D2MIF: A Malicious Model Detection Mechanism for Federated-Learning-Empowered Artificial Intelligence of Things

Wenxin Liu, Hui Lin, Xiaoding Wang, Jia Hu, Georges Kaddoum, *Senior Member, IEEE*, Md. Jalil Piran, *Senior Member, IEEE*, and Atif Alamri, *Member, IEEE*

***Abstract*—Artificial Intelligence of Things (AIoT), as a fusion of artificial intelligence (AI) and Internet of Things (IoT), has become a new trend to realize the intelligentization of industry 4.0 and the data privacy and security is the key to its successful implementation. To enhance data privacy protection, the federated learning has been introduced in AIoT, which allows participants to jointly train AI models without sharing private data. However, in federated learning, malicious participants might provide malicious models by launching the poisoning attack, which will jeopardize the convergence and accuracy of the global model. To solve this problem, we propose a malicious model detection mechanism based on the isolation forest (iforest), named D2MIF, for the federated learning-empowered AIoT. In D2MIF, an iforest is constructed to compute the malicious score for each model uploaded by the corresponding participant, and then, the models will be filtered if their malicious scores are higher than the threshold, which is dynamically adjusted using reinforcement learning (RL). The validation experiment is conducted on two public data sets Mnist and Fashion_Mnist. The experimental results show that the proposed D2MIF can effectively detect malicious models and significantly improve the global model accuracy in federated learning-empowered AIoT.**

***Index Terms*—Artificial Intelligence of Things (AIoT), federated learning, isolation forest (iforest), poisoning attack, security.**

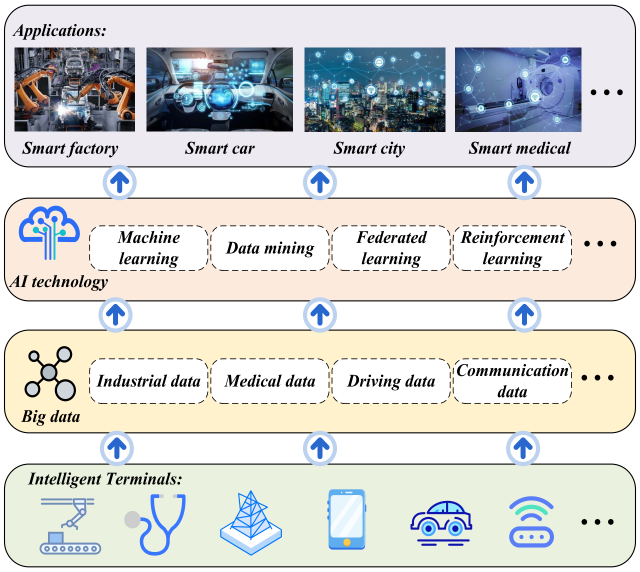


Fig. 1. AIoT structure.

I. Introduction

With the advent of the Industry 4.0, the Industrial Internet of Things (IIoT) will usher in a new development opportunity, leading to a variety of industrial equipments linked to each other through the IIoT and massive industrial application data generated in the IIoT [1]. In order to make full use of these data to support dynamic, real-time, and accurate decision making for various IIoT applications and services, the artificial intelligence (AI) technology is introduced to the IIoT, forming a new architecture, namely, AI of Things (AIoT) [2].

As a new trend, AIoT combines AI and IIoT to provide intelligent communications between industrial devices and efficient industrial data processing [3]. Fig. 1 shows the structure of AIoT. In AIoT, a variety of data, such as communication data, medical data, industrial data, etc., are collected through environment-device interactions by intelligent terminals. With the help of the AI technology, the deep relationship between data and data is discovered and analyzed to support various intelligent applications.

