster the local models and then conduct individual inspections by randomly selecting models from each cluster. The local model clustering process categorizes the models into multiple classes based on the density of local model similarity, instead of just two clusters. During model neuron inspection, malicious models are identified based on differences in neuron performance between clean samples with and without backdoors.
- Subsequently, we ensure that clean weights in malicious models continue to contribute to the global model by implementing the malicious weight elimination process. In the local weight contribution evaluation process, we evaluate the contribution of malicious model parameters based on the median of clean model weights, rather than considering all model weights.
- Finally, to ensure robust classification results during inference, we employ a smoothing classifier that utilizes majority voting.

Design Challenges: However, we face several challenges in achieving malicious model detection and malicious parameter elimination:

- Our first challenge arises from clustering local model based on model parameter similarity. We observe a progressive decrease in the similarity of model parameters in IID scenarios during the training process. Furthermore, the similarity between local models is even lower in non-IID settings. Consequently, our primary challenge lies in clustering the local models based on the similarity of model parameters.
- After completing clustering, we proceed to randomly sample a subset of local models from each cluster for neuron inspection to identify anomaly classes. Neurons within a model respond to the activation state of each pixel point, showing differences between clean and backdoor samples. Thus, theoretically, identifying malicious local models through neuron examination is feasible. However, as defenders, we lack direct insight into attackers' strategies. Therefore, the challenge lies in identifying malicious models by neurons using clean data and local models.
- The malicious parameter elimination process presents another challenge due to the curse of dimensionality problem with model parameters. Existing methods typically employ a uniform clipping strategy on individual models, inadvertently affecting the benign parameters of malicious models. Consequently, identifying the malicious parameters within malicious models and subsequently neutralizing their effects pose our third challenge.

A. Malicious Model Detection

Current mainstream defenses identify local models as malicious based on low similarity during malicious model detection.

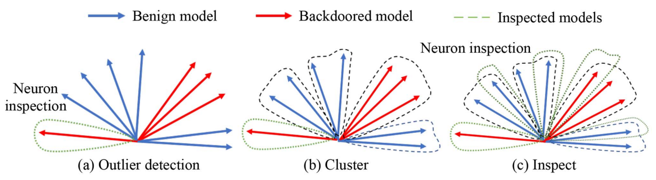

Fig. 4. The detection flow of BADFL. (a) Initially, the LOF algorithm is employed for anomaly detection, followed by neuron inspection of the local models with detected anomalies. (b) Subsequently, the non-anomalous models identified in step (a) are clustered using similarity density. (c) Based on the clustering results from step (b), a subset of local models from each cluster is sampled for neuron inspection.

Algorithm 1: Local Model Clustering Algorithm (LMCA).

```
Input: $Local\_model\_weights$,
       $W = \left\{w^{(1)}, w^{(2)}, \cdots, w^{(n)}\right\}$
Output: $Malic\_cand$
1 $Malic\_cand \leftarrow$ LocalOutlierFactor($W$);
2 $C \leftarrow$ K-means($W \backslash Malici\_cand$);
3 for $i = 1, \cdots, |C|$ do
4     $rmg =$ RandomSelectionParameter($C_i$);
5     $Malic\_cand.append(rmg)$
6 end
```

However, as training progresses, we've found that model similarity naturally decreases, complicating the differentiation between models. To tackle this issue, we've introduced a two-step strategy comprising a local model clustering algorithm and a model neuron inspection algorithm. Diverging from traditional methods, our approach utilizes clustering not directly to pinpoint malicious models but to efficiently sample and inspect models within each cluster, as illustrated in Fig. 4.

1) Local Model Clustering Algorithm: In the Section Introduction, we analyzed that there are significant differences between benign local weights. Hence it is not sufficient to filter malicious models based on their similarity. Existing clustering-based defenses identify the suspicious model by clustering local weights into two clusters, where the small cluster is always considered malicious and removed. The state-of-the-art FLAME believes that fixing the number of clusters increases the risk of malicious models entering within benign model clusters. Therefore, they use a dynamic clustering algorithm that sets the minimum cluster size to $\frac{N}{2} + 1$ and obtains only one cluster at the end. However, the similarity variations make this requirement impossible to guarantee. Hence, FLAME also unsuccessfully separates the benign from the malicious. BADFL first finds k_1 local local weights with significant differences by LOF and then clusters the reachable density of local models. Here we use K-means for clustering and set the number of k to $\frac{N-k_1}{2} + 1$. Then one from each cluster is randomly selected. Finally, $\frac{N+k_1}{2} + 1$ local models will be checked for neurons. BADFL guarantees that most benign models are not deleted by mistake. The specific algorithm is shown in Algorithm 1.

The AS first selects the local weights that are not similar to others using the LOF algorithm (line 1). Then, the local weights

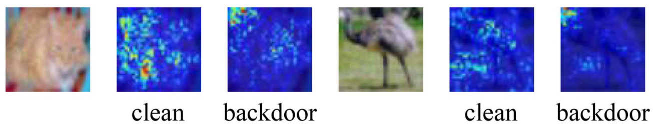

Fig. 5. Interpretable heatmaps of different images for different models.

are divided into multiple classes according to the local reachable density (line 2). Finally, some local weights are randomly selected from each cluster for neuron inspection (lines 3 to 5). After completing the local model clustering, we determine which local models require examination in the subsequent step.

2) *Model Neuron Inspection Algorithm:* RLR [9] identifies malicious parameters by recognizing differences in optimization goal directions between attackers and honest clients on specific dimensions. However, when dealing with non-IID data distributions across clients, their optimization goal directions may also differ, rendering the method less effective in accurately identifying malicious models. Interpretability techniques play a crucial role in understanding how models make decisions. By gaining insights into the decision-making process of a local model, we can detect if the model is under attack. If the model emphasizes irrelevant features of input images for classification, it is likely to be compromised by a backdoor attack. To achieve this, we generate interpretable heatmaps for all categories using a set of clean images and then identify outliers in these interpreted maps.

Without any backdoor samples, it is difficult to generate an interpretable heatmap with only clean data. Therefore, during the generation of the interpretable heatmap, we need to modify the generation algorithm for our task [28]. First, given a deep neural network, we use guided back-propagation (GBP) instead of traditional back-propagation (BP). The difference between the two methods is that GBP can back-propagate only the positive values of the CNN derivatives through ReLU. Therefore, the noise component is smaller than that of normal BP. Then, we replace the softmax layer with a linear layer, which can maximize output by minimizing the other outputs. However, the particular output in the softmax layer depends on the other output. To find the trigger locations hidden in the weights of a neural network of clean images, we should pay more attention to the regions that contribute more positively to the output.

