Computers & Security 129 (2023) 103205

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Two-phase Defense Against Poisoning Attacks on Federated Learning-based Intrusion Detection

Yuan-Cheng Lai[a,*], Jheng-Yan Lin[a], Ying-Dar Lin[b], Ren-Hung Hwang[c], Po-Chin Lin[d], Hsiao-Kuang Wu[e], Chung-Kuan Chen[f]

[a] Department of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan
[b] Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan
[c] College of Artificial Intelligence, National Yang Ming Chiao Tung University, Tainan, Taiwan
[d] Department of Computer Science & Information Engineering, National Chung Cheng University, Chiayi, Taiwan
[e] Department of Computer Science & Information Engineering, National Central University, Taoyuan, Taiwan
[f] Cycraft Technology, Taipei, Taiwan

ARTICLE INFO

Article history:
Received 2 December 2022
Revised 10 February 2023
Accepted 23 March 2023
Available online 24 March 2023

Keywords:
Federated Learning
Intrusion Detection
Poisoning Attack
Backdoor Attack
Local Outlier Factor

ABSTRACT

The Machine Learning-based Intrusion Detection System (ML-IDS) becomes more popular because it doesn't need to manually update the rules and can recognize variants better, However, due to the data privacy issue in ML-IDS, the Federated Learning-based IDS (FL-IDS) was proposed. In each round of federated learning, each participant first trains its local model and sends the model's weights to the global server, which then aggregates the received weights and distributes the aggregated global model to participants. An attacker will use poisoning attacks, including label-flipping attacks and backdoor attacks, to directly generate a malicious local model and indirectly pollute the global model. Currently, a few studies defend against poisoning attacks, but they only discuss label-flipping attacks in the image field. Therefore, we propose a two-phase defense mechanism, called Defending Poisoning Attacks in Federated Learning (DPA-FL), applied to intrusion detection. The first phase employs relative differences to quickly compare weights between participants because the local models of attackers and benign participants are quite different. The second phase tests the aggregated model with the dataset and tries to find the attackers when its accuracy is low. Experiment results show that DPA-FL can reach 96.5% accuracy in defending against poisoning attacks. Compared with other defense mechanisms, DPA-FL can improve F1-score by 20∼64% under backdoor attacks. Also, DPA-FL can exclude the attackers within twelve rounds when the attackers are few.

1. Introduction

An Intrusion Detection System (IDS), which is a real-time network monitoring and defense system, will trigger alerts to administrators once suspicious or malicious activities are discovered. In general, there are two types of IDS, signature-based and anomaly-based detection, where the former utilizes signatures while the latter adopts rules to identify intrusions. However, these predefined signatures and rules require manual updates by experts. In an increasingly complex network, it is impossible to keep pace with the evolution of quirky intrusions.

To resolve the shortcomings of traditional IDSs, Machine Learning (ML) has been utilized to detect intrusions. Currently, many studies have employed Machine Learning-based Intrusion Detection Systems (ML-IDS) to verify the feasibility of such approaches. Although ML-IDS excels at intrusion detection, it requires gathering a copious amount of network data to be trained in a centralized ML server. Unfortunately, some private data are privacy-concerned. Thus adopting an ML-IDS approach is not suitable for use in sensitive or confidential environments where data leakage will generate serious security and privacy issues (Rey et al., 2022).

Federated Learning (FL) resolves the data privacy issue without the necessity of storing data on a centralized server in ML training. In each round of a Federated Learning-based Intrusion Detection System (FL-IDS) (Al-Marri et al., 2020, Nguyen et al., 2019), each participant first trains its local model and sends the model's weights to the global server, which then aggregates the received weights and distributes the aggregated global model to participants. Therefore, FL-IDS has the advantage of cooperative learning to obtain a better model by using a large amount of data. Also,

* Corresponding author.
E-mail address: laiyc@cs.ntust.edu.tw (Y.-C. Lai).

https://doi.org/10.1016/j.cose.2023.103205
0167-4048/

it does not require centralizing data and sharing data, resulting in the protection of data privacy.

Both ML-IDS and FL-IDS systems are prone to adversarial attacks (Papernot et al., 2016, Biggio et al., 2013, Biggio et al., 2012, Xiao et al., 2015, Wang et al., 2019), which create adversarial samples to interfere with machine learning, resulting in an inaccurate model. The most common types of adversarial attacks are evasion attacks (Biggio et al., 2013) and poisoning attacks (Biggio et al., 2012). The former evades detection by altering the samples, but it does not pollute the training data itself. Poisoning attacks, such as label-flipping attacks (Xiao et al., 2015) and backdoor attacks (Wang et al., 2019), directly pollute the training data to generate a corrupted model. The label-flipping attacks flip the labels in training data while backdoor attacks place some hidden triggering conditions in given training targets.

Although ML-IDS and FL-IDS both are susceptible to poisoning attacks, there are some significant differences between them: (1) The FL global model, which will be affected by weights uploaded from malicious participants (called attackers sometimes in this paper; they are interchangeable), damages other benign local models. Therefore, in FL, a participant must not only defend against the attacks from its model but also defend against the attacks from other local models, which will affect the global model. On the other hand, the ML model is only affected by its training data. (2) The FL global server only receives weights uploaded by local models, so it cannot perceive the original data and suspect data using some data analysis techniques, resulting in more difficulty in intrusion detection. (3) There are usually a lot of participants in FL-IDS, so it is more difficult to screen many participants, making them susceptible to being attacked. (4) FL can detect attackers during each round and exclude them from the system while ML can only detect them after the training process has been finished.

Currently, there is a little research on poisoning attacks and defense in the field of FL. The research focusing on attacks mainly discussed attack methods and demonstrated the effects of poisoning attacks on FL (Zhang et al., 2019, Zhang et al., 2020, Bagdasaryan et al., 2020, Zhou et al., 2021, Nguyen et al., 2020). The research focusing on defense mainly discussed how to defend against label-flipping attacks (Short et al., 2020, Liu et al., 2021, Doku and Rawat, 2021, Huang et al., 2021, Chen et al., 2020, Zhou et al). Most of them relied on utilizing the overall accuracy of testing data to determine whether the FL is under attack. However, these methods overlook backdoor attacks, which do not affect overall accuracy and thus are difficult to be detected. The research in (Gu and Yang, 2021) defended against both label-flipping and backdoor attacks, but it only focused on the image field. Also, it used compression and decompression methods, which are not directly applicable to IDS.

A defense mechanism against poisoning attacks in FL can be handled in participants or in the global server. The former is similar to defending against poisoning attacks in ML models, so the latter is our focus. In this paper, we propose the Defense against Poisoning Attacks in Federated Learning (DPA-FL) method, which adopts a two-phase mechanism, to defend the global model against poisoning attacks. The first phase, called the Relative Phase (RP), screens for possible attackers using the relative differences between the weights of attackers and those of benign participants. The second phase, called Absolute Phase (AP), conducts accuracy testing on a small-size dataset. When the model's accuracy is too low, it is very likely to be attacked by attackers, so AP can utilize the accuracy to determine whether any attacker contributes to this global model. Since testing a dataset takes much time even for a small-size dataset, AP usually takes more time but can obtain better classification accuracy than RP. Therefore, using such a two-phase method not only enhances classification accuracy but also raises detecting efficiency.

Specifically, our proposed DPA-FL is composed of RP and AP. RP uses the local outlier factor (LOF) to calculate the deviation of weights for each local model. The second quartile is used for classifying benign participants, while the third quartile distance and the interquartile range (IQR) are used to screen attackers. After excluding some obvious participants, including benign participants and attackers, the remaining uncertain participants are tested in AP. Since backdoor attacks will not affect overall accuracy, they cannot be detected with testing by using the original dataset. Therefore, we insert some data which suffers from backdoor attacks into the testing dataset and conduct model testing with this new backdoor-embedded dataset to especially defend against backdoor attacks. At the same time, AP employs reinforcement learning (RL) to adjust the screening threshold and utilizes the dichotomy method to screen a set of participants which include attackers.

The contribution of this paper is as follows. First, this paper investigates how to detect poisoning attacks, including label-flipping attacks and backdoor attacks, in the FL-IDS, unlike other studies which only focus on label-flipping attacks or focus on the ML-IDS. Second, this paper proposes a two-phase method, DPA-FL, which utilizes both relative comparison and absolute accuracy to quickly reduce the damage caused by poisoning attacks. Third, this paper compares DPA-FL with other defense approaches under different scenarios, investigates their performance, and verifies the outperformance of DPA-FL.

This paper is organized as follows. Section 2 introduces some background of FL and poisoning attacks, as well as related works. Section 3 describes the system model and gives the problem statement. Section 4 describes the concept of DPA-FL and its detailed algorithms. The experiments and results are given in Section 5. Finally, Section 6 gives the conclusions and some future work.

2. Background

In this section, we first describe federated learning and then move to poisoning attacks. Afterward, we provide a detailed review of related work to show their differences from this paper. Finally, some techniques, including local outlier factor and reinforcement learning used in this study, are introduced.

2.1. Federated Learning

Federated Learning (FL) was developed to guarantee that data is stored within local devices for privacy while allowing for cooperative learning (Yang et al., 2019). A FL framework usually consists of two entities; some participants and a global server. Let $\mathbb{N} = \{1, \ldots, N\}$ represent the set of N participants, where participant $i \in \mathbb{N}$ has its private dataset D_i, utilizes this dataset to train a local model $\mathbf{w}_i$, and uploads the resulting weights to the global server. The global server then aggregates all of the received weights, $\mathbf{w} = \cup_{i\in\mathbb{N}}\mathbf{w}_i$, into a global model $\mathbf{w}_G$. Here, local models refer to models trained by participants, while the global model refers to the model aggregated by the global server. The FL training process usually consists of the following three steps:

Step 1 (Task initialization): The global server first specifies the global model used and the training weights, and then broadcasts the initialized global model weights $\mathbf{w}_G^0$ to the given participants where the superscript represents the round number and a value of 0 represents the initial round.

Step 2 (Training for local model): After receiving global model $\mathbf{w}_G^{t-1}$, each participant utilizes its local data to do training and further updates the local model weight $\mathbf{w}_i^t$ to minimize the loss function $L(\mathbf{w}_i^t)$ as

$$\mathbf{w}_i^{t*} = \arg\min_{\mathbf{w}_i^t} L\left(\mathbf{w}_i^t\right). \quad (1)$$

The updated local model's weights will be sent back to the global server.

Step 3 (Aggregation for global model): The global server aggregates local models' weights from participants and then returns the updated global model weight $\mathbf{w}_G^t$ to participants. The loss function $L(\mathbf{w}_G^t)$ for the global server is as

$$L\left(\mathbf{w}_G^t\right) = \frac{1}{N}\sum_{i=1}^{N} L\left(\mathbf{w}_i^t\right). \tag{2}$$

Steps 2 to 3 are repeated until the global loss function converges or reaches the desired accuracy.