However, the traditional machine learning (ML) methods are subject to data privacy leakage during the data processing for AIoT [4]–[6]. To overcome this challenge, the federated learning [7], [8] that enables the data exchange between the server and the participants without the need of original data has

Manuscript received 31 December 2020; revised 4 March 2021 and 2 April 2021; accepted 4 May 2021. Date of publication 18 May 2021; date of current version 24 January 2023. This work was supported by the Deanship of Scientific Research at King Saud University, Riyadh, Saudi Arabia, through the Vice Deanship of Scientific Research Chairs: Research Chair of Pervasive and Mobile Computing. *(Corresponding authors: Hui Lin; Md. Jalil Piran.)*

Wenxin Liu, Hui Lin, and Xiaoding Wang are with the College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350117, China (e-mail: sixwenxin@163.com; linhui@fjnu.edu.cn; wangdin1982@fjnu.edu.cn).

Jia Hu is with the Department of Computer Science, University of Exeter, Exeter EX4 4QJ, U.K. (e-mail: j.hu@exeter.ac.uk).

Georges Kaddoum is with the Department of ECE, École de Technologie Supérieure, Montreal, QC H3C 1K3, Canada (e-mail: georges.kaddoum@etsmtl.ca).

Md. Jalil Piran is with the Department of Computer Science and Engineering, Sejong University, Seoul 05006, South Korea (e-mail: piran@sejong.ac.kr).

Atif Alamri is with the Research Chair of Pervasive and Mobile Computing, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia (e-mail: atif@ksu.edu.sa).

Digital Object Identifier 10.1109/JIOT.2021.3081606

become the promising solution in AIoT. Although the federated learning can protect privacy better than the traditional ML, the server cannot verify the security and availability of local models of the participants [9]–[11]. For example, malicious participants might use poisoned data sets for model training and upload malicious local models to affect the accuracy of the global model. Therefore, in federated learning-empowered AIoT, it is necessary for the server to detect and filter malicious local models before model aggregation. To cope with aforementioned challenges, a malicious model detection mechanism based on isolation forest (iforest) [12], named D2MIF, is proposed for the federated learning-empowered AIoT. The main contributions of this article are summarized as follows.

1) To effectively detect malicious models, the iforest algorithm is integrated with the federated learning framework. To be specific, we consider the model uploaded by each participant as a leaf node of an isolation tree (itree) of the iforest. Considering the leaf nodes that represent malicious models are closer to the root, the distance between the leaf node and the root is used to calculate the anomaly score of each model. By giving a proper threshold, malicious models can be detected. For example, if the model's malicious score is higher than the threshold, then the model is malicious.
2) To improve the accuracy of malicious model detection, we employ the deep RL (DRL) method, i.e., the twin delayed deep deterministic policy gradients (TD3) algorithm [13], to dynamically adjust the detection threshold.
3) To validate the proposed mechanism D2MIF, we conduct the validation experiment on two public data sets: a) Mnist and b) Fashion_Mnist. The experimental results show that the proposed D2MIF can effectively detect malicious models and significantly improve the global model accuracy in federated learning-empowered AIoT.

II. Related Work

The detection of the malicious model has achieved certain research results in the context of federated learning. Blanchard *et al.* [14] proposed an aggregation method to enhance security in federated learning. The top m models with the farthest Euclidean distance from the average model are removed from the process of model aggregation. Cao *et al.* [15] proposed a detection mechanism that maps the model to vertices and edges on a graph through Euclidean distance, and finds the largest optimal clique on the graph to select the updated model and filter the malicious model. Zhao *et al.* [16] proposed a detection mechanism which the server generates a set of audit data by using generative malicious networks to detect whether the participant uploads a malicious model by auditing the accuracy of the participant's uploaded model. Fung *et al.* [17] proposed a mitigating sybils attack method named FoolsGold using the cosine similarity and pardon mechanism to distinguish malicious participants from normal participants. Li *et al.* [18] proposed an autoencoder-based malicious detection method