We observe from each label's saliency mapping that the attack target's heatmap has a low internal correlation and is most stable over different images as shown in Fig. 5. In Fig. 5, we analyze two clean samples and generate heat maps highlighting the attacker-specified targets on both the clean model and the backdoor model. Notably, the heat map of the backdoor model indicates a strong focus on the upper-left region of the input image, precisely where we embed the trigger. Additionally, the backdoor model allocates considerably less attention to other regions of the input image compared to the clean model. Therefore, we design features from the interpretation of the heatmap.

The interpretable heatmap for the normal model highlights all pixels in the input image relevant to the output prediction of the DNN. In contrast, the interpretatable heatmap highlights

Algorithm 2: Model Neuron Inspection Algorithm (MNIA).

```
    Input: Local_model_gradients
           $W = \left\{w^{(1)}, w^{(2)}, \cdots, w^{(n)}\right\}$,
    Output: Malicious_models
1   $Malic\_cand \leftarrow LMCA(W)$;
2   $Local\_models \leftarrow Malic\_cand$;
    // Check the neurons of each local
       model
3   for $k = 1, \cdots, n$ do
4       $f_{correlation_k}(M) \leftarrow \sum_{i=1}^{h}\sum_{j=1}^{w}|M_{i,j}|$;
5       $f_{stability_k}(M_1, M_2, \cdots, M_k) \leftarrow$
          $\|H(M_1) \oplus H(M_2) \cdots \oplus H(M_k)\|_1$;
6       $f_k \leftarrow \lambda_{co} f_{correlation} + \lambda_{st} f_{stability}$;
7       $Real\_Malic \leftarrow$
          $K - means(f_{correlation}, f_{stability}, f)$
8   end
```

more the location of the triggers, which indicates that the malicious model has less correlation with the interpretable heatmap. Therefore, we design the correlation function as follows:

$$f_{correlation}(M) = \sum_{i=1}^{h}\sum_{j=1}^{w}|M_{i,j}|, \tag{6}$$

where $|M_{i,j}|$ is the output of saliency map.

For a backdoor model, we observe it is stable on different images for the output heatmap corresponding to the attack target label. Thus, we put forward the following property to measure the stability of the output explanation.

$$f_{stability}\left(M_1, M_2, \ldots, M_k\right) = \left\|H\left(M_1\right) \oplus \cdots \oplus H\left(M_k\right)\right\|_1, \tag{7}$$

where $\oplus$ refers to the XOR computation of two Boolean matrices, H indicates a hash function that encodes a continuous matrix into a binary. In this paper, we adopt the difference hashing algorithm. The input sample $M_1, M_2, \ldots, M_k$ are clean samples.

The two features described above can successfully detect the attack target in the majority of cases, respectively. However, there is occasional misreporting. We associate these two features with weighting coefficients as a balance between the different components to solve this problem.

$$f = \lambda_{co} f_{correlation} + \lambda_{st} f_{stability}. \tag{8}$$

With the features extracted from the interpretable heatmap, we can identify specific mappings that show small correlations and high stability outliers. Thus, we treat each local model's correlation, stability, and combined features as characteristics and perform a secondary clustering to identify the malicious model. The specific algorithm is described in Algorithm 2.

The first line of Algorithm 2 calls Algorithm 1 to get the local weights that need to be checked, and the second line gets the local model based on the local weights. Then, the algorithm's line 3 to 10 performs a neuron inspection on the local model. In lines 4 to 6, the AS calculates the features extracted from interpretable heatmap correlation. Finally, line 7 performs a clustering to identify malicious models.

B. Malicious Local Weight Elimination

Existing defenses typically handle malicious models by either directly removing them upon detection or applying uniformly reduced aggregation weights to the entire model. However, we believe that benign parameters within malicious models can actually enhance global model performance. As a result, we focus on reducing the influence of malicious model parameters rather than entirely eliminating them. Our approach involves designing a model parameter contribution function that uses benign model parameters as a criterion. We then use this function value to clip the malicious model parameters, specifically targeting high-impact parameters within the malicious model while preserving the beneficial benign parameters.

1) Malicious Local Weight Clipping Algorithm: Malicious model also contains information trained on clean data because the attacker needs to ensure that the main task can be achieved. We design the local weight contribution function to evaluate the importance of local weight. The local weight contribution is defined as:

$$Contri(w_{t,j}^{(i)}) = \begin{cases} \frac{median(w_{t,j})}{w_{t,j}^{(i)}}, & if\ w_{t,j}^{(i)} \geq median(w_{t,j}) \\ 1 \quad\quad\quad , & else \end{cases}, \tag{9}$$

where $median(w_{(t,j)})$ denotes the median of the j-th weight of the other benign models in round t. We compare the weight in the malicious model with the median of the local weights in the benign models. The local weight that exceeds the median impacts the global model significantly. Therefore, the AS clips such weights according to their contributions, $Clip\rho_t(w_t^{(i)}) \leftarrow w_t^{(i)} \cdot \min(1, \rho_t)$, where $\rho_t = Contri(w_t^{(i)})$. So $w_t^{(i)}$ is bounded by ρ_t, and added Gaussian noise for perturbation, eventually aggregating with other local weights. The clipping function can be defined as follows:

$$Clip_{\rho_t}\left(w_t^{(i)}\right) = \begin{cases} w_t^{(i)} \cdot \min(1, \rho_t), & if\ w_t^{(i)} \text{ is malicious} \\ \quad w_t^{(i)} \quad\quad\quad , & else \end{cases}. \tag{10}$$

The malicious local weight clipping algorithm is shown in Algorithm 3.

In line 2 of Algorithm 3, the AS sends the global model to DOs. Each DO trains local models locally in lines 3 to 6. In line 7, the AS detects the malicious local weight. And the adaptive dynamic local weights clipping step is shown in line 8. The AS aggregates the local weights in line 9. Finally, noise is applied to the global model in lines 10 to 13.

C. Smoothing Classifier

We discuss the multi-classification model and describe a classifier $h : (X; W) \rightarrow Y, Y = \{y_1, y_2, \dots, y_C\}$, where C denotes the number of classes. We apply the stochastic smoothing method [29] to parameter smoothing as a means of constructing a new smoothed classifier h_s on an optional base classifier h.