In the FL training process, the aggregation of the global model is a key operation. To minimize the global loss function in (2), a commonly used method is FedAVG, which averages the shared weights of different participants (Nilsson et al., 2018), as

$$\mathbf{w}_G^t = \frac{\sum_{i\in\mathbb{N}} |D_i|\mathbf{w}_i^t}{\sum_{i\in\mathbb{N}} |D_i|}, \tag{3}$$

where $|D_i|$ denotes the amount of data in $|D_i|$.

2.2. Poisoning Attacks

Poisoning attacks can be classified into two categories, label-flipping attacks and backdoor attacks, based on their approaches. Label-flipping attacks flip the labels of the original training samples (Xiao et al., 2015), causing the trained model to deviate from its original detection boundary and resulting in detecting errors. Backdoor attacks utilize "hidden triggers" to train the model (Wang et al., 2019). Since backdoor attacks expand the original dataset to plant hidden triggers, the accuracy of detecting original data is still preserved. That is, the detection accuracy of the entire model is not affected. Errors only appear when certain targets are triggered.

A poisoning attack is usually composed of two steps:

Step 1 (poisoned data generation): The attacker generates poisoned data based on the given target. To generate a label-flipping attack, the attacker obtains a sample x_j (the j-th sample) from the training dataset datasets D. The original label y_j is flipped to become the erroneous one y_j', as

$$D_j^P = \left\{\left(x_j, y_j^{'}\right)\right\},\ j \in 1,\ \ldots,\ |D|, \tag{4}$$

where $|D|$ represents the number of data in training set D.

On the other hand, a backdoor attack copies or emulates a sample x_j from the training dataset D and inserts a hidden trigger L for this sample. The label of this sample is then changed into the attack target y_j' to generate a backdoor sample as

$$D_j^P = \left\{\left(L\left(x_j\right), y_j^{'}\right)\right\},\ j \in 1,\ \ldots, |D|. \tag{5}$$

Step 2 (mixture of malicious data and original data): The poisoned dataset D_j^P is mixed with the original training dataset D, resulting in a corrupted dataset $D^* = D \cup D_j^P,\ j \in 1,\ \ldots,\ |D|$. The attackers can generate the desired amount of malicious samples according to their real needs. This new dataset is then used to train a corrupted model, which cannot defend against poisoning attacks.

2.3. Related Work

Table 1 contains related work, which can be divided into two main categories: attacks (Zhang et al., 2019, Zhang et al., 2020, Bagdasaryan et al., 2020, Zhou et al., 2021, Nguyen et al., 2020) and defense (Short et al., 2020, Liu et al., 2021, Doku and Rawat, 2021, Huang et al., 2021, Chen et al., 2020, Zhou et al, Gu and Yang, 2021) in FL. We summarize them according to the attributes of *domain, environment, technique*, and *approach*. Most papers on FL attack/defense mainly considered the image domain and used datasets like MNIST and CIFAR-10. Few papers discussed the IDS domain and mainly employed open-source datasets like CICIDS2017 (Panigrahi and Borah, 2018). The local classifiers adopted were usually neural networks like Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN). Although most papers focusing on attacks covered backdoor attacks, most papers focusing on defense only discussed label-flipping attacks instead.

Papers focusing on attacks stressed how to increase the attacking successful rate. In order to generate malicious data to be similar to benign ones, many studies utilized Generative Adversarial Network (GAN) to pose as benign participants, obtained sufficient training parameters, and converted them into data used in backdoor or label-flipping attacks (Zhang et al., 2019, Zhang et al., 2020). Because FL is unable to perceive the training process of local models, some works directly replaced local models with the model with backdoor attacks (Bagdasaryan et al., 2020). Because local models in FL are self-trained and self-managed, (Zhou et al., 2021) mentioned the possibility of directly modifying weights to achieve backdoor attacks. However, the above papers only investigated image datasets. In the IDS domain, a study assumed that the participant continues to acquire training data on detection, so the attacker utilizes this characteristic to transfer backdoor data to the participant (Nguyen et al., 2020).

Most papers on defense considered image or character datasets (Short et al., 2020, Liu et al., 2021, Doku and Rawat, 2021, Huang et al., 2021, Gu and Yang, 2021). By employing blockchain technology to verify and evaluate participants, it is possible to defend against label-flipping attacks (Short et al., 2020). D2MIF was proposed to remove label-flipping attackers by conducting accuracy testing on the global server and adopting Isolation Forest (IForest) algorithm to determine model saliency (Liu et al., 2021). The paper in (Doku and Rawat, 2021) constructed a mediator between the global server and participants, as well as analyzed local data to determine whether local models are under label-flipping attacks. Through multiple random recombination and accuracy verification tests, it is possible to separate benign participants and attackers to defend against label-flipping attacks (Huang et al., 2021). The study in (Gu and Yang, 2021) used a Conditional Variational Autoencoder (CVAE) to calculate the errors in model updating and employed the results to determine whether the model is under attack. The only study against poisoning attacks in IDS utilizes the attention mechanism of the Gated Recurrent Unit (GRU) algorithm to calculate the importance of uploaded weights and determine the saliency of local models (Chen et al., 2020).

Although a Differentially Private Federated Learning model (DPFL) focused on the field of image (Zhou et al), it also used accuracy testing and compare the accuracy of different participants to identify attackers. At first glance, it seems a little bit similar to our approach, but there are four main differences: (1) DPFL method requires manually pre-defining and updating important parameters, while ours relies on RL to dynamically do so. (2) DPFL only decreases the influence degrees of attackers, but ours cleans up the attackers completely. (3) DFPL is not particularly designed to defend against backdoor attacks, while ours can defend against both backdoor and label-flipping attacks. (4) The last and most important difference is that DFPL only adopts one phase, but ours adopts

Table 1
Methods of poisoning attacks and defense in federated learning

Type	Paper	Domain	Environment		Technique			
			Dataset	Classifier	Attack	Defense	Phase	Method
Attack	(Zhang et al., 2019)	Image	MNIST, AT&T	CNN	LF	N/A	1	• Generative adversarial networks • Mimicking other participants
	(Zhang et al., 2020)	Image	MNIST, Fashion-MNIST, CIFAR-10	CNN	LF, BD	N/A	1	Generative adversarial networks
	(Bagdasaryan et al., 2020)	Image	CIFAR-10	CNN	BD	N/A	1	Model replacement
	(Zhou et al., 2021)	Image	MNIST, CIFAR-10	CNN	BD	N/A	1	Injecting poisoning neurons
	(Nguyen et al., 2020)	IDS	DÏoT, UNSW, Private dataset	RNN	BBD	N/A	1	Indirect attack
Defense	(Short et al., 2020)	Image	MNIST	SGD	LF	Blockchain	1	Data validation
	(Liu et al., 2021)	Image	MNIST, Fashion-MNIST	CNN	LF	RL, IForest	1	Pre-aggregation
	(Doku and Rawat, 2021)	Text	IMDB	SVM	LF	PoCI	1	Data vetting (mediator)
	(Huang et al., 2021)	Image	MNIST, Fashion-MNIST	CNN	LF	SAGE	1	• Shuffling • Regrouping
	(Chen et al., 2020)	IDS	KDD CUP 99, CICIDS2017, WSN-DS	GRU-SVM	LF	FedAGRU	1	Attention mechanism
	(Zhou et al)	Image	MNIST, Fashion-MNIST, CIFAR–10	ResNet20	LF	Accuracy detection	1	Data testing
	(Gu and Yang, 2021)	Text, Image	Vehicle, Synthetic, MNIST, FEMINIST	BDL	LF, BD	CVAE	1	Comparison of local models' weight
	DPA-FL	IDS	CICIDS2017	CNN	LF, BD	RL, LOF	2	RP: Comparison of local models' weights AP: Data testing

*BDL: Bayesian Deep Learning, LF: Label-flipping, BD: Backdoor, PoCI: Proof of Common Interest, RL: Reinforcement Learning, IForest: Isolation forest, CVAE: Conditional Variational AutoEncoder, SAGE: Shuffling and Regrouping based Defense Framework

two phases to simultaneously take efficiency and accuracy into account.

2.4. Used Techniques

2.4.1. Local Outlier Factor

Local Outlier Factor (LOF) is a detection method based on spatial density anomalies. When a data point is in a dense region, it is regarded as a normal one. On the other hand, when it is in a sparse region, it will be regarded as an outlier. LOF consists of two steps: (1) For each data point, calculate its distances to other data points and sort these distances. (2) Find the neighbors of each data point and calculate its LOF score.

Let the distance between the data point P and O be $d(P, O)$ and the k-distance of O, which means the distance between O and the k-th closest neighbor, be $d_k(O)$. Use these values to calculate the reachability distance between point P and point O, $r_k(P, O)$, as the maximum of the k-distance of O and the distance between P and O, such as

$$r_k(P, O) = \max[d_k(O), d(P, O)]. \tag{6}$$

Next, calculate the Local Reachability Density (LRD) using equation (6) to determine whether a data point is in a dense region. Taking point P as an example. Its LRD can be calculated as

$$LRD_k(P) = \frac{1}{\frac{\sum_{O \in N_k(P)} r_k(P,O)}{|N_k(P)|}}, \tag{7}$$

where $N_k(P)$ represents the first k nearest neighbors of point P.

If the LRD is large, point P is located in a dense region, and vice versa. Finally, we further obtain the LOF by summing up the LRD of all neighbors of P and dividing the average by its LRD, as

$$LOF_k(P) = \frac{\sum_{O \in N_k(P)} LRD_k(O)}{|N_k(P)| LRD_k(P)}. \tag{8}$$

A larger LOF means that the point has less density than its neighbors, so it is more likely to be an outlier. On the contrary, a smaller LOF means the point has a higher density than its neighbors, meaning that it is more likely to be a normal one.

2.4.2. Reinforcement Learning

Supervised learning methods require labeled data for training, but it is impossible to collect such type of data in many situations. RL can solve this problem in training (Sutton and Barto, 1998). In an agent of RL, there are three related parts: actions (a), states (s), and rewards (r), where a is the action chosen by the agent from the set of all actions $\boldsymbol{A}$, s represents the state to denote external environment, and r is the feedback from the environment given the chose action. In RL, a policy is used to instruct the agent on how to proceed with the next action. This is done by calculating the expected value of the cumulative reward received by conducting a certain action.