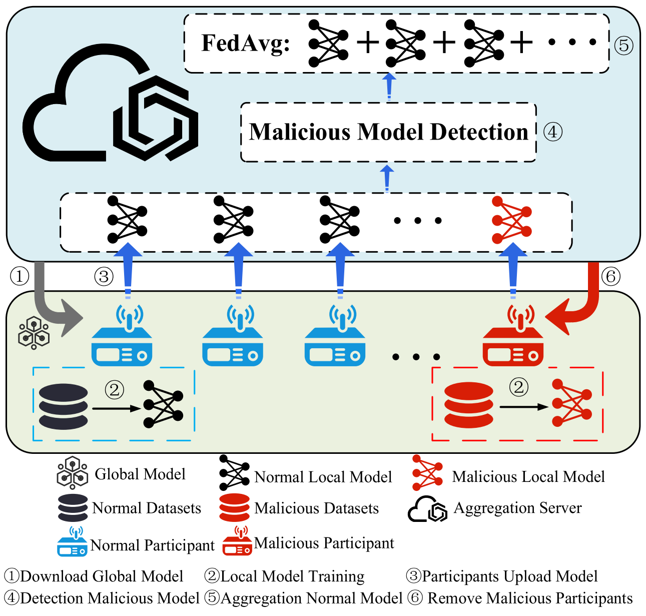


Fig. 2. Proposed system structure.

and introduced a credit score for each participant to advance the model aggregation. Zhao *et al.* [19] proposed a mechanism to assign the detection task to participants in collaborative learning with stable evaluation performance, and to detect malicious models through cross-validation between participants. Tan *et al.* [20] proposed a verify-before-aggregate (VBA) procedure using DRL to detect malicious updates from normal updates. Kang *et al.* [21] introduced a reputation mechanism for each participant to filter out malicious model updates for federated learning, and used a consortium blockchain to management reputation mechanism in security. Tolpegin *et al.* [22] proposed a method to identify malicious updates by extracting the unique characteristics of malicious participants uploading parameter updates by dimensionality reduction and principal component analysis (PCA).

III. System Model

A. System Model

In this article, we consider a client–server model as shown in Fig. 2. Participants including some malicious ones in the federated learning system download the initialized neural network model from the aggregation server. Each participant uses the local data set to train the local model. Once the local training is completed, participants upload the local models to the aggregation server and then the malicious model detection mechanism is activated. Once the malicious model detection is completed, the aggregation server utilizes the federated average algorithm (FedAvg) [23] to aggregate all normal models. We assume the aggregation server is honest; therefore, it cannot access local data of any participant. The aggregation server can only receive local models uploaded by participants and perform model aggregation. In addition, no participant can access the others' local data.

B. Attack Model

Malicious participants use the poisoning attack to train and generate malicious models. According to the attack purpose of the malicious participant, the poisoning attack can be divided into the targeted poisoning attack and the untargeted poisoning attack [24], [25].

1) *Targeted Poisoning Attack:* The targeted poisoning attack aims to change a small amount of prediction behavior of the model according to the intention of the attacker, while maintaining high accuracy in global prediction, such as backdoor attacks [26].
2) *Untargeted Poisoning Attack:* An untargeted poisoning attack is designed to reduce the overall accuracy of the global model and make the global model unable to converge normally, such as flip label attacks [22].

According to the attack methods of malicious participants, the poisoning attack can be divided into data poisoning attacks and model poisoning attacks [24].

1) *Data Poisoning Attack:* Malicious participants modify the local data set by using label flipping and other methods, and then use the poisoned data for local model training. Incorrect model updates are generated and uploaded to the server, which will ultimately affect the accuracy of the global model.
2) *Model Poisoning Attack:* An untargeted model poisoning attack can be Byzantine, and multiple malicious parties do not need to modify the local data set, and directly generate random local models through some predefined rules [24]. Malicious participants upload these random models to the server, causing the server to fail to converge when aggregating global models.

In this article, we consider the data poisoning attack. Specifically, malicious participants impose label flip operations on the local data set to perform data poisoning. Then, the local model trained using the poisoned data set will deteriorate the quality of the aggregated model. Let the sample-label pair of the normal data set be

$$\{(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_N, y_N)\}, \tag{1}$$

where $x_i, i = 1, \ldots, N$ is the sample, and $y_i, i = 1, \ldots, N$ is the label corresponding to the sample. The sample-label pair of the data set after the label flip operation is

$$\{(x_1, \overline{y_1}), (x_2, \overline{y_2}), (x_3, \overline{y_3}), \ldots, (x_N, \overline{y_N})\}, \tag{2}$$