Algorithm 3: Malicious Local Weight Clipping Algorithm.

```
   Input: $AS$'s input: initial model weights
          $w_0, \widetilde{w}_0 \leftarrow w_0$; $DO$'s input: local dataset $S^{(i)}$
          and learning rate $\eta_i$
   Output: Global model weights $w_T$
1  for $t = 1, \cdots, T$ do
2      The $AS$ sends $\widetilde{w}_{t-1}$ to the $DOs$
3      for $i = 1, \cdots, N$ do
4          Initialize local model $w_t^{(i)} \leftarrow \widetilde{w}_{t-1}$;
5          Local train and send $w_t^{(i)} - \widetilde{w}_{t-1}$ to $AS$;
6      end
       // Detect the malicious local weight
7      $Real\_Malic \leftarrow MNIA(Candidates)$;
       // Clip the malicious local weights
8      $Clip_{\rho_t}(w_t^{(i)}) \leftarrow w_t^{(i)} \cdot \min(1, \rho_t)$;
       // Aggregate the local weights
9      $w_t \leftarrow \widetilde{w}_{t-1} + \sum_{i=1}^{R_t} p_i Clipe_{\rho_t}(w_t^{(i)}) + \sum_{i=R_t+1}^{N} p_i w_t^{(i)}$;
10     if $t < T$ then
           // Add noise
11         $\varepsilon_t \leftarrow$ A sample drawn from $\mathrm{N}(0, \sigma_t^2 \mathrm{I})$;
12         $\widetilde{w_t} \leftarrow w_t + \varepsilon_t$
13     end
14 end
```

The smoothing classifier can be verified for robustness. Considering the model h parameters w, on querying a test sample x_{test}. We first perform a majority vote for the prediction in the probability distribution over the random model parameters with the base classifier h. Then, the smoothing classifier comes back with the most probable labels (majority voting winners) in all categories [30]. Formally,

$$h_s(x_{test}; w) = \arg\max_{y \in Y} H_s^y(x_{test}; w),$$
$$H_s^y(x_{test}; w) = \mathrm{P}_{w \sim \mu(w)}[h(x_{test}; w) = y]. \tag{11}$$

To be consistent with the Gaussian noise (perturbation) within the training phase, we continue applying the Gaussian smoothing metric $\mu(w) = N(w, \sigma_t^2 \mathrm{I})$ during the test time. For the neural networks, it is difficult to get the exact value of the probability $p_y = \mathrm{P}_{w \sim \mu(w)}[h(x_{test}; w) = y]$ of the label y in practice. Therefore, we employ Monte Carlo estimation [21], [29] to obtain a close approximation $\widehat{p_y}$. In round $t = T$, we acquire M sets of noise model parameters by adding M times Gaussian noise to the given global model w_T. We classify the test sample with a set of model parbameters $\{\widetilde{w}_T^{(1)}, \widetilde{w}_T^{(2)}, \dots, \widetilde{w}_T^{(M)}\}$ which returns a class count vector. As a result, we select the most likely class $\widehat{y}_A$ and the second most likely class $\widehat{y}_B$ for the corresponding probability $\widehat{p}_A$ and $\widehat{p}_B$. Considering the error tolerance α, we perform the lower bound $\underline{p_A}$ of $H_s^{y_A}(x_{test}; w)$ and the upper bound $\overline{p_B}$ of $H_s^{y_B}(x_{test}; w)$ according to $\underline{p_A} = \widehat{P}_A - \sqrt{\frac{\log(1/\alpha)}{2\,N}}$, $\overline{p_B} = \widehat{P}_B + \sqrt{\frac{\log(1/\alpha)}{2\,N}}$, using the Hoeffding inequality [31]. If $\underline{p_A} > \overline{p_B}$, then the certified radius CR is obtained according to $\underline{p_A}$ and $\overline{p_B}$. The calculation of CR is presented later in the security analysis of BADFL together.

VI. Theoretical Analysis of BADFL

In this section, drawing inspiration from [32] and [33], we begin by analyzing the security aspects during the training phase, followed by an examination of security concerns during the inference phase. Finally, we assess the convergence properties of BADFL. To present our analysis, we outline these common assumptions in Appendix, available online.

A. Training Phase Security

Theorem 1: When $\eta_i \leq \frac{1}{\beta}$ and Assumptions 1 and 2 satisfied, assume $y_A \in Y, \underline{p_A}, \overline{p_B} \in [0,1]$ satisfies $H_S^{y_A}(x_{test}; A(D')) \geq \underline{p_A} \geq \overline{p_B} \geq \max_{y \neq y_A} H_S^{y}(x_{test}; A(D'))$, if

$$\frac{\varepsilon}{\Pi} \geq \prod_{t \in T_{adv}} \frac{2R_t\Sigma}{\sigma_t^2}. \tag{12}$$

where $\Pi = \prod_{t \in T \setminus T_{adv}} 2\Phi(\frac{\rho_t}{\sigma_t}) - 1, \Sigma = \sum_{i=1}^{R_t}(\alpha_i p_i \eta_i \frac{q_i}{bn_i}\|\delta_i\| \tau_i)^2$

As well, we can restate Theorem 1 as the following corollary.

Corollary 1: When $\eta_i \leq \frac{1}{\beta}$ and Assumptions 1 and 2 satisfied, assume $y_A \in Y$, $\underline{p_A}$, $\overline{p_B} \in [0,1]$ satisfies $H_S^{y_A}(x_{test}; A(D')) \geq \underline{p_A} \geq \overline{p_B} \geq \max_{y \neq y_A} H_S^{y}(x_{test}; A(D'))$ $\|\delta\| = \|\delta_i\|$ and $h_s(x_{test}; A(D')) = h_s(x_{test}; A(D)) = y_A$, if $\|\delta\| < CR$,

$$CR = \sqrt[2|T_{adv}|]{\frac{-\log\left(1 - \left(\sqrt{\underline{p_A}} - \sqrt{\overline{p_B}}\right)^2\right)}{\prod_{t \in T_{adv}} \frac{2R_t\Sigma}{\sigma_t^2}\Pi}}. \tag{13}$$

Theorem 2: When $\eta_i \leq \frac{1}{\beta}, u(w) = \mathrm{N}(w, \sigma_T^2 \mathrm{I})$ and Assumptions 1 and 2 satisfied, there is a bounded KL divergence between $u(A(D))$ and $u(A(D'))$

$$D_{KL}\left(u(A(D))\,\|u(A(D'))\right) \leq \prod_{t \in T_{adv}} \frac{2R_t\Sigma}{\sigma_t^2}\Pi. \tag{14}$$

The detailed proof is given in the Appendix A, available online.

B. Inference Phase Security

By Theorem 2, we associate the model closeness with the prediction's consistency. The smoothing classifier h is guaranteed to be bounded at $u(w')$ concerning the KL divergence, $D_{KL}(u(w), u(w')) \leq \varepsilon$.

Theorem 3: Assume $y_A \in Y$, $\underline{p_A}$, $\overline{p_B} \in [0,1]$ satisfy $H_S^{y_A}(x_{test}; u(w')) \geq \underline{p_A} \geq \overline{p_B} \geq \max_{y \neq y_A} H_S^{y}(x_{test}; u(w'))$ $\|\delta\| = \|\delta_i\|$ and $h_s(x_{test}; u(w')) = h_s(x_{test}; u(w)) = y_A$ for all w such that

$$D_{KL}\left(u(w), u(w')\right) \leq \varepsilon, \tag{15}$$

where $\varepsilon = -\log(1 - (\sqrt{\underline{p_A}} - \sqrt{\overline{p_B}})^2)$.