Q-Learning, Deep Q-learning Network (DQN) (Mnih et al., 2013), and Twin Delayed Deep Deterministic Policy Gradient (TD3) (Thrun and Schwartz, 1993) are typical methods of RL algorithms. Q-Learning records learned policies to determine which action to take. That is, each state, action, and corresponding reward are recorded in Q-table for future reference. However, it is very difficult to construct Q-table in a complicated environment. To solve this problem, DQN uses a neural network, which inputs a state and outputs a Q value, to replace the Q-table. Also to increase efficiency, DQN adds an experience pool, so the policies can be stored and repeatedly used.

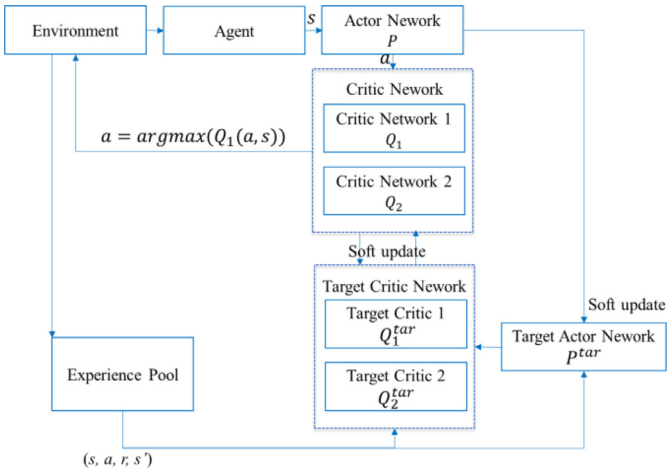

Fig. 1. TD3 neural network architecture

However, using a neural network to estimate the Q values may cause its overestimation, resulting in the instability of the model. Especially in an environment that requires continuous actions, which may generate cascading errors, TD3 was proposed to solve this problem (Thrun and Schwartz, 1993). Figure 1 depicts the architecture of TD3, which is composed of six networks: Actor Network: $P(s;u)$, Target Actor Network: $P^{tar}(s;u')$, Critic Network 1: $Q_1(a,\ s;w_1)$, Target Critic Network 1: $Q_1^{tar}(a,\ s;w_1{}')$, Critic Network 2: $Q_2(a,\ s;w_2)$, and Target Critic network 2: $Q_2^{tar}(a,\ s;w_2{}')$. The variables before and after the semicolon represent input and neural network parameters, respectively. $P(s;u)$ is responsible for updating the parameters of the policy network. It chooses an action a from the possible action set A depending on the current state s and calculates the expected reward r and the next state s'. $P^{tar}(s;u')$ calculates the best action a' given s based on the experience pool. Its parameters are periodically updated from $P(s;u)$. $Q_1(a,\ s;w_1)$ and $Q_2(a,\ s;w_2)$ calculate the current Q values and provide iterative updates. $Q_1^{tar}(a,\ s;w_1{}')$ and $Q_2^{tar}(a,\ s;w_2{}')$ calculate the target Q-values, which are updated periodically from $Q_1(a,\ s;w_1)$ and $Q_2(a,\ s;w_2)$ and eventually recorded into the experience pool.

In brief, TD3 utilizes double critic networks to prevent Q values from being overestimated. That is, the calculation is conducted using the smaller Q value from two networks.

3. System Model and Problem Statement

This section describes the system model and gives a formal problem statement.

3.1. System Model

The system architecture is shown in Fig. 2. This paper mainly investigates how the global server defends against poisoning attacks initiated by participants. There are a total of N participants, including a few attackers. Each participant i has his own training data D_i. Attacks are initiated by the attacker i by polluting dataset D_i. All participants train their own local models using the same machine learning method ml. The weight $\mathbf{w}_i^t$ for participant i is uploaded to the global server, where t represents the t-th round. After receiving all $\mathbf{w}_i^t$ from all participants, the global server utilizes DPA-FL to determine the set of benign participants, **B**, and the set of attackers, **M**. It then aggregates the weights of all the benign models into the global model $\mathbf{w}_G^t$ and distributes the model back to benign participants for further training.

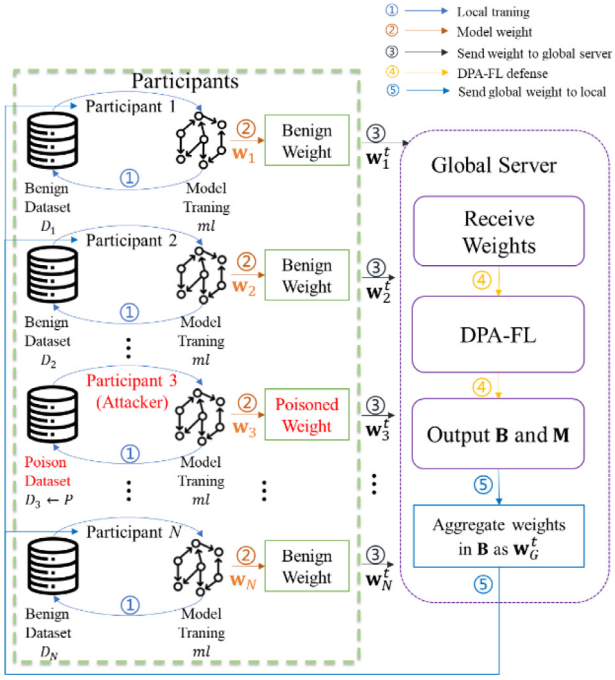

Fig. 2. System architecture

Table 2 lists notations used in our study and is organized into three categories: *participant, model,* and *attack/defense*. We will describe the meaning of notations when they are used.

3.2. Problem Description

The goal of this study is to differentiate benign participants from attackers in the global server to mitigate the damage of poisoning attacks by preventing pollution to the global model, which will propagate to other benign participants. The formal problem statement is defined as follows.

Given:

(1) FL parameters: the participant set $\mathbb{N}$ and the number of participants N, the machine model ml, and the dataset of the i-th participants D_i $(1 \le i \le N)$.
(2) Poisoning attack P and the stability threshold α.

Output: the set of benign participants, **B,** and the set of attackers, **M**.

Objective: maximize the F1-Score, *F1*, on classifying the network traffic.

Assumptions:

(1) The datasets of all participants are Independent and Identically Distributed (IID).
(2) The global server is trusted. Since attackers only indirectly affect the global model by uploading their poisoned local models, they cannot seize the control of the global server.
(3) Attackers can spontaneously alter their training samples since they completely control their models and arbitrarily access/modify the data.

4. Defending Poisoning Attacks in Federated Learning

We propose a defense mechanism called DPA-FL to mitigate the damage of poisoning attacks, including label-flipping and backdoor attacks, in FL-IDS. This section first illustrates the overview of DPA-FL and then describes its two phases: relative phase and absolute phase.

Table 2
Notation table

Categories	Notations	Descriptions	Property
Participant	$\mathbb{N}$	The participant set $\mathbb{N} = \{1, \ldots, N\}$ where N is the number of participants.	Input
	D_i	Dataset of the i-th participant	Input
	ml	Local machine learning method	Input
Model	$\mathbf{w}_G$	Global model	Variable
	$\mathbf{w}_i$	The i-th local model (the i-th participant)	Variable
	$\mathbf{w}_{i,\ j}$	The j-th weight in the i-th local model	Variable
	$\mathbf{w}_i^t(\mathbf{w}_G^t)$	$\mathbf{w}_i(\mathbf{w}_G)$ in the t-th round	Variable
Attack/	P	Poisoning attack	Input
Defense	α	Stability threshold used in RP and AP	Input
	$\mathbf{U},\ \mathbf{B},\ \mathbf{M}$	Uncertain, Benign, and malicious sets of participants, respectively	Output
	s_i	Anomaly score (LOF) for the i-th participant (used in RP)	Variable
	x_i	benignity/maliciousness count of the i-th participant	Variable
	D^E	Dataset for absolute phase	Variable
	$A^B,\ A^M,\ A^U$	Accuracy for benign, malicious, and uncertain sets of participants, respectively (used in AP)	Variable
	θ	The threshold to judge benign participants and is adjusted with RL (used in AP)	Variable
	$F1$	F1-score	Result

4.1. DPA-FL Overview

DPA-FL classifies participants into three sorted sets: benign participants **B**; malicious participants (attackers) **M**, and uncertain participants **U**. In the beginning, both **B** and **M** are empty sets, i.e., **B**=**M**=∅, while **U** is the set of all participants. Participants in each set are ordered according to their participants' identifiers. After each round, DPA-FL will move confirmed participants to **B** or **M**. DPF-FL will execute many rounds until **U** becomes an empty set. Note that although in each round the global server will receive the weights $\mathbf{w} = [\mathbf{w}_1,\ \mathbf{w}_2,\ \ldots,\ \mathbf{w}_N]$ from all participants, it only processes those participants in the set **U** because others have been confirmed. To move participants in **U** to **B** or **M**, DPA-FL will conduct two-phase operations: the relative phase (RP) compares the local models of these participants while the absolute phase (AP) conducts accuracy testing on the dataset to determine whether they are benign/malicious or not.

In RP, although participants provide different model's weights, the model's weights of an attacker usually differ from those of benign participants. Since the number of benign participants is usually much more than the number of attackers, RP uses LOF to calculate the anomaly score and adopts this value to determine whether a participant belongs to the benign, malicious, or still uncertain category. Once RP is complete, AP will provide a more accurate screening. As mentioned in subsection 2.2, label-flipping attacks do affect overall accuracy, but backdoor attacks only affect the accuracy of specific targets. If the testing dataset of this phase lacks backdoor data, only label-flipping attacks can be detected. Thus randomly generated backdoor data is inserted into the testing dataset, so conducting testing can detect the backdoor attacks. When the accuracy is higher than a given threshold, which will be adjusted by RL, the model is not under attack. On the other hand, when the accuracy is lower than this threshold, AP will further find the attacker(s) using the dichotomy.

The purpose of adopting a two-phase approach is that RP can quickly determine whether there is a possibility of malicious/benign participants, but only for those with significant differences (with obviously high/low LOF anomaly scores). However, since some participants have middle LOF anomaly scores, directly classifying them may cause misclassification. Therefore, AP is used to accurately determine whether a poisoning attack appears using accuracy testing, but it takes a longer time. Therefore, DPA-FL uses RP to quickly screen the participants that can be clearly classified, and then passes the uncertain participants to AP for further data testing to confirm them. Thus DPA-FL can obtain both high accuracy and efficiency.

Note although DPA-FL is a two-phase approach, the relationship of RP and AP is a many-to-one mapping, rather than a one-to-one mapping. That is, DPA-FL repeats RP many times and conducts AP once because RP consumes significantly less time than AP. Thus, in each round, DPA-FL will repeat RP until no participant can be further classified and then AP is executed. The overall process of DPA-FL continues many rounds until all participants have been classified or the convergence is reached.