where $x_i, i = 1, \ldots, N$ is the sample, and $\overline{y_i}, i = 1, \ldots, N$ is the label corresponding to the sample after label flipping. The set of all labels of the normal local data set Y and the set of all the labels of the malicious data set $\overline{Y}$ are, respectively, expressed as

$$Y = \{y_1, y_2, y_3, \ldots, y_N\}, \tag{3}$$
$$\overline{Y} = \{\overline{y_1}, \overline{y_2}, \overline{y_3}, \ldots, \overline{y_N}\}, \tag{4}$$

where $\overline{Y} = Y$ but $y_i \neq \overline{y_i}, i = 1, \ldots, N$.

Since the aggregation server is inaccessible to the participants' local data sets, the authenticity of participants' local data sets cannot be verified. That allows malicious participants to use the label flipped data sets for model training and upload the malicious model to the server to launch the data poisoning attack, which causes the global model to fail to converge normally.

In our attack model, malicious participants use the above-mentioned flip label operation to poison the local data set and train the poisoned local model. Therefore, we define the capabilities of malicious participants as follows.

1) The malicious participants are familiar with all the rules in federated learning.
2) The malicious participant can download the initialized model from the server.
3) The malicious participant can perform label flipping operations on any local data set.
4) The malicious participants have enough computing power to train the local model.

IV. Implementation of the D2MIF

In order to protect the data privacy of each participant, Konečný *et al.* [7] proposed a new distributed ML paradigm-federated learning. Federated learning requires multiple participants to collaborate to train an ML model. Participants do not need to upload local data sets to the server, and use private data sets to complete the model training locally. During each round of federated training, the participants download the initial model from the server for local training, and upload the trained model to the server after completing the training, and the server aggregates the received models. When a new round starts, the participants download the aggregated model from the server to continue training. Through multiple rounds of model interaction between the participants and the server until the global model reaches convergence.

Unlike normal datapoint, outlier has special characteristics. For example, the outliers are always fewer in number and are different from normal datapoints [12], which make outliers easy to be detected. In this article, we introduce the Isolation forest (IForest) that is an outlier detection algorithm [12], [27]. Specifically, the IForest randomly picks a value in attribute and divides the data point set into two subsets, which are presented in two leaf nodes. Then, it continues to divide with a certain attribute until there is only one datapoint in each leaf node. Due to the large number and high density of normal datapoints, the normal datapoints are divided many times before the division stops, but the outlier will soon be divided into a subset on a leaf node. Therefore, outliers are always closer to the root, while normal datapoints are farther from the root. That suggests the IForest algorithm can construct multiple Itrees through multiple sampling, and use the average depth as the final output depth. By calculating an anomaly score for each datapoint in leaf node, the first batch of datapoints with high abnormal scores is outliers, which are malicious models in our scheme.

Reinforcement learning (RL) is an ML process that maps the action selection through feedback from the external environment [28], [29]. The agent interacts with the external environment through RL, and learns how to make decisions to maximize the future cumulative rewards. RL can be modeled by a Markov decision process (MDP). Similar to MDP, the

Algorithm 1 Malicious Model Detection Process

```
1: Accepting the uploaded local model set;
2: Pre-aggregating local model in set and Calculating the $pre - accuracy$;
3: Comparing $pre - accuracy$ and accuracy of initial model ($init - accuracy$);
4: if $pre - accuracy > init - accuracy$ then
5:     Aggregation models in local model set;
6: else
7:     Calculating malicious score $m\_score(m^i)$ for each model $m^i$ in local model set by Isolation Forest;
8:     if $m\_score(m^i) >$ threshold then
9:         Model $m^i$ is malicious;
10:        Rejecting model $m^i$;
11:        Marking malicious model's participant as a sensitive participant once;
12:        if participant's $marking\quad times \geq 3$ then
13:            Removing this participant;
14:        else
15:            Retaining this participant in the following federated learning;
16:        end if
17:    else
18:        This model is normal model;
19:        Aggregation normal models;
20:    end if
21: end if
```

RL process is defined as a four-tuple $<A, S, R, STM>$, where A represents the set of actions that the agent can perform, S represents the state of the external environment that the agent can perceive, R represents the reward of the environment for the agent to take the current action, and STM is the state transition matrix of the agent. In RL, policy is defined to guide the agent on how to make the next action in the current state. The defined value function calculates the expectation of the future cumulative return that the agent can obtain after performing the action, and evaluates the performance of the current action by the agent.