The proofs are given in the Appendix B, available online. Finally, combining Theorems 2 and 3 yields 1.

C. Convergence Analysis

In this section, we demonstrate that BADFL converges under the same assumptions as other methods [25], [34] when there are no malicious client attacks in the FL system.

In our defense strategy, we initially perform clustering and neuron checking on the model. Subsequently, we apply clipping and perturbation to the updates based on the parameters' contributions. When there are no malicious client attacks on the system, the convergence bound of BADFL remains consistent with vanilla FedAvg, provided the detection process does not misclassify the benign model as malicious. However, if there is a misclassification during detection, we resort to clipping the local model and adding perturbations, which alters the convergence bounds. This clipping process can be seen as a modification of the aggregation weights of the local update. Let's consider a scenario where all clients participate in FL. We assume that the detection algorithm suffers from misclassification, and based on Assumptions 3–6, we can derive the following convergence guarantees for our defense on FedAvg.

Theorem 4 (Convergence Guarantee): Assuming A.3 to A.6 hold and ℓ, μ, ζ_k, G be defined therein. Choose $\iota = \frac{\ell}{\mu}$, $\varsigma = \max\{8\iota, \tau\}$Then we obtain the following bound on BADFL:

$$\mathbb{E}\left[L(\mathbf{w}_T)\right] - L^* \leq \frac{2\iota}{\varsigma + T}\left(\frac{B}{\mu} + 2\ell\|\mathbf{w}_0 - \mathbf{w}^*\|^2\right) \tag{16}$$

where

$$B = \sum_{k=1}^{N} p_k^2 \rho_k^2 \zeta_k^2 + 6\ell\Gamma + 8(\tau - 1)^2 G^2. \tag{17}$$

VII. Experiment Analysis

A. Experiment Setup

The general setup of our experiments is as follows: In the absence of special instructions, we simulate a 100-rounds FL process including 100 clients. Before sending local weights, the clients perform local training on E rounds with the local dataset's size n_i. To highlight the advantages of BADFL, except for four recent robust schemes, i.e., CRFL [18], RLR [9] and FLAME [15], FLTrust [16], we establish a control group, unmodified FedAvg, as a Baseline which takes average as the aggregation rule. Then, we evaluate BADFL against the state-of-the-art backdoor attacks called constrain-and-scale [26], DBA [7], and Edge-Case [27].

Dataset: Our experiments are performed on 3 prominent image classification datasets, namely MNIST, EMNIST, and CIFAR-10. In practical applications, our data often deviate from the assumption of being independent and identically distributed (i.i.d.). To address this, we adopt a methodology consistent with Baghdashayan et al. which models the non-i.i.d. distribution by employing the Dirichlet distribution with α as a parameter of 0.5. The poison ratio signifies the percentage of poison data added to every training batch, and by default, it is set at 50%, meaning that half of each batch consists of poison data when no other value is specified.

Model Architectures: We find that different model structures in the experiment affect the clustering results. However, malicious models are identified by neuron inspection, so different models do not affect the detection results. Then, for MNIST and EMNIST, we employed an image classification task utilizing a federated setup. The task consists of a 5-layer Convolutional Neural Network (CNN) with the following structure: two convolutional layers, one max-pooling layer, and two fully connected layers. On the other hand, for CIFAR-10, we employed the ResNet18 model as the federated task. ResNet, short for Residual Network, is a deep neural network architecture known for its ability to handle complex tasks, particularly in image classification. This setup allows us to perform image classification tasks effectively on MNIST, EMNIST, and CIFAR-10 datasets using different models, depending on the dataset, to achieve the best results.

Performance Metrics: We employ four evaluation indicators to demonstrate the effectiveness of the proposed scheme from various perspectives. These indicators include global model accuracy (ACC), attack success rate (ASR), true positive rate (TPR), and true negative rate (TNR), and they are defined as follows:

- *Accuracy (ACC):* To evaluate the impact of defenses on the performance of the global model, we use the accuracy of the global model on the benign test set. We set a control group of Baseline, an unmodified FedAvg. Thus, as the ACC of the defense scheme gets closer to Baseline, it indicates that the defense has less impact on the global model performance.
- *Attack Success Rate (ASR):* To evaluate the defensive performance, we also measure the accuracy of the global model on the test data using backdoor triggers, i.e., attack success rate (ASR). The lower the ASR, the better the defense performance of the global model.
- *True Positive Rate (TPR):* TPR indicates the ability of the defense system to identify malicious models, i.e., the rate of the number of models correctly classified as malicious (True Positive - *TP*) to the number of models classified as malicious: $TPR = \frac{TP}{TP+FP}$, where FP is a false positive indicating the number of models misclassified as malicious.
- *True Negative Rate (TNR):* TNR calculates the rate of the number of models correctly classified as benign (true negative - *TN*) to those classified as benign: $TNR = \frac{TN}{TN+FN}$, where FN is a false negative indicating the number of models misclassified as benign.

B. Heatmap

To demonstrate the distinctions in explanation heatmaps between the backdoored model and the clean model on clean samples, we employ the BadNets attack on ResNet18 with a 10% poisoning rate and a target label of class 0, training it for 50 epochs to obtain a backdoored model. Due to spatial limitations, we randomly selected two clean samples and generated explanation heatmaps for both the backdoored and clean models, illustrated in Fig. 6. Upon analyzing the saliency heatmaps of

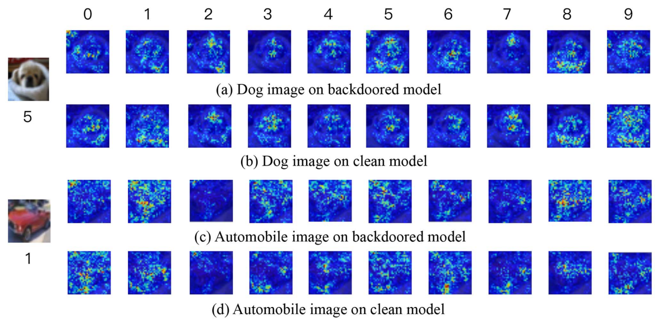

Fig. 6. Different images on different models with different labels heatmaps.