4.2. Relative phase

Fig. 3 depicts the flow chart of RP. The participants in **U** can be defined as $\mathbf{U}=[u_1, u_2, \ldots, u_{|\mathbf{U}|}]$, where u_i represents the i-th participant in **U**, and $|\mathbf{U}|$ represents the number of elements in **U**. RP will calculate the anomaly score $\mathbf{S}=[s_1, s_2, \ldots, s_{|\mathbf{U}|}]$ of each participant in **U**, where s_i is the anomaly score of the i-th participant in **U**. Once all anomaly scores are calculated, RP will classify each participant in **U** into **B, M** or still **U**.

The anomaly score of the local model i, $\mathbf{w}_i$, is calculated by using LOF. However, $\mathbf{w}_i$ is a vector that includes multiple weights, i.e., $\mathbf{w}_i = [w_{i,\ 1},\ w_{i,\ 2},\ \ldots,\ w_{i,|\mathbf{w}_i|}]$, where $w_{i,j}$ denote the j-th weight of the i-th local model and $|\mathbf{w}_i|$ is the number of weights in $\mathbf{w}_i$, Here, instead of directly using Euclidian distance to calculate LOF, each weight is considered individually. This is because we observed a fact that the model generated by an attacker usually has more anomalous weights, rather than the case that some weights have large differences. Thus, we calculate the anomaly score of $\mathbf{w}_i$ as

$$F(\mathbf{w}_i) = \sum_{j=1}^{|\mathbf{w}_i|} O\left(w_{i,j}\right), \tag{9}$$

$$O\left(w_{i,j}\right) = \begin{cases} 1, & \text{if } LOF_k\left(w_{i,j}\right) > 1, \\ 0, & \text{if } LOF_k\left(w_{i,j}\right) \leq 1. \end{cases} \tag{10}$$

The LOF function is from Equation 8. Here we set k as 20 (Breunig et al., 2000) and the threshold for the LOF score as 1 (Auskalnis et al., 2017).

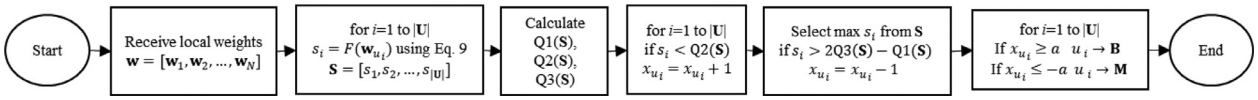

Fig. 3. Flowchart in relative phase

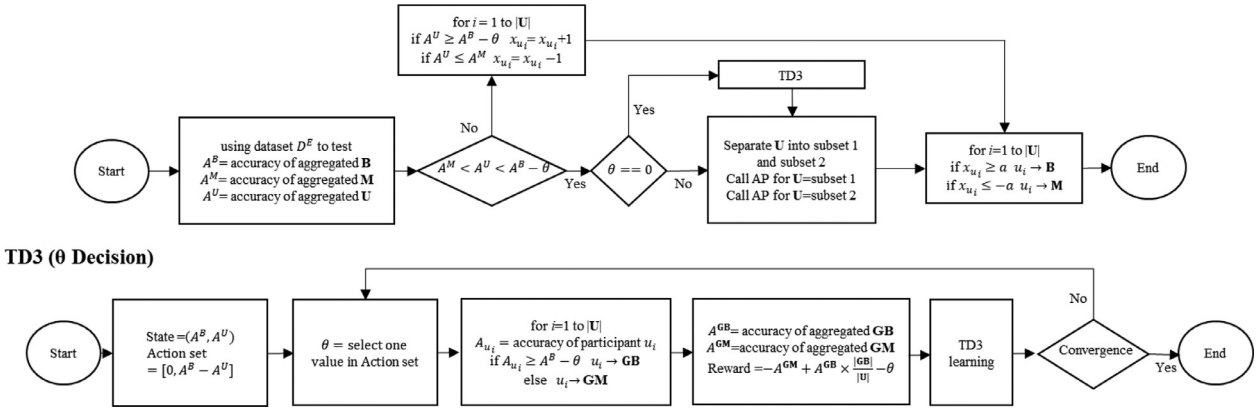

Fig. 4. Flowchart in absolute phase

Thus, for each participant in **U**, we can obtain its corresponding anomaly score by using $s_i = F(\mathbf{w}_{u_i})$ to create the anomaly score set $\mathbf{S}=[s_1, s_2, \ldots, s_{|\mathbf{U}|}]$. Then as (Ma et al., 2022), we calculate quartiles in **S**. Let Q1(**S**), Q2(**S**) and Q3(**S**) represent the first, second and third quartiles in **S**, respectively. Consider three facts: (1) attackers occupy a small portion; (2) the anomaly score of an attacker is usually larger than that of benign ones; (3) anomaly scores for benign participants are usually similar. Thus the median Q2(**S**) is the threshold to classify the benign participants, that is, the participants with anomaly scores below this value are regarded as benign ones. To judge the attackers, only the participant with the highest anomaly score is considered. Also the sum of the third quartile Q3(**S**) and the interquartile range IQR(**S**)=Q3(**S**)-Q1(**S**) as a tougher threshold for attackers to prevent misclassification.

However, even though some participants are classified as benign and malicious ones, RP will not directly move these participants from **U** into **B** and **M** because of the following reason. Since all of the models' weights are updated and varied during each round. In order to avoid misclassification caused by such variations, the classification should be based on a stable state. Thus, we use x_i to record the benignity/maliciousness count of participant i, and set 0 as its initial value. In a round, if participant i is defined as a benign one, x_i is incremented by one, while if it is a malicious one, x_i is decremented by one. Once x_i is greater than a given threshold, α, there is sufficient evidence to show its benignity, so now participant i is confirmed as a benign one and moved from **U** to **B**. On the contrary, if x_i is less than $-\alpha$, participant i will be confirmed as an attacker and moved from **U** to **M**.

4.3. Absolute phase

Fig. 4 is the flowchart of AP, which only screens benign participants and attackers in the set of uncertain participants $\mathbf{U}=[u_1, u_2, \ldots, u_{|\mathbf{U}|}]$. Thus, AP first aggregates the models in **U, B**, and **M**, and uses the dataset D^E to test these aggregated models to obtain the corresponding accuracy A^U, A^B, and A^M. When a special case of $\mathbf{B}=\emptyset$ happens, A^B is set as 1. Similarly, A^M is set as 0 when a special case of $\mathbf{M}=\emptyset$ appears.

A^M and A^B provide a useful guide to judge whether some attackers exist in the set of **U**. Observing that **U** usually includes more benign participants and fewer attackers, so A^U should be closer to A^B and far away A^M. Thus A^M is directly set as the malicious threshold because if **U** contains some benign participants, its accuracy A^U should not be lower than the accuracy A^M. That is, when $A^U \leq A^M$, all participants in **U** are regarded as attackers, i.e., each x_{u_i} is decremented by one. On the other hand, directly using A^B as the benign threshold is not proper. The reasons are based on two observations: (1) **U** usually includes many benign participants and few attackers, causing that A^U might be very close to A^B. (2) The aggregated model and its accuracy will vary in each round. Therefore, it is necessary to adopt a tolerance θ to prevent misclassification. Thus $A^B - \theta$ is set as the benign threshold. That is, when $A^U \geq A^B - \theta$, all participants in **U** are regarded as benign ones, i.e., each x_{u_i} is increased by one. Finally, when $A^M < A^U < A^B - \theta$, the set **U** is most likely to contain both benign participants and attackers. In this case, AP divides **U** into two subsets through the dichotomy and recursively repeats the same procedure for these two subsets to classify the participant in **U**. Finally, each participant u_i in **U** can be classified into **B, M,** or **U** according to its x_{u_i}.

Two things need to handle in AP: creating dataset D^E and determining θ. About dataset D^E, in order to detect backdoor attacks, we sample the original dataset, generate backdoor attack data using Equation (5), and insert the poisoned data into the original dataset, allowing DPA-FL to defend against backdoor attacks.

The tolerance θ is an important parameter and is determined by using TD3. The state in TD3 is (A^U, A^B) while the action is adjusting (increasing or decreasing) θ. For higher efficiency, the range of adjustment is limited. If θ is less than 0, the benign threshold, $A^B - \theta$, will be higher than A^B. This case is unreasonable and will result in serious misclassification of benign participants. On the other hand, if θ is larger than $A^B - A^U$, the benign threshold will be lower than A^U and this case is also unreasonable. Therefore, the TD3 action is limited to set θ ranging between 0 and $A^B - A^U$. According to the adjusted θ and the accuracy of participants in **U**, we partition **U** into two subsets, **GB**, and **GM**, where **GB** includes the participants with accuracy better than to $A^B - \theta$ in **U** while others in **U** belong to **GM**. The accuracy of participants exist in **GB** and **GM** is calculated as A^{GB} and A^{GM}, respectively. Finally, the TD3 uses the following equation as its reward, as

$$r = -A^{GM} + A^{GB} \times \frac{|\mathbf{GB}|}{|\mathbf{U}|} - \theta, \tag{11}$$

where |**GB**| is the number of participants in **GB** and |**U**| is the number of participants in **U**.

We expect that the accuracy of benign participants is high and the accuracy of attackers is low. Thus the reward is set as adding A^{GB} and subtracting A^{GM}. However, to avoid wrongly classifying the attackers, the influence degree of A^{GM} should be higher and set as 1 while the influence degree of A^{GB} is according to the percentage of benign participants in **GB**. Also if θ is not considered, the reward would be the same regardless of this value, making it impossible to find the best tolerance which is the closest border to classify benign participants.

Through this reward setting, TD3 finds the tolerance θ which is the closest border that does not cause misclassification of benign participants. Finally, this step in TD3 will be repeated several times until the reward reaches convergence. Considering that the TD3 process is time-consuming and the accuracy difference of benign participants is not significant, once θ is decided in a round, it will be fixed and don't need to be recalculated by TD3 in future rounds.

4.4. Example

An example is provided to illustrate the complete operation of DPA-FL. Assuming there are 12 participants, including one attacker (No. 1) and 11 benign participants (No. 2-12). The stability threshold value α is 1 in this example for simplification although it will be set larger in a realistic environment.