To solve the problem caused by malicious participants in the federated learning-empowered AIoT system, we propose a detection method named D2MIF, aiming to detect the malicious models and eliminate the corresponding participants from the federated learning System. Since malicious participants perform the flip label operation on local data sets to generate poisoned data sets, each local model trained on the poisoned data set is an outlier compared with normal models. When all participants complete local model training and upload local models, the aggregation server will employ the D2MIF to detect outliers in the model set.

The detection process is shown in Algorithm 1. The detection mechanism will preaggregate all the currently received local models to calculate a $pre - accuracy$. When the $pre - accuracy$ is greater than the accuracy of initialization model ($init - accuracy$), the server does not execute the malicious model detection; if the $pre - accuracy$ is lower than the $init - accuracy$, the server will execute it. When performing malicious model detection, the server will build itrees and an iforest to calculate a malicious score for the local model uploaded by each participant. Then, we set a threshold for filtering malicious models, and use RL to dynamically adjust the threshold. A model with a malicious score less than the threshold is a normal model. Conversely, when a malicious score is greater than the threshold, the model is a malicious model. The server will perform formal model aggregation on normal models. Participants uploading malicious models will be marked as sensitive participants, and these malicious models will not be used for formal model aggregation. Sensitive participants will not be removed from the federated learning system for the time being, and can continue with the following federated learning tasks. In this way, it is ensured that some normal participants will not be judged as malicious participants due to poor models generated by one or two mistakes in training. When a participant is marked as a sensitive participant three times, the participant will be removed from the federated learning system. All removed participants will not be able to participate in the following federated training process.

In order to ensure the consistency between the local model of the participants and the initialization model in the server, we define a convolutional neural network (CNN) for all network models in the federated learning system. Specifically, each CNN model consists of two convolutional layers, two pooling layers, one flatten layer, and three fully connected layers. Among them, the convolutional layer 1 contains three 3×3 convolution kernels, and the activation function is ReLU; the convolutional layer 2 contains six 3×3 convolution kernels, and the activation function is ReLU; the fully connected layer 1 contains 256 units, the activation function is ReLU; the fully connected layer 2 contains 128 units, the activation function is ReLU; and the fully connected layer 3, which is the output layer, contains 10 units.

At the beginning of the Tth round of the federated training, all participants download the initialization model m_g from the server and use stochastic gradient descent (SGD) to optimize the local model m_l with the local data sets $D_{\text{local}} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i)\}$ for model training as follows:

$$m_l \leftarrow m_g - \eta \cdot \partial \frac{\text{Loss}\big(f\big(x_i, m_g\big), y_i\big)}{\partial x_i}, \tag{5}$$

where $\text{Loss}(\cdot)$ is the loss function, $f(\cdot)$ is the simulation function of the local neural network, and η is the learning rate of local model training. Before starting the local training, the malicious participants poisoned the local data set by flipping the labels, and used the poisoned data set for training the malicious model.

Once all participants have completed local training, the server randomly selects N participants, which may include malicious participants. Server preaggregate all the currently received local models and calculate a $pre - accuracy$. Due to the presence of malicious participants, the $pre - accuracy$ will be lower than the $init - accuracy$. The server will implement the malicious model detection mechanism. The collection of models received by the server can be expressed as SET_m

$$\mathrm{SET}_m = \left\{ m_l^0, m_l^1, \ldots, m_{l\_m}^{i+1}, m_{l\_m}^{i+1}, \ldots, m_l^{N-1} \right\}, \tag{6}$$

where m_l^i and $m_{l\_m}^i$ are, respectively, denoted as the normal local model and malicious local model, $i = 0, \ldots, N-1$.