TABLE III
COMPARISON WITH EXISTING WORKS IN ACC AND ASR

Baseline	MNIST		E-MNIST		CIFAR-10	
	ACC	ASR	ACC	ASR	ACC	ASR
FedAvg	0.9866	-	0.7973	-	0.7523	-
FLAME [15]	0.9132	0.1298	0.7608	0.1099	0.6873	0.1582
RLR [9]	0.9763	0.3628	0.7134	0.0671	0.7321	0.0752
CRFL [18]	0.8894	0.0302	0.5644	0.0671	0.6911	0.0782
FLTrust [16]	0.9726	0.4812	0.7992	0.0769	0.7244	0.1218
Ours	0.9861	0.0004	0.7958	0.0592	0.7516	0.0314

the backdoored model, we observed notable consistency in the attention regions for the target label across diverse image sets. In essence, the attention regions for label 0 remain stable across different images. Moreover, the internal feature correlations within the heatmap for label 0 are relatively weak. Conversely, upon inspecting the saliency heatmaps of the clean model, we note significant fluctuations in the attention regions across varying images, alongside a pronounced feature correlation within individual heatmaps. Hence, it is reasonable to detect backdoored models using explanation heatmaps on clean samples.

C. Compared With Existing Works

Comparison With Existing Works in ACC and ASR: Table III shows a comparison of the performance of existing defenses with our scheme on different datasets. FLAME suffers from the problem of identifying malicious models as benign models, resulting in the ASRs above 10%. CRFL can achieve a better defense using a uniform clipping and perturbation strategy, and its ASR can stay below 7%. However, clipping and perturbation impose a large impact on the ACC, making the ACC of CRFL on the EMNIST dataset only 56.44%. Compared with CRFL, our method outperforms its ACC by 23.14%. RLR identifies malicious weights by checking each local weight, which is a simple approach and does not affect the ACC. However, RLR fails to defend effectively, whose attack success rate is as high as 43.01% in the worst case. Our proposed scheme provides an effective defense against backdoor attacks by performing neuron inspection on individual models to identify malicious models. In addition, we reserve the benign weights in the malicious model so that our ACC is very close to the Baseline. Our

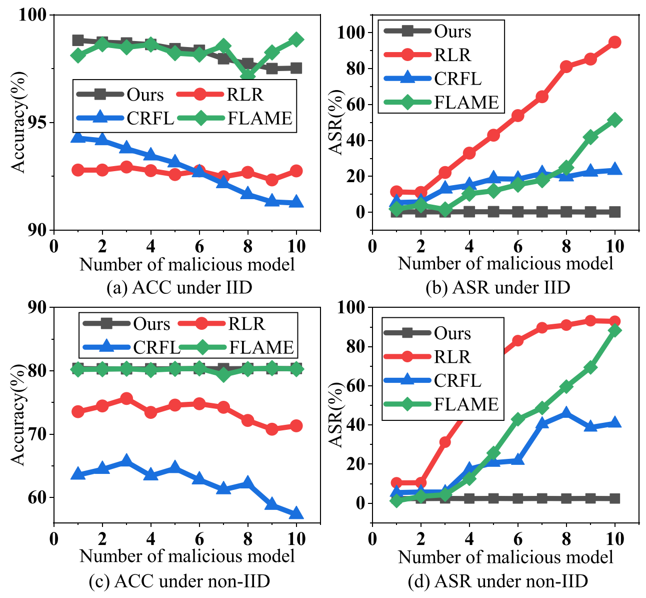

Fig. 7. Comparison with existing works in ACC and ASR.

method trades computational costs for better model accuracy and defense performance. The aggregation server has significant computational power, so this computational cost is acceptable.

Robustness of BADFL With Different Numbers of Malicious Clients: To highlight the robustness of BADFL, we investigate how different numbers of malicious clients affect defense performance. As shown in Fig. 7, our method is relatively stable compared to the others with increased malicious clients. Fig. 7(b) and (d) show that ours do not increase substantially in ASR. The addition of small perturbations results in the weak defense of FLAME. When the number of malicious clients does not exceed the defense threshold of CRFL, the increased malicious client number does not significantly impact CRFL. However, the ACC of CRFL is the lowest when the number of malicious clients exceeds 5 in Fig. 7(a) and (c).

Malicious Local Weight Detection: TPR and TNR evaluate the performance of malicious model detection. Since FLAME is also involved in malicious model detection, we only compare it with FLAME. FLAME is relatively close to ours because we use clustering in the malicious model detection phase. However, unlike FLAME, our clustering is not designed to filter malicious models. Instead, we employ clustering to classify local models into $\frac{N-k_1}{2}+1$ clusters and select a local model from each cluster for neuron inspection. We add the clustering process because neuron inspection requires more time than clustering. The comparison results are shown in Tables IV and V. FLAME not only misidentifies malicious models but also identifies benign models as malicious ones. The TNR of ours is basically above 90%. The most malicious models are identified for different poisoning rates and the number of malicious models.

D. Ablation Experiments

Effectiveness of Different Components: Our scheme comprises three main components: malicious model detection

TABLE IV
COMPARISON OF DETECTION PERFORMANCE WITH DIFFERENT POISONING RATES

Defenses	Different poisoning rates	TPR	TNR
FLAME [15]	0.3	0.249	0.495
	0.4	0.249	0.495
	0.5	0.251	0.465
	0.6	0.315	0.485
	0.7	0.358	0.465
	0.8	0.367	0.448
Ours	0.3	0.761	0.979
	0.4	0.813	0.947
	0.5	0.826	0.978
	0.6	0.913	0.986
	0.7	0.938	0.963
	0.8	0.937	0.983

TABLE V
COMPARISON OF DETECTION PERFORMANCE WITH DIFFERENT NUMBERS OF ATTACKERS

Defenses	Number of malicious model	TPR	TNR
FLAME [15]	2	0.316	0.483
	4	0.283	0.498
	6	0.251	0.465
	8	0.283	0.515
	10	0.341	0.526
Ours	2	0.916	0.968
	4	0.883	0.947
	6	0.826	0.984
	8	0.913	0.966
	10	0.938	0.973

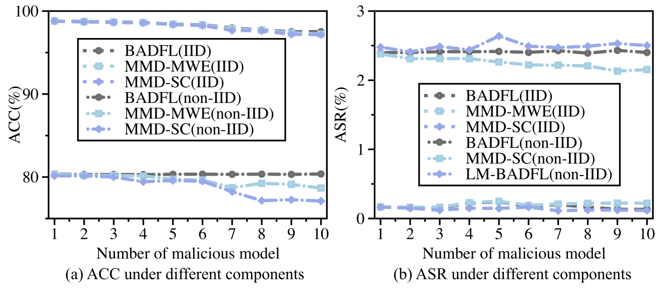

Fig. 8. The effectiveness of different components.

(MMD), malicious parameter elimination (MWE), and smoothing classifier (SC). Among them, detection is a crucial step, as subsequent clipping relies on its results. To assess the effectiveness of other components in the proposed scheme, we compare the model's performance after removing MWE and SC. This procedure is illustrated in Fig. 8. We conducted experiments in both non-IID and IID scenarios. From Fig. 8, it is evident that removing MWE leads to a degradation in global model accuracy while removing SC results in a slight decline in both defense performance and global model accuracy.