In the first round, since all participants have not been classified, **U** is equivalent to set $\mathbb{N}$, i.e., **U**=$\mathbb{N}$= [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. On the other hand, **B** and **M** are empty sets. In RP, LOF will be used to calculate the anomaly score **S**= [11, 7, 6, 7, 5, 4, 5, 6, 7, 9, 9, 8] of all participants in **U**. By calculating the quartiles $Q1(\mathbf{S}) = 5.5$, $Q2(\mathbf{S}) = 7$, $Q3(\mathbf{S}) = 8.5$, we define the benign threshold as $Q2(\mathbf{S}) = 7$, and malicious threshold as $2 \times Q3(\mathbf{S}) - Q1(\mathbf{S}) = 11.5$. Here, five participants 3, 5, 6, 7, and 8 have anomaly scores 5, 5, 4, 5, and 6, respectively, which are less than the benign threshold 7, so their corresponding labels x_3, x_5, x_6, x_7, x_8 will be incremented by one. On the other hand, participant 1 with the largest anomaly score, 11, does not reach the malicious threshold, 11.5, meaning that no participant will be labeled as malicious. Thus, since stability threshold α is 1, RP will move five benign participants to **B** and no participant to **M**, i.e., **B**=[3, 5, 6, 7, 8] and **M**=$\emptyset$.

In the beginning of AP, **M**=$\emptyset$, **B**=[3, 5, 6, 7, 8] and **U**=[1, 2, 4, 9, 10, 11, 12]. The three groups will be separately aggregated and tested for accuracy. Assume we obtain $A^B = 0.92$, $A^U = 0.88$, and $A^M = 0$ because of **M**=$\emptyset$. To further classify participants in **U**, we use the RL algorithm TD3 to determine θ according to the state (0.88, 0.92), with the action of adjusting the range locating between 0 and 0.04 ($A^B - A^U$). Assuming that the action is setting θ as 0.03, malicious participant **GM**=[1] is successfully screened and its accuracy A^{GM} as 0.84, and benign participants **GB**=[2, 4, 9, 10, 11, 12] are also screened out and its accuracy as 0.92. Then the reward calculation is -0.84+(0.92 $\times$ 6/7)-0.03=-0.0814. When the action is setting θ as 0.02, the attackers and benign participants same as that in the case of θ as 0.03 are also screened out and the reward can be calculated as -0.84+(0.92 $\times$ 6/7)-0.02=-0.0714. Further, if the θ is set to 0.01, the classification results are the same, but the reward is -0.84+(0.92 $\times$ 6/7)-0.01=-0.0614. Here assume that the final θ generated by TD3 is 0.01.

Since $A^M < A^U < A^B - \theta$ $(0 < 0.88 < 0.92 - 0.01)$, AP cannot directly determine the participants in **U** to belong to **B** and **M**, AP will split the participants in **U** by dichotomy. For example, splitting **U** into two subsets [1, 2, 4] and [9, 10, 11, 12], and repeating the same procedure in AP for these two subsets. Assume finally that the attacker (No. 1) is successfully screened out, so the value of x_1 is decreased by one. Others are classified as benign participants, so $x_2, x_4, x_9, x_{10}, x_{11}$, and x_{12} are increased by one. As the stability threshold α is 1 in this example, AP further moves six benign participants [2, 4, 9, 10, 11, 12] to **B** and one participant [1] to **M**. Finally, we obtain **B**=[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12] and **M**=[1].

Table 3
Experimental data structure

Type			
Benign	80.3%	DoS GoldenEye	0.4%
DoS Hulk	8.2%	FTP-Patator	0.4%
PortScan	5.6%	SSH-Patator	0.3%
DDoS	4.5%	Other	0.3%
Category			
Training	60%	Absolute-phase testing	20%
Testing	20%		

5. Experiments

This section conducts some experiments to compare DPA-FL with other defense mechanisms. It also discusses the effect of different parameters on DPA-FL.

5.1. Experiment Setting

5.1.1. Dataset

Experimental data were obtained from the CICIDS2017 dataset compiled by the Canadian Institute of Network Security (Panigrahi and Borah, 2018). This dataset includes 15 types of network traffic, such as Benign, PortScan, and DDoS, and 79 different features, such as traffic variation, temporal continuity, and traffic type. However, the scarcity of some types makes them unsuited to machine learning which needs a large amount of data, so they are combined into a category named "Other". The rest remains unchanged, composing a total of 8 categories, as shown in Table 3. We partition the dataset into training data, testing data in AP, i.e., D^E, and testing data on measuring performance using a ratio of 60%:20%:20%. The training data will be equally distributed among all participants using the method in (Chen et al., 2020).

Although our used dataset is imbalanced with 80% of benign samples, we don't extra execute any balancing process because the amount of attacks is still enough for well-training. Most of the literature did not do data balancing specifically for datasets. For example, (Chen et al., 2020) also used the CICIDS2017 dataset for training and testing, but only split the dataset equally without balancing the attack data with the benign data. Other studies in (Preuveneers et al., 2018, Neto et al., 2022), which focused on feature deletion and normalization, also did not execute any balancing process.

5.1.2. FL-IDS Model

The experiment architecture includes a global server, which uses the FedAVG algorithm for aggregation, and 12 participants, as shown in Table 4. Our proposed DPA-FL is installed in the global server to defend against poisoning attacks. The participants train their models using CNN which consists of two convolutional layers and one fully connected layer. The convolutional layers have 3 $\times$ 3 kernels with a 20% dropout ratio and a 2 $\times$ 2 max pooling layer.

The stability threshold, α, between 3 to 5 is suggested from our experiment results, although they are not shown here. Thus, in the experiment, the default setting of this parameter is 3.

5.1.3. Attack

To verify the capability of DPA-FL against poisoning attacks, the experiment uses two label-flipping attacks and one backdoor attack. (1) Label-flipping (Random): The attacker takes a portion of

Table 4
Model parameters

Model			
Learning algorithm Model Parameters	CNN	Aggregation algorithm	FedAVG
Local model N Batch size Maximum round	12 256 20	Learning rate Epoch	0.001 5
Defense Parameter			
α	3		

the data from his own training data to poison it, and then replaces the original data with the poisoned data in the training dataset. Random flipping utilizes random numbers to generate erroneous labels from the sampled data without altering features. (2) Label-flipping (PGD): The attacker uses the Projected Gradient Descent (PGD) algorithm (Madry et al., 2017) to alter the labels. (3) Backdoor: The attacker embeds backdoors that trigger in the "destination port" feature, causing all poisoned data with port 1200 to be misclassified as benign traffic. Common backdoor attacks bury customized triggers into training data features. In the image domain, a typical approach is embedding tiny special shapes or symbols into the normal picture (Wang et al., 2019, Zhang et al., 2020). In the IDS domain, the time-to-live (TTL) value is used as a trigger (Bachl et al., 2019). Following the same concept and without losing any generalization, we choose "destination port" as the feature to bury the triggers in our experiment because some network protocols do not have the TTL field.

Since most participants are benign and poisoned data usually occupy 20% to 40% of all data (Short et al., 2020, Liu et al., 2021, Chen et al., 2020, Zhou et al), we assume that only one attacker exists and the poisoned data is 30% of the training data.

5.1.4. Competitors and Performance Metrics

To evaluate the performance of DPA-FL, we compare it with three other mechanisms: (1) FedAVG: This is a baseline without any defense. (2) D2MIF (Liu et al., 2021): This method is based on the accuracy testing on the dataset. Once this accuracy is lower than that of the original model, the IForest algorithm will be used to calculate the score of each participant. It also uses RL to modify a screening threshold. When the participant's score is lower than this threshold, it will be blocked from the current aggregation and permanently removed after three blockings. (3) DPFL (Zhou et al): This method evaluates the models' weights of participants using accuracy testing. If the anomaly score of a participant exceeds a threshold, its influence degree on aggregation is decreased.

To evaluate the performance of the above mechanisms and DPA-FL, we adopt three performance metrics: (1) F1-Score (Participant): This is used to investigate the classification of participants. (2) F1-Score (Traffic): This is used to determine whether each defense method is able to correctly classify begin and malicious traffic. (3) Detection round: This is the total number of rounds needed to detect the attackers. The first two are the performance metric about accuracy while the last is the performance metric about efficiency.

The results are averaged by conducting 10 experiments, which randomly partition the dataset and distribute the partitioned dataset among participants.

5.2. Dynamics Observation

This experiment compares DPA-FL with other defense mechanisms against poisoning attacks. In order to understand the accuracy of each mechanism in each round, Fig. 5 shows the dynamics of accuracy in classifying traffic for DPA-FL, FedAVG, D2MIF, and DPFL. There are three insightful observations.

First, for all mechanisms except FedAVG, more rounds will generate higher accuracy in classifying traffic. In the first few rounds, these methods do not have enough knowledge to differentiate the benign participants from attackers, so their accuracy will be lower. However, as the rounds pass, these mechanisms can learn some knowledge according to their specific designs, so their accuracy is increased. On the other hand, except for this first round, FedAVG will be almost stable at a lower accuracy no matter how many rounds. This is because FedAVG does not have any defense mechanism, so the attackers' models are always involved in the aggregation in the global model. The increase of accuracy in the first round (to the second round) for FedAVG is because of the effect of the aggregation.

Regarding attacking techniques, label-flipping (random) and label-flipping (PGD) attacks have a similar trend, but backdoor attacks significantly differ. It is noteworthy that the accuracy under backdoor attacks is measured only for the poisoned data while the accuracy under label-flipping attacks is measured for all testing data. Conducting this kind of measurement seems a little bit strange but it is the same as that in other previous literature.