Because the parameters of model are usually matrices of higher latitude, it is not conducive to our construction of an iforest anomaly detection model. Where we need to flatten all model parameters into 1-D vectors. The reason for flattening the model parameters into a 1-D vector can be attributed to the following two points. First, we represent the value at each position in the model parameter as a data feature of the model. After the model is flattened into a 1-D vector, the characteristics of the model can be expressed more intuitively. Moreover, after flattening the model into a 1-D vector, it is more convenient to apply the IForest algorithm.

After all the model are processed, there are N parameters in SET_m, respectively. The malicious model detection frame based on the iforest consists of three stages [12] as follow.

Stage 1: Using the SET_m to build the itree and the iforest, respectively.

Stage 2: The segment uses itree to predict the malicious score of each m^i.

Stage 3: Constructing a threshold adjustment model based on DRL, and selecting the malicious model according to threshold.

Specifically, in stage 1, we represent every model $m^i \in \mathrm{SET}_m, i = 0, \ldots, N-1$ as a leaf node in the binary tree. The height of each model m^i in the binary tree is represented as $H(m^i)$, and the average height in the forest is $H_{\mathrm{avg}}(m^i)$. The limit height of each tree is $H_{\mathrm{lim}} = \mathrm{ceiling}(\log_2 l)$, where l is the number of models from when constructing each tree and $\mathrm{ceiling}(\cdot)$ function is all values that the limit height of each tree H_{lim} can obtain do not exceed $\log_2 l$. In this stage, the iforest is constructed by recursively dividing the SET_m until each model parameter is isolation or reaches a preset tree height. Due to the length of the article refer to the original paper [12] for the algorithm of iforest and itree generation.

In stage 2, the server calculates a malicious score $\mathrm{m\_score}(m^i)$ for each model node through the iforest. The malicious score decreases as the average height of the model node in the tree increases, and the closer the data node to the root node, the greater the malicious score

$$\mathrm{m\_score}\left(m^i\right) = 2^{-\frac{H_{\mathrm{avg}}\left(m^i\right)}{c(N)}}, \tag{7}$$

where $c(N)$ is the normalization term

$$c(N) = 2H(N-1) - (2(N-1)/N), \tag{8}$$

where $H(\cdot)$ is the number of harmonics estimated by $\ln(N)+\gamma$, where γ is the Euler constant and $\ln(\cdot)$ is the natural logarithm. It presents the average length of unsuccessful searches in a binary search tree with N model node. For each model m^i, traverse each itree from the root node to the external node, and calculate the average height $H_{\mathrm{avg}}(m^i)$. Then, the server calculates the malicious score $\mathrm{m\_score}(m^i)$ reflecting the abnormal degree of the model according to $H_{\mathrm{avg}}(m^i)$.

In stage 3, we use the RL algorithm TD3 [13] to dynamically adjust the threshold used to filter malicious models. Since the TD3 solves the problem of overestimation of the action Q value in the critic network, and introduces a delay update and noise mechanism in the actor network to make the algorithm update more stable, we use TD3 for the threshold adjustment in our solution. Let the server act as an agent used to intelligently dynamically select the threshold. Then, the state: s, actions: a, and reward: r in the RL are defined as follows.

1) *s:* We define the state in which the server agent located as the malicious score of the participants uploads models calculated in stage 2. When the server calculates all the malicious score for the received model, the state s of the server at this time is $s = [\mathrm{m\_score}(m^0), \mathrm{m\_score}(m^1), \ldots, \mathrm{m\_score}(m^{N-1})]$.
2) *a:* We define the action a of the server agent in the RL process as adjusting the size of the threshold θ used to filter malicious models. Specifically, after the server observes the current environment, it will increase or decrease the size of the threshold used to filter malicious models according to the current state s and the goal of the accumulated maximum reward.
3) *r:* We define the reward r of environment feedback to the server agent as the prediction accuracy of the global model after model aggregation.