Effectiveness of Malicious Model Detection: Fig. 9 demonstrates the effectiveness of the malicious model detection component in BADFL, using HDBSCAN and K-means as comparative methods for detecting malicious models. To ensure fairness in the evaluation process, we replace our proposed detection

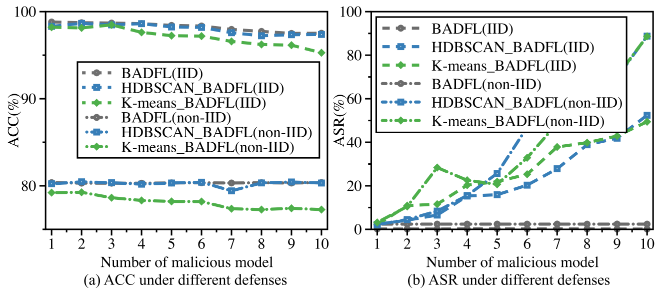
Fig. 9. The effectiveness of malicious model detection.

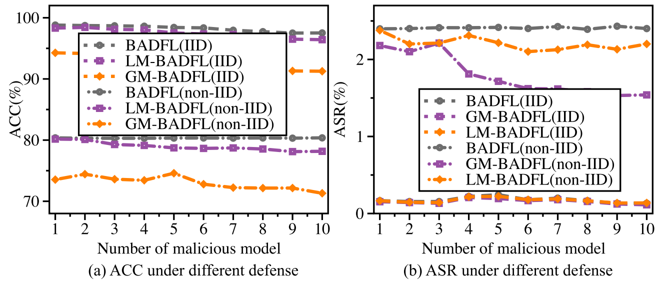
Fig. 10. The effectiveness of malicious weight elimination.

method with HDBSCAN and K-means under IID and non-IID settings for comparison, denoted as HDBSCAN_BADFL and K-means_BADFL, respectively. As depicted in Fig. 9, there is a significant increase in ASR, while ACC experiences a slight decrease after substituting the detection algorithms. This can be attributed to the subsequent step of eliminating malicious parameters, which involves computing the clipping bounds of the malicious model parameters based on those of the benign model. Hence, our proposed detection method plays a crucial role in defending against backdoor attacks.

Effectiveness of Malicious Weight Elimination: Fig. 10 illustrates the effectiveness of malicious weight elimination in BADFL. We compare our approach with the clipping methods of FLAME and CRFL. CRFL utilizes the L2 norm of the global model for clipping, denoted as GM, while FLAME selects the median of the L2 norm of the local model, denoted as LM. In contrast, BADFL clips based on the median of the L2 norm of the layer parameter, providing a more fine-grained approach than FLAME.

During our experiments, we substitute our clipping methods with GM and LM for a comparison. We conduct experiments in both non-IID and IID scenarios. As depicted in Fig. 10, in the non-IID scenario, our approach exhibits advantages over existing methods. Conversely, in the IID scenario, BADFL and LM-BADFL yield similar effects. This is because in the IID scenario, the data between clients are similar, and removing the benign parameters of the malicious model does not significantly impact the global model. In contrast, in the non-IID scenario, the malicious client's local clean data and the honest client's data distribution differ substantially, containing distinct data features. Completely removing the malicious model would result in the loss of the clean data features from the malicious client's local

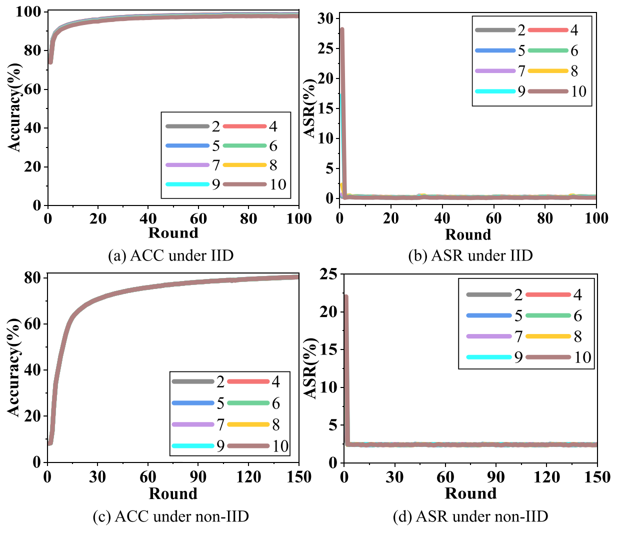
Fig. 11. The effect of different numbers of malicious clients.

data. Therefore, retaining the contribution of benign parameters to the global model can enhance the performance of the global model.

E. The Influence of Hyperparameters in BADFL

Number of Malicious Clients: We analyze the impact of different numbers of malicious clients on defense performance. We aggregate 20 clients and use different colors to indicate the impact of different numbers of attackers. Interestingly, even though we accurately identify the malicious local weights, the global model accuracy is slightly affected, as shown in Fig. 11. We believe it is due to the fact that the increasing number of attackers makes the clean data decrease, resulting in a slight decrease in accuracy.

Adding Noise: We design different noise levels to choose appropriate noise parameters. Fig. 12 shows the effects of different noise levels on two datasets. We use lines of different colors to represent different noise levels. Under the IID, the noise range is [0.001, 0.05]. However, we find that using the same noise level applied to non-IID makes the model divergent. Therefore, under the non-IID, we set the noise level to [0.00001, 0.01]. Results confirm that adding noise affects the ASR and dramatically impacts the global model. From Fig. 12(a) and (b), we can see that when the noise level is set to 0.05, the ACC is only about 10%, and the ASR is close to 0. However, when the range is in [0.001, 0.025], the ACC is above 90%, and the ASR is also lower than 1.5%. Fig. 12(c) and (d) show that the noise increases to 0.01, the ACC is less than 60%, while the ASR is close to 2%.

Number of Clients in the Aggregation: We set the parameter for the number of clients in the aggregation as {20, 25, 30, 35, 40, 45, 50}. From Fig. 13(a) and (b), we can see that this parameter has a minimal impact on our method. After 100 rounds of training, the ACC can reach more than 95%. At the same time, the ASR is also as low as about 0.5%. Fig. 13(c) and (d) show the defense effect on the non-IID. The experimental results on the similar distributions dataset are similar to the

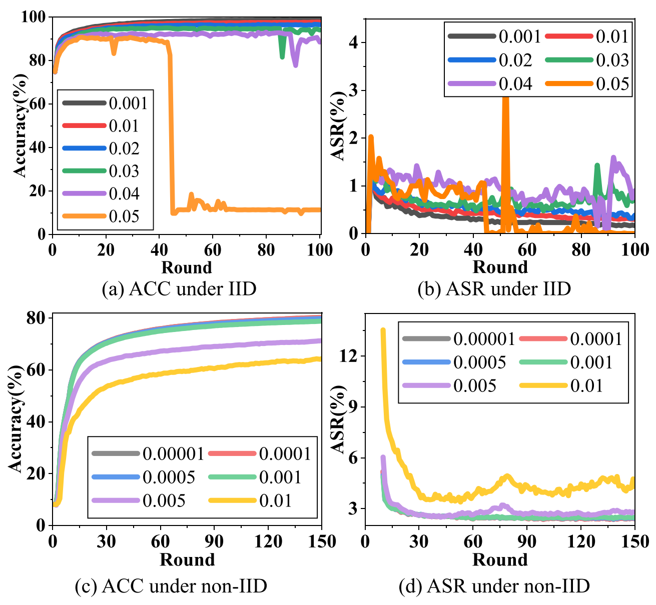

Fig. 12. The effect of adding noise.