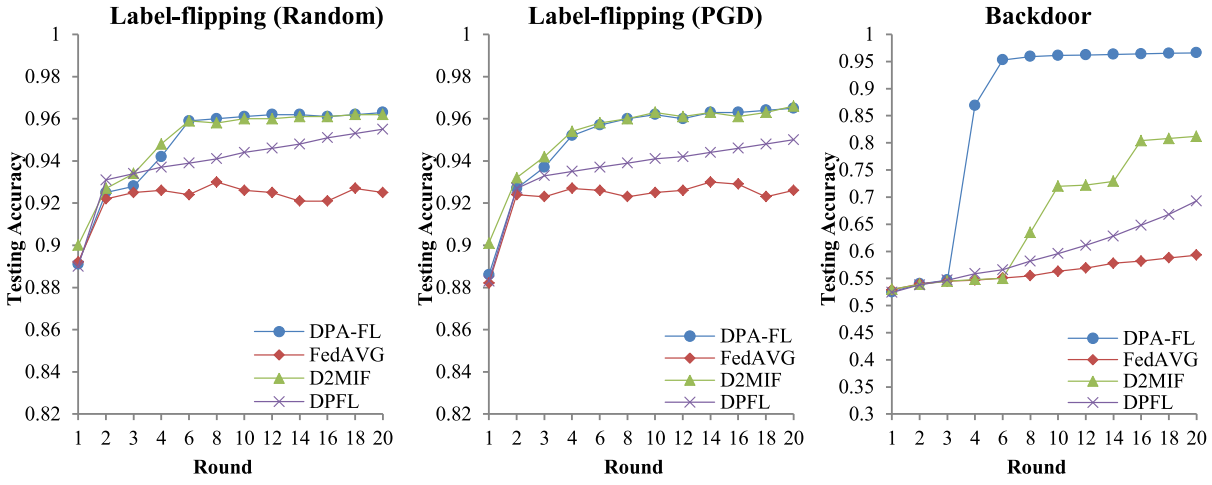

Fig. 5. Accuracy dynamics of DPA-FL and other mechanisms

Fig. 5. Accuracy dynamics of DPA-FL and other mechanisms

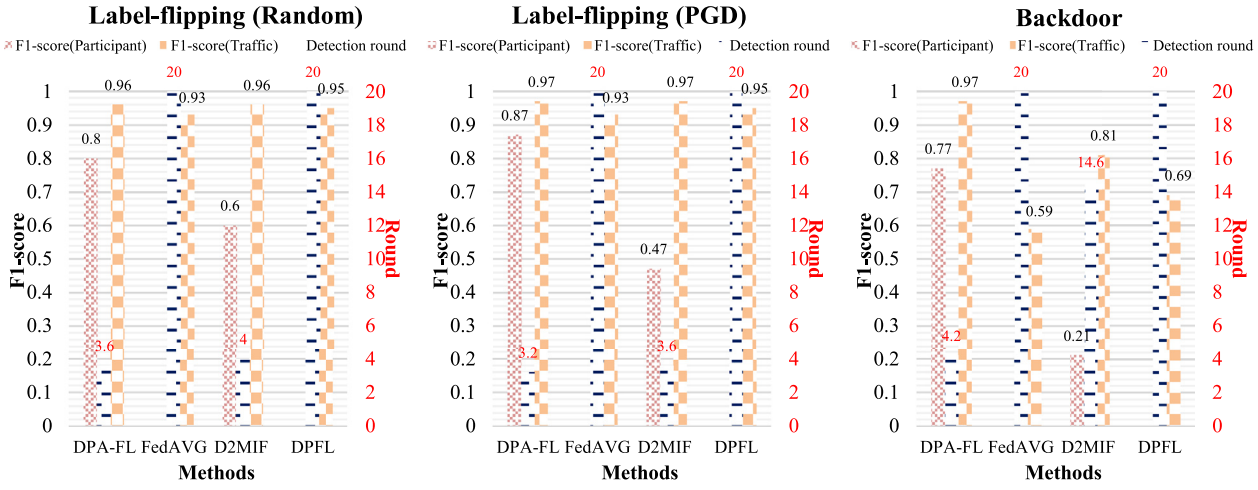

Fig. 6. Performance comparison of DPA-FL and other mechanisms

The reason is that the label-flipping attacks will affect the accuracy of overall testing data, while the backdoor attacks will not change this accuracy and only affect the accuracy of poisoned data. Observing from this figure, the accuracy of label-flipping attacks is moderate in the first few rounds because some attackers contribute to their corrupted models. However, once the attackers are detected and removed, the accuracy begins to rise steadily at a slow rate until reaching convergence. On the other hand, the accuracy of the mechanisms except for DPA-FL for backdoor attacks is very low, representing that these mechanisms encounter difficulties in classifying the attackers using backdoors. These approaches still slowly raise the accuracy of classifying the traffic because FedAVG, which is a baseline, can generate a better global model as more rounds pass even if it cannot defend against any backdoor attack.

About the comparison of four defense mechanisms, for label-flipping attacks, DPA-FL and D2MIF have a significant increase in accuracy in the second to fourth rounds, which is caused by the exclusion of the attacker. For DPFL, because the attacker is not removed and only its influence degree is adjusted, there is only a small improvement in accuracy. As more rounds pass, the accuracy of DPFL is increased a little because the influence degree of the attacker will be diluted. For FedAVG, the accuracy is almost stable because it has no defense mechanism. For backdoor attacks, only DPA-FL has a significant increase in accuracy because it excludes the attacker in the fourth round. The accuracy of FedAVG is kept at a low level because it does not have any defense mechanism. A little bit faster increase in accuracy for DPFL is that it only adjusts the influence degree of the attacker, but does not remove it. D2MIF can detect some backdoor attacks and detect more as more rounds pass, but its accuracy and efficiency are significantly lower than that of DPA-FL.

5.3. Comparison of DPA-FL and Other Mechanisms

The final results of each mechanism are shown in Fig. 6 where the left y-axis is the F1-score and the right y-axis is the number of detection rounds. Many interesting things can be observed from this figure. First, it can be found that both label-flipping attacks have few differences in their attacking effects, especially for DPA-FL and FedAVG. This represents that the effect of using a carefully designed approach, PGD, to generate label-flipping has similar results to the effect of using random label-flipping. This may be because DPA-FL can detect most label-flipping attacks, no matter random or PGD, while FedAVG cannot detect any attack.

Second, about F1-score (participant), FedAVG and DPFL will generate undefined values, which are not shown in the figure, because their false negative rate is one, as shown in Fig. 7. The reason is that FedAVG cannot identify any attacker, so it regards all participants as benign ones. Similarly, DPFL also cannot explicitly identify any attacker and only reduces the influence degree according to its suspiciousness. D2MIF can achieve a low value on F1-score (participant), i.e., 0.60 for label-flipping (random), 0.47 for label-flipping (PGD), and 0.21 for backdoor attacks, representing that sometimes D2MIF cannot identify the attacker and/or some benign participants are misclassified as the attackers. On the other hand, DPA-FL can more correctly classify the attacker and benign participants, so its F1-score (participant) is higher as 0.8, 0.87, and 0.77, for label-flipping (random), label-flipping, and backdoor, respectively. For the detection round, the value of FedAVG and DPFL is the largest number of rounds set in the experiment because they do not exclude the attacker. However, DPA-FL and D2MIF achieve similar detection rounds in label-flipping, but the detection round of D2MIF for backdoor attacks is as large as 14.6. In fact, this value is an average result. In our experiments, sometimes D2MIF can identify the attacker which uses the backdoor attacks, but sometimes it cannot.

Third, in the comparison of four defense mechanisms, for label-flipping attacks, because FedAVG does not have any defense mechanism, the F1-score (traffic) is the lowest at 93.2%. DPFL only reduces the influence degree of the attackers, so its F1-score (traffic) can rise to about 94.6%. DPA-FL and D2MIF methods both remove the attacker, so both reach about 96.5%. Note although D2MIF can achieve a high F1-score (traffic), but obtain a very low F1-score (participant). That is, as shown in Fig. 7, D2MIF has a high false positive rate, i.e., misclassifies benign participants as attackers and excludes them. Under this case, the global model will aggregate fewer local models, sometimes resulting in a less accurate global model or a longer converge time. In terms of backdoor attacks, FedAVG achieves 0.59 on F1-score (traffic) because it cannot detect any backdoor attack. However, DPFL can reduce the influence degree of the attacker, so it can obtain a higher value of about 0.69. Our proposed mechanism DPA-FL can correctly classify the attacker which adopts backdoor attacks and benign participants, so it can achieve 0.77 on F1-score (participant) and 0.97 on F1-score (traffic). Overall speaking, DPA-FL can detect the label-flipping attacker and excludes it, so the F1-score (traffic) is improved by about 3% compared to other mechanisms. The small improvement is mainly caused that the original accuracy achieved by FedAVG is quite high. Also, the DPA-FL can detect the backdoor at-

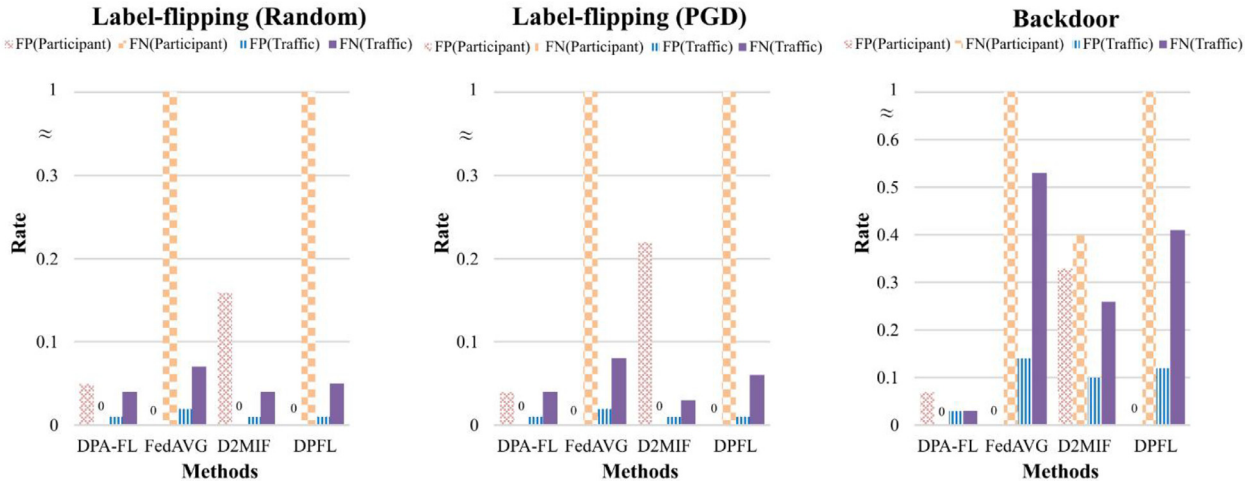

Fig. 7. False positive rate and false negative rate of all mechanisms

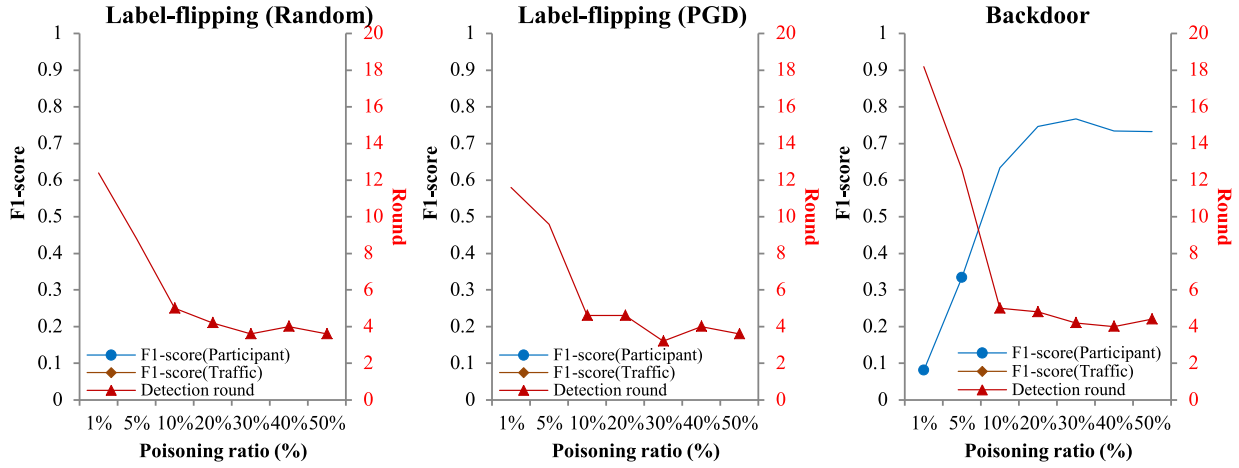

Fig. 8. Effect of data poisoning ratio

tacker and excludes it, so the F1-score (traffic) is improved by 20 ($\frac{0.97-0.81}{0.81}$)~64% ($\frac{0.97-0.59}{0.59}$) compared to other mechanisms.