In our proposed threshold adjustment algorithm based on RL, six networks need to be initialized: 1) actor network: $P(s; \mu)$; 2) target actor network: $P^{\mathrm{tar}}(s; \mu')$; 3) critic network 1: $Q_1(a, s; \omega_1)$; 4) target critic network 1: $Q_1^{\mathrm{tar}}(a, s; \omega_1{}')$; 5) critic network 2: $Q_2(a, s; \omega_2)$; and 6) target critic network 2: $Q_2^{\mathrm{tar}}(a, s; \omega_2{}')$. Among them, the value before the semicolon represents the input of the neural network, and the value after the semicolon represents the parameters of the neural network. The actor network $P(s; \mu)$ is responsible for updating the parameters of the strategy network, and selects the action a to be performed according to the current state s of the agent, and calculates the reward r obtained by performing the current action and the next state s'. The target actor network $P^{\mathrm{tar}}(s; \mu')$ is responsible for calculating the optimal action a' in the s' based on the experience pool, and its parameters are regularly copied and updated from the actor network. The critic network 1 $Q_1(a, s; \omega_1)$ is responsible for the iterative update of the value network parameter, and the calculation is responsible for calculating the current Q value. The target critic network 1 $Q_1^{\mathrm{tar}}(a, s; \omega_1{}')$ is responsible for calculating the target Q value and its parameters are regularly copied and updated from the critic network 1. The functions of critic network 2 $Q_2(a, s; \omega_2)$ and target critic network 2 $Q_2^{\mathrm{tar}}(a, s; \omega_2{}')$ are basically the same as the former is a strategy adopted to suppress overestimation. When updating the target value y^{tar}, a smaller target critic network value is selected for update.

The server agent observes the malicious scores of all models in the current model set SET_m to determine the current state s and makes a threshold selection action a according to the policy $P(s; \mu)$. That is, the models with malicious scores higher than the current threshold will be detected as malicious ones and get removed from the model set SET_m. Then, the server observes the reward r, which is the accuracy of the current global model and the next state s', which is the malicious score of the model still exists in the current collection. This process can be expressed as a four-tuple transition $<s, a, r, s'>$. Then, the server agent calculates the value $Q_1(a, s; \omega_1)$ and $Q_2(a, s; \omega_2)$ of the current action performed in the current state. The server stores each explored four-tuple transition to the experience buffer for training.

Critic Network Update: The server randomly samples n transitions $<s, a, r, s'>$ from the experience buffer to update the critic network. The target value y^{tar} is calculated according to the following formula:

$$y^{\text{tar}} = r + \gamma \min\left(Q_i^{\text{tar}}\left(P^{\text{tar}}\left(s'; \mu'\right), s'; \omega_i'\right)\right)_{i=1,2}, \tag{9}$$

where γ is a discount factor and $\min(\cdot)$ represents the smaller value of $Q_i^{\text{tar}}(P^{\text{tar}}(s'; \mu'), s'; \omega_i')$ when i takes 1 or 2, respectively. The loss function is defined as follows:

$$\text{Loss}(\omega_i) = \frac{1}{n}\sum_n \left(y^{\text{tar}} - Q_i(a, s; \omega_i)\right)^2, \tag{10}$$

where $\sum_n(\cdot)$ means to sum $y^{\text{tar}} - Q_i(a, s; \omega_i')$ items.

The critic network parameter ω_i is updated according to the following:

$$\omega_i \leftarrow \omega_i - \alpha \cdot \frac{\partial \text{Loss}(\omega_i)}{\partial \omega_i}, \tag{11}$$

where α is the learning rate, and $[(\partial \text{Loss}(\omega_i))/(\partial \omega_i)]$ represents to find the derivative of $\text{Loss}(\cdot)$ with respect to ω_i.