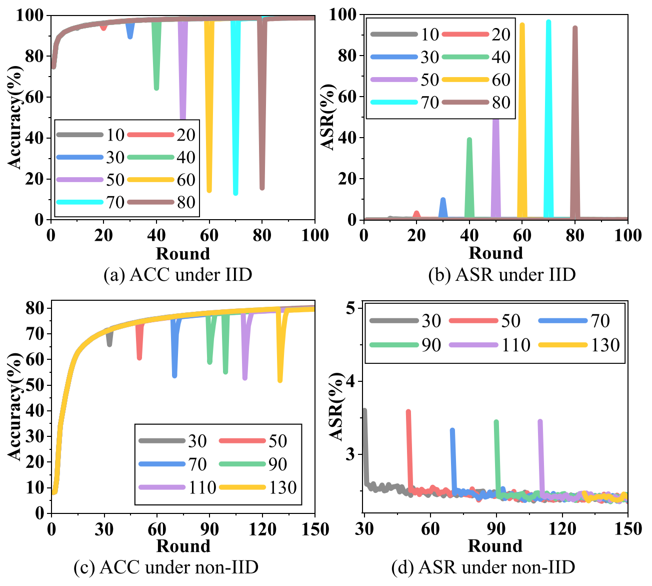

Fig. 14. The effect of the round of poisoning.

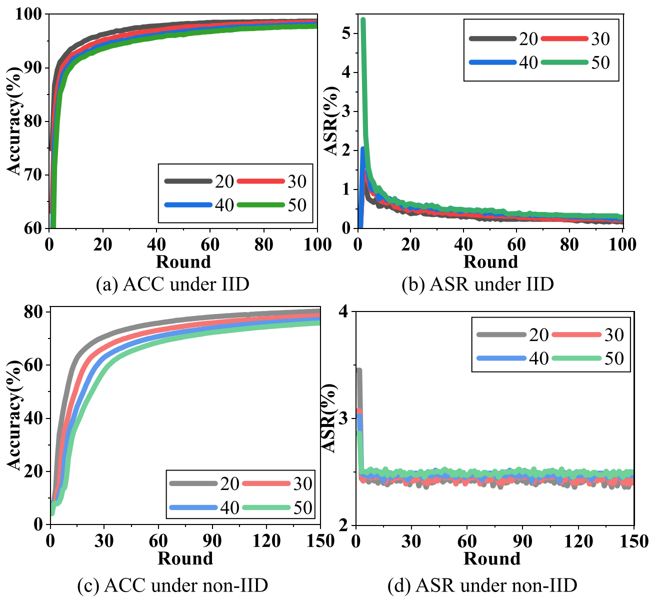

Fig. 13. The effect of the number of clients in the aggregation.

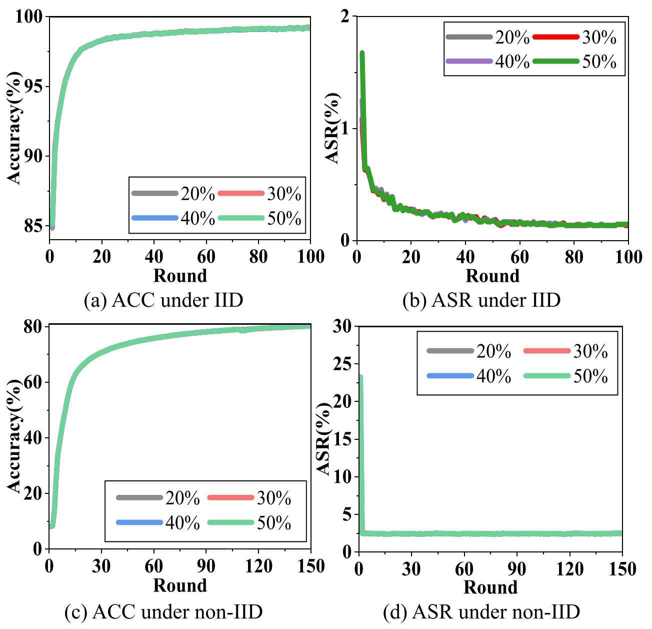

Fig. 15. The effect of the poisoning rate.

different distributions dataset. Therefore, our defense scheme is not sensitive to the clients' numbers in the aggregation.

Round of Poisoning: The effect of the rounds of poisoning on the global model should be different. To visualize this difference, we do not defend in the poisoning round but defend in the next round of poisoning. We set up attacks on different rounds. The number of training rounds under the IID is set to 100, while the rounds of poisoning are set to {10, 20, 30, 40, 50, 60, 70, 80}. The number of training rounds under the non-IID is set to 150, and the rounds of poisoning are set to {30, 50, 70, 90, 110, 130}. Fig. 14 verifies our conjectures. First, poisoning on the different rounds has different effects. In Fig. 14(a), we can see that a backdoor attack at the initial stage of training has little impact on ACC. However, it reduces to less than 70% when the attacker launches a backdoor attack in the 40th round. If the attack is launched after the 60th round, the ACC is only close to 10%. However, our defense can improve the ACC to the same ACC as no backdoor attack after no more than five rounds. Interestingly, poisoning different distributions dataset has little effect on ACC. If local data distributions are diverse, the backdoor training will be discontinuous. Therefore, the effect of the backdoor is reduced.

Poisoning Rate: In addition, we also verify the effect of poisoning rates on our defense shown in Fig. 15. The poisoning rates set for IID are {15%, 20%, 25%, 30%, 35%, 40%, 45%, 50% }, the poisoning rate set for non-IID are {2.5%, 5%, 10%, 20%, 30%, 40%, 50%, 60%}. Fig. 15(a) and (d) show that the poisoning rate has little effect on two datasets. It only affects the ACC and ASR in the current poisoning round. After the defense, the ACC and ASR quickly recover. Thus, our defense is not sensitive to the poisoning rate.