In summary, D2MIF is based on accuracy testing, while backdoor attacks only pinpoint specific targets and do not affect accuracy. This causes that D2MIF is useful against label-flipping attacks but weak in identifying the attackers which use backdoor attacks. Although DPFL can successfully suspect both attackers, it merely adjusts their influence degrees to slightly relieve their damages but cannot completely avoid them because the global model is still polluted. DPA-FL undergoes relative comparison during RP, detecting both label-flipping attacks and backdoor attacks because their models' weights are different from that of benign participants. The testing data in AP is particularly designed to include backdoor data, so DPA-FL can further strengthen the detection of backdoor attacks. Also when the attackers are detected, they are excluded to avoid polluting the global model. Thus DPA-FL can achieve the highest F1-score (participant), meaning that it correctly classifies the attackers regardless of label-flipping (random), label-flipping (PGD), or backdoor attacks, and most benign participants.

Fig. 7 shows the intermittent results of the false positive rate (participant), false negative rate (participant), false positive rate (traffic), and false negative rate (traffic). These results can provide evidence for the above explanations about these mechanisms. For example, for label-flipping attacks, we can see the false negative rates (participant) achieved by DPA-FL and D2MIF are zero, representing that they can correctly detect these attacks. On the other hand, the false negative rates (participant) achieved by FedAVG and DPFL are one, representing that they cannot detect these attacks. The reason is mentioned before. DPA-FL and D2MIF achieve a false positive rate of 0.05 and 0.16 for label-flipping (random) and 0.04 and 0.22 for label-flipping (PGD), respectively, representing that sometimes they will misclassify a few benign participants as attackers. However, DPA-FL still can perform better than D2MIF. For backdoor attacks, DPA-FL always has a significantly smaller false positive rate (traffic) and false negative rate (traffic), compared with other mechanisms, representing that it can more correctly classify malicious and benign traffic.

5.4. Effect of Data Poisoning Ratio

Fig. 8 shows the effects of data poisoning ratios on the performance of DPA-FL. When the ratio is small, more detection rounds are needed. This is because when the amount of poisoned data is small, the difference between the benign participants and the attacker becomes small, resulting in more difficult detection. In some cases, e.g. backdoor attacks with a 1% poisoning ratio, false negatives even happen. However, F1-score (traffic) does not change much because of a similar reason. When the amount of poisoned

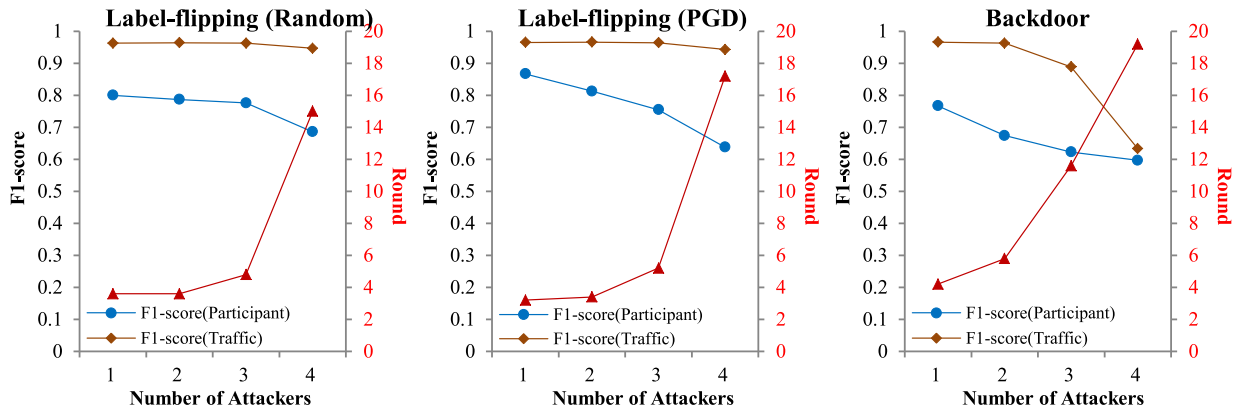

Fig. 9. Effect of the number of attackers

data is small, the local model of the attacker is similar to that of the benign participants. As the global model aggregates the few-difference malicious model with many benign models, the malicious weights only generate few influences on the global model. Thus, even the attacker with a little portioned data cannot be correctly classified, but its effect on the global model is minor, so DPA-FL still generates a good result on F1-score (traffic).

For backdoor attacks, although the number of detection rounds and F1-Score (participant) slightly fluctuate due to the experimental variation, it can still be found that the larger the poisoning ratio, the faster and the easier the detection. Note that the F1-score (participant) value of 0.3∼0.8 in the figure is because that DPA-FL misclassifies some benign participants as attackers, but it still correctly classifies the attacker. However, aggregating fewer but enough benign participants can still generate a good and almost same F1-score (traffic).

5.5. Effect of the Number of Attackers

Fig. 9 shows the performance of DPA-FL with the increase in the number of attackers. We can see that regardless of the attacking techniques, the more attackers, the larger the number of detection rounds, the lower the F1-score (participant), and the lower the F1-score (traffic). Because RP provides relative comparison, when more attackers exist, RP detection might make a wrong decision because of two reasons: (1) RP only detects at most one attacker; (2) the values of quartiles will be biased because of too many attackers. In this case, DPA-FL must rely on the defense mechanism of AP. However, the effect of defense in AP is also affected by the classification results, i.e., **B** and **M**, done by RP, Thus, when the number of attackers is larger, the performance of DPA-FL becomes worse. Overall, in the case of a small number of attackers, i.e., less than one-quarter of all participants, DPA-FL can exclude the attackers within twelve rounds and achieve a low false positive rate.

Although the number of attackers does affect the defense performance, fortunately, only few attackers likely appear in a realistic environment.

6. Conclusions and Future Works

This paper proposes a two-phase defense method, DPA-FL, applied to the global server to mitigate the effects of poisoning attacks. The first phase, RP, uses LOF to compare models' weights among participants, removing the obvious attackers. The second phase, AP, screens attackers using verifiable testing data prepared in advance. We also dynamically adjust the threshold by applying RL, enhancing the classification accuracy.

There are some interesting observations from our experiment: (1) using random or PGD to conduct label-flipping attacks does not differ significantly for DFA-FL because it has an excellent defense performance. (2) Backdoor attacks are more effective than label-flipping attacks because most of the defense mechanisms cannot well defend against the former. (3) the more attackers, the more effective the attacks. When the number of attackers occupies more than one-quarter of the participants, the accuracy of defense will significantly drop. (4) An attacker who increases the amount of poisoning data will not cause more serious damage because it is more easily detected. On the other hand, although an attack with a low poisoning ratio is less likely to be detected, it only causes tiny damage to accuracy. (5) Lastly, compared with the mechanism without any defense, FedAVG, and other defense mechanisms, DPA-FL can improve about 3% on F1-score (traffic) for label-flipping attacks and 20∼64% for backdoor attacks. Also, DPA-FL can remove the attackers within twelve rounds when the attackers are few. The small improvement for label-flipping is mainly caused that the original accuracy achieved by FedAVG is quite high.

As the number of participants in FL might be large, the quality of trained data and model in each participant can greatly vary. Therefore, we want to consider the significance of each participant and increase the weights of important ones. Such adjustment can be conducted through a comprehensive credit scoring system. Those with larger credits have larger impacts on the detection, increasing the accuracy of defense. By the way, currently the testing data used in AP is obtained through random sampling. In the future, we will carefully select fewer representative samples to reduce the testing time. Finally, this study only considers an IID environment, so how to handle a non-IID dataset should be another direction in the future.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Yuan-Cheng Lai: Conceptualization, Methodology, Writing – review & editing, Supervision. **Jheng-Yan Lin:** Software, Investigation, Writing – original draft. **Ying-Dar Lin:** Validation, Supervision, Writing – review & editing. **Ren-Hung Hwang:** Validation, Project administration, Funding acquisition. **Po-Chin Lin:** Writing – review & editing. **Hsiao-Kuang Wu:** Writing – review & editing. **Chung-Kuan Chen:** Funding acquisition.

Data Availability

Data will be made available on request.

References

Rey, V., Sánchez, P.M.S., Celdrán, A.H., Bovet, G., 2022. Federated Learning for Malware Detection in IoT Devices. Computer Networks 204 (108693). doi:10.1016/j.comnet.2021.108693.

Al-Marri, N.A.A.-A., Ciftler, B.S., Abdallah, M.M., 2020. Federated Mimic Learning for Privacy Preserving Intrusion Detection. In: International Black Sea Conference on Communications and Networking (BlackSeaCom), pp. 1–6. doi:10.1109/BlackSeaCom48709.2020.9234959.

Nguyen, T.D., Marchal, S., Miettinen, M., Fereidooni, H., Asokan, N., Sadeghi, A.-R., 2019. DÏoT: A Federated Self-Learning Anomaly Detection System for IoT. In: 39th International Conference on Distributed Computing Systems (ICDCS), pp. 756–767. doi:10.1109/ICDCS.2019.00080.

N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the Science of Security and Privacy in Machine Learning," *arXiv preprint arXiv:1611.03814*, 2016. doi: 10.48550/arXiv.1611.03814

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Srndic, N., Laskov, P., Giacinto, G., Roli, F., 2013. Evasion Attacks Against Machine Learning at Test Time. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 387–402. doi:10.1007/978-3-642-40994-3_25.

Biggio, B., Nelson, B., Laskov, P., 2012. Poisoning Attacks Against Support Vector Machines. In: 29th International Conference on International Conference on Machine Learning, pp. 1467–1474. doi:10.48550/ArXiv.1206.6389.

Xiao, H., Biggio, B., Nelson, B., Xiao, H., Eckert, C., Roli, F., 2015. Support Vector Machines under Adversarial Label Contamination. Neurocomputing 160, 53–62. doi:10.1016/j.neucom.2014.08.081.

Wang, B., Yao, Y, Shan, S., Li, H., Viswanath, B., Zheng, H., et al., 2019. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In: IEEE Symposium on Security and Privacy (SP), pp. 707–723. doi:10.1109/SP.2019.00031.

Zhang, J., Chen, J., Wu, D., Chen, B., Yu, S., 2019. Poisoning Attack in Federated Learning using Generative Adversarial Nets. In: 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), pp. 374–380. doi:10.1109/TrustCom/BigDataSE.2019.00057.

Zhang, J., Chen, B., Cheng, X., Binh, H.T.T., Yu, S., 2020. PoisonGAN: Generative Poisoning Attacks Against Federated Learning in Edge Computing Systems. IEEE Internet of Things Journal 8 (5), 3310–3322. doi:10.1109/JIOT.2020.3023126.

Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., Shmatikov, V., 2020. How to Backdoor Federated Learning. In: International Conference on Artificial Intelligence and Statistics, pp. 2938–2948. doi:10.48550/ArXiv.1807.00459.

Zhou, X., Xu, M., Wu, Y., Zheng, N., 2021. Deep Model Poisoning Attack on Federated Learning. Future Internet 13 (3), 73. doi:10.3390/fi13030073.

Nguyen, T.D., Rieger, P., Miettinen, M., Sadeghi, A.-R., 2020. Poisoning Attacks on Federated Learning-Based IoT Intrusion Detection System. In: Workshop Decentralized IoT Syst. Secur (DISS), pp. 1–7.

Short, A.R., Leligou, H.C., Papoutsidakis, M., Theocharis, E., 2020. Using Blockchain Technologies to Improve Security in Federated Learning Systems. In: IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), pp. 1183–1188. doi:10.1109/COMPSAC48688.2020.00-96.

Liu, W., Lin, H., Wang, X., Hu, J., Kaddoum, G., Piran, Md.Jalil, et al., 2021. D2MIF: A Malicious Model Detection Mechanism for Federated Learning Empowered Artificial Intelligence of Things. IEEE Internet of Things Journal doi:10.1109/JIOT.2021.3081606.

Doku, R., Rawat, D.B., 2021. Mitigating Data Poisoning Attacks on A Federated Learning-Edge Computing Network. In: IEEE 18th Annual Consumer Communications & Networking Conference (CCNC), pp. 1–6. doi:10.1109/CCNC49032.2021.9369581.

Huang, S.-M., Chen, Y.-W., Kuo, J.-J., 2021. Cost-Efficient Shuffling and Regrouping Based Defense for Federated Learning. In: IEEE Global Communications Conference (GLOBECOM), pp. 1–6. doi:10.1109/GLOBECOM46510.2021.9685499.

Chen, Z., Lv, N., Liu, P., Fang, Y., Chen, K., Pan, W., 2020. Intrusion Detection for Wireless Edge Networks Based on Federated Learning. IEEE Access 8, 217463–217472. doi:10.1109/ACCESS.2020.3041793.

J. Zhou, N. Wu, Y. Wang, S. Gu, Z. Cao, X. Dong, et al., "A Differentially Private Federated Learning Model against Poisoning Attacks in Edge Computing," *IEEE Transactions on Dependable and Secure Computing*. doi: 10.1109/TDSC.2022.3168556

Gu, Z., Yang, Y., 2021. Detecting Malicious Model Updates from Federated Learning on Conditional Variational Autoencoder. In: IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 671–680. doi:10.1109/IPDPS49936.2021.00075.

Yang, Q., Liu, Y., Chen, T., Tong, Y., 2019. Federated Machine Learning: Concept and Applications. ACM Transactions on Intelligent Systems and Technology (TIST) 10 (2), 1–19. doi:10.1145/3298981.

Nilsson, A., Smith, S., Ulm, G., Gustavsson, E., Jirstrand, M., 2018. A Performance Evaluation of Federated Learning Algorithms. In: 2nd workshop on distributed infrastructures for deep learning, pp. 1–8. doi:10.1145/3286490.3286559.

Panigrahi, R., Borah, S., 2018. A Detailed Analysis of CICIDS2017 Dataset for Designing Intrusion Detection Systems. Int. J. Eng. Technol. 7 (3.24), 479–482.

Sutton, R.S., Barto, A.G., 1998. Reinforcement Learning: An Introduction. IEEE Transactions on Neural Networks 9 (5). doi:10.1109/TNN.1998.712192, 1054-1054.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *arXiv preprint arXiv: 1312.5602*, 2013. doi: 10.48550/ArXiv.1312.5602

Thrun, S., Schwartz, A., 1993. Issues in Using Function Approximation for Reinforcement Learning. Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum.

Breunig, M.M., Kriegel, H.-P., Ng, R.T., Sander, J., 2000. LOF: Identifying Density-Based Local Outliers. In: ACM SIGMOD International Conference on Management of Data, pp. 93–104. doi:10.1145/342009.335388.

Auskalnis, J., Paulauskas, N., Baskys, A., 2017. Application of Local Outlier Factor Algorithm to Detect Anomalies in Computer Network. Elektronika ir Elektrotechnika 24 (3), 96–99. doi:10.5755/j01.eie.24.3.20972.

Ma, J., Teng, Z., Tang, Q., Qiu, W., Yang, Y., Duan, J., 2022. Measurement Error Prediction of Power Metering Equipment Using Improved Local Outlier Factor and Kernel Support Vector Regression. IEEE Transactions on Industrial Electronics 69 (9), 9575–9585. doi:10.1109/TIE.2021.3114740.

Preuveneers, D., Rimmer, V., Tsingenopoulos, I., Spooren, J., Joosen, W., Ilie-Zudor, E., 2018. Chained Anomaly Detection Models for Federated Learning: An Intrusion Detection Case Study. Applied Sciences 8 (2663). doi:10.3390/app8122663.

Neto, H.C., Dusparic, I., Mattos, D., Fernandes, N.C., 2022. FedSA: Accelerating Intrusion Detection in Collaborative Environments with Federated Simulated Annealing. IEEE 8th International Conference on Network Softwarization (NetSoft) doi:10.1109/NetSoft54395.2022.9844024.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," *arXiv preprint arXiv:1706.06083*, 2017. doi: 10.48550/arXiv.1706.06083

Bachl, M., Hartl, A., Fabini, J., Zseby, T., 2019. Walling up Backdoors in Intrusion Detection Systems. In: 3rd ACM CoNEXT Workshop on Big Data, Machine Learning and Artificial Intelligence for Data Communication Networks, pp. 8–13. doi:10.1145/3359992.3366638.

Yaun-Cheng Lai received his Ph.D. degree in the Department of Computer and Information Science from National Chiao Tung University in 1997. He joined the faculty of the Department of Information Management at National Taiwan University of Science and Technology in August 2001 and has been a distinguished professor since June 2012. His research interests include performance analysis, wireless networks, network security and machine learning.

Jheng-Yan Lin received Master degree of Information Management of National Taiwan University of Science and Technology, in 2022. His research interests include machine learning, network security, and intrusion detection.

Ying-Dar Lin is a Chair Professor of computer science at National Yang-Ming Chiao-Tung University (NYCU), Taiwan. He received his Ph.D. in computer science from The University of California at Los Angeles (UCLA) in 1993. He was a visiting Scholar at Cisco Systems in San Jose during 2007–2008, CEO at Telecom Technology Center, Taiwan, during 2010–2011, and Vice President of National Applied Research Labs (NARLabs), Taiwan, during 2017–2018. He was the founder and director of Network Bench marking Lab (NBL) in 2002–2018, which reviewed network products with real traffic and automated tools, and has been an approved test lab of the Open Networking Foundation (ONF). He also cofounded L7 Networks Inc. in2002, later acquired by D-Link Corp, and O'Prueba Inc. as pin-off from NBL, in2018. His research Interests include network security, wireless communications, network softwarization, and machine learning for communications. His work on multi-hop cellular was the first along this line, and has been cited over 1000 times and standardized into IEEE 802.11s, IEEE 802.15.5, IEEE 802.16j, and 3GPP LTE-Advanced. He is an IEEE Fellow (class of 2013), IEEE Distinguished Lecturer (2014–2017), ONF Research Associate (2014–2018), and received in 2017 Research Excellence Award and K. T. Li Break through Award. He has served or is serving on the editorial boards of several IEEE journals and magazines, including Editor-in-Chief of IEEE Communications Surveys and Tutorials (COMST,1/2017-12/2020). He published a textbook, Computer Networks: An Open Source Approach, with Ren-Hung Hwang and Fred-Baker (McGraw-Hill, 2011).

Ren-Hung Hwang received his Ph.D. degree in computer science from the University of Massachusetts, Amherst, Massachusetts, USA, in 1993. He is the Dean of the College of Artificial Intelligence, National Yang Ming Chiao Tung University (NYCU), Taiwan. Before joining NYCU, he was with National Chung Cheng University, Taiwan, from 1993 to 2022. He is currently on the editorial boards of IEEE Communications Surveys and Tutorials and IEICE Transactions on Communications. He received the Best Paper Award from The 6th International Conference on Internet of Vehicles 2019, IEEE Ubi-Media 2018, IEEE SC2 2017, IEEE IUCC 2014, and the IEEE Outstanding Paper Award from IEEE IC/ATC/ICA3PP 2012. He served as the general chair of the International Computer Symposium (ICS), 2016, and International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN) 2018, International Symposium on Computer, Consumer and Control (IS3C) 2018, IEEE DataCom 2019 (The 5th IEEE International Conference on Big Data Intelligence and Computing). He received the Outstanding Technical Achievement Award of the IEEE Tainan Section in 2022. His research interests include deep learning, network security, wireless communications, Internet of Things, cloud and edge computing.

Po-Ching Lin received his Ph.D. degree in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 2008. He joined the faculty of the Department of Computer Science and Information Engineering, CCU, in August 2009. He is cur-

rently a professor. His research interests include network security, network traffic analysis, and performance evaluation of network systems.

Hsiao-Kuang Wu received the B.S. degree in computer science and information engineering from National Taiwan University, Taipei, Taiwan, in 1989, and the M.S. and Ph.D. degrees in computer science from the University of California at Los Angeles, Los Angeles, CA, USA, in 1993 and 1997, respectively. He is a Professor of Computer Science and Information Engineering with National Central University, Chung-Li, Taiwan. His primary research interests include wireless networks, mobile computing, and broadband networks. Prof. Wu is a member of the Institute of Information and Computing Machinery.

Chung-Kuan Chen received his Ph.D. degree in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 2018. Currently he is working on CyCraft Technology. His research interests include network security, network traffic analysis, and system security.