VIII. Conclusion

In this paper, we propose BADFL to defend against backdoor attacks in FL. First, we propose a malicious model detection method based on clustering and interpretability, which takes into account that the similarity of local weights varies with the training rounds. Then, considering that benign local weights in the malicious model also have contributions to the global model, we propose a malicious local weight elimination method based on weight contributions. Finally, we analyze the security of the proposed method from the model closeness and then verify the effectiveness of the proposed method through experiments. In the future, we will also consider detecting encrypted local weights to identify malicious ones. A trustworthy FL must consider ensuring that the user's data privacy remains private while guaranteeing the robustness of the model against adversarial operations.

References

[1] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, pp. 1–210, 2021.

[2] M. Shayan, C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Biscotti: A blockchain system for private and secure federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1513–1525, Jul. 2021.

[3] J. S.-P. Díaz and A. L. García, "Study of the performance and scalability of federated learning for medical imaging with intermittent clients," *Neurocomputing*, vol. 518, no. C, pp. 142–154, 2023.

[4] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "DÏot: A federated self-learning anomaly detection system for IoT," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 756–767.

[5] I. Achituve, A. Shamsian, A. Navon, G. Chechik, and E. Fetaya, "Personalized federated learning with Gaussian processes," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 8392–8406.

[6] S. P. Karimireddy et al., "Breaking the centralized barrier for cross-device federated learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 28 663–28 676.

[7] X. Chulin, H. Keli, C. Pin-Yu, and L. Bo, "DBA: Distributed backdoor attacks against federated learning," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–19.

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

[9] S. Ö. Mustafa, K. Murat, and R. G. Yulia, "Defending against backdoors in federated learning with robust learning rate," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 9268–9276.

[10] Y. Dong, C. Yudong, K. Ramchandran, and B. Peter, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5650–5659.

[11] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *IEEE Trans. Signal Process.*, vol. 70, pp. 1142–1154, Mar. 2023.

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

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

[14] E. M. El Mahdi, G. Rachid, and R. Sébastien, "The hidden vulnerability of distributed learning in Byzantium," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3518–3527.

[15] T. D. Nguyen et al., "Flame: Taming backdoors in federated learning," in *Proc. USENIX Secur. Symp.*, 2022, pp. 1415–1432.

[16] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "FLTrust: Byzantine-robust federated learning via trust bootstrapping," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2021, pp. 1–18.

[17] J. Sun, A. Li, L. DiValentin, A. Hassanzadeh, Y. Chen, and H. Li, "FL-WBC: Enhancing robustness against model poisoning attacks in federated learning from a client perspective," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 12 613–12 624.

[18] C. Xie, M. Chen, P. Chen, and B. Li, "CRFL: Certifiably robust federated learning against backdoor attacks," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 11 372–11 382.

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

[20] M. Naseri, J. Hayes, and E. D. Cristofaro, "Toward robustness and privacy in federated learning: Experimenting with local and central differential privacy," 2020, *arXiv: 2009.03561*.

[21] M. Lécuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "Certified robustness to adversarial examples with differential privacy," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 656–672.

[22] B. Peva, E. M. El Mahdi, G. Rachid, and S. Julien, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 119–129.

[23] F. Clement, J. M. Y. Chris, and B. Ivan, "The limitations of federated learning in sybil settings," in *Proc. 23rd Int. Symp. Res. Attacks Intrusions Defenses*, 2020, pp. 301–316.

[24] T. Zhang, A. Song, X. Dong, Y. Shen, and J. Ma, "Privacy-preserving asynchronous grouped federated learning for IoT," *IEEE Internet Things J.*, vol. 9, no. 7, pp. 5511–5523, Apr. 2022.

[25] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on Non-IID data," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–18.

[26] B. Eugene, V. Andreas, H. Yiqing, E. Deborah, and S. Vitaly, "How to backdoor federated learning," in *Proc. 23rd Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2938–2948.

[27] W. Hongyi et al., "Attack of the tails: Yes, you really can backdoor federated learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, Art. no. 1348.

[28] S. Karen, V. Andrea, and Z. Andrew, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *Proc. Int. Conf. Learn. Representations*, 2014, pp. 1–8.

[29] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1310–1320.

[30] K. D. Dvijotham et al., "A framework for robustness certification of smoothed classifiers using f-divergences," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–26.

[31] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Collected Works Wassily Hoeffding*, pp. 409–426, 1994.

[32] S. Asoodeh, M. Diaz, and F. P. Calmon, "Privacy analysis of online learning algorithms via contraction coefficients," 2020, *arXiv: 2012.11035*.

[33] R. L. Dobrushin, "Central limit theorem for nonstationary Markov chains. I," *Theory Probability Appl.*, vol. 1, no. 1, pp. 65–80, 1956.

[34] C. Zhu, S. Roos, and L. Y. Chen, "LeadFL: Client self-defense against model poisoning in federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 43 158–43 180.

Haiyan Zhang received the MS degree in computer science and technology from Anhui Normal University, in 2021. She is currently working toward the PhD degree with the School of Security of Cyberspace, Xidian University, China. Her research interests include AI model security and privacy protection.

Xinghua Li (Member, IEEE) received the ME and PhD degrees in computer science from Xidian University, China, in 2004 and 2007, respectively. He is currently a professor with the School of Cyber Engineering, Xidian University. His research interests include wireless networks security, privacy protection, cloud computing, and security protocol formal methodology.

Mengfan Xu received the PhD degree in computer system structure from Xidian University, in 2020. His main research interests include network and information security, IoT attack detection.

Ximeng Liu (Senior Member, IEEE) received the BS degree in electronic engineering and the PhD degree in cryptography from Xidian University, Xi'an, China. He is currently a full professor with the College of Mathematics and Computer Science, Fuzhou University. He was a research fellow with the School of Information System, Singapore Management University, Singapore. His research interests include cloud security, applied cryptography, and Big Data security. He is a member of the ACM.

Tong Wu received the BS degree in information security from the China University of Mining and Technology, in 2020. He is currently working toward the MS degree in cyberspace security with Xidian University. His research interests include AI model security and watermark.

Jian Weng (Member, IEEE) received the PhD degree from Shanghai Jiao Tong University, Shanghai, China, in 2008. He is currently a professor with the College of Information Science and Technology, Jinan University, Guangzhou, China. His research interests include public key cryptography, cloud security, blockchain, etc. He served as a PC co-chairs or PC member for more than 20 international conferences.

Robert H. Deng (Fellow, IEEE) is AXA chair professor of cybersecurity and professor of information systems with the School of Information Systems, Singapore Management University since 2004. His research interests include data security and privacy, multimedia security, network, and system security. He served/is serving on the editorial boards of many international journals, including the *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Dependable and Secure Computing*. He has received the Distinguished Paper Award (NDSS 2012), Best Paper Award (CMS 2012), Best Journal Paper Award (IEEE Communications Society 2017).