Actor Network and Target Network Update: The objective function to be optimized for the actor network is $J(\mu)$

$$J(\mu) = \frac{1}{n}\sum_n Q_1(a, s; \omega_1)|_{a=P(s|\mu)} P(s; \mu). \tag{12}$$

Use the value function of critic to update the actor network every d period of time, where β is the learning rate

$$\mu \leftarrow \mu + \beta \cdot \frac{\partial J(\mu)}{\partial \mu}. \tag{13}$$

The updates of the target critic network and target actor network are as follows, where τ is the hyperparameter $\tau \in (0, 1)$

$$\omega_i' \leftarrow \tau\omega_i + (1-\tau)\omega_i' \tag{14}$$
$$\mu_i' \leftarrow \tau\mu_i + (1-\tau)\mu_i'. \tag{15}$$

If the actor network and the critic network cannot converge, then the server cannot aggregate models or remove the malicious participants, but conducts actor network and critic network model training. Once both actor network and critic network converge, the server will set a threshold that maximizes the accuracy of the global model based on the malicious score of the model uploaded by the current participants, and detect the malicious model based on the current threshold.

Recall that a model with a malicious score less than the threshold is a normal model, while a model with a malicious score greater than the threshold is a malicious one. To further improve the malicious model detection accuracy, we design the following mechanism. Let participants who upload malicious models be marked as sensitive participants.

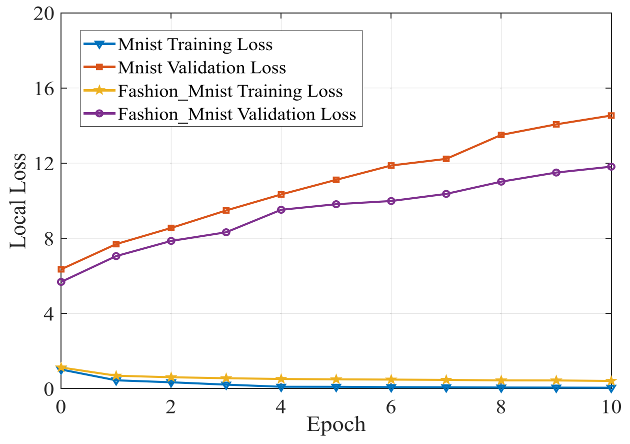


Fig. 3. Loss of malicious participants local training.

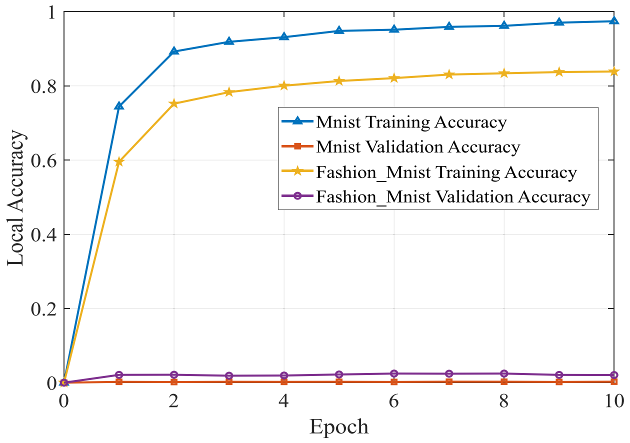


Fig. 4. Accuracy of malicious participants local training.

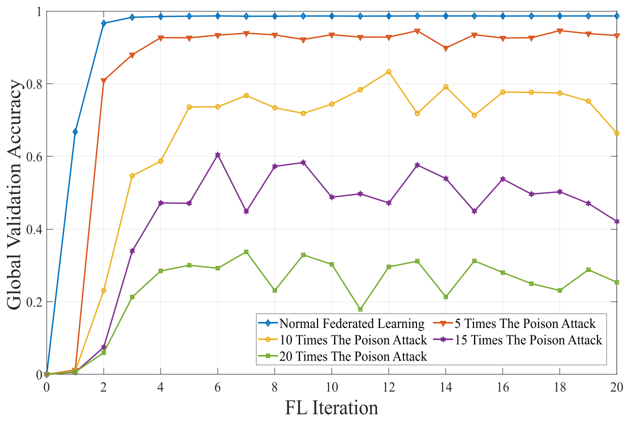


Fig. 5. Global accuracy under the different attack times.

All malicious models will not be used for model aggregation and sensitive participants will not be removed from the federated learning system. If a participant is marked as a sensitive participant three times, then this participant is considered as a malicious participant and will be removed from the federal learning system. All removed participants will not be able to participate in the following federated training process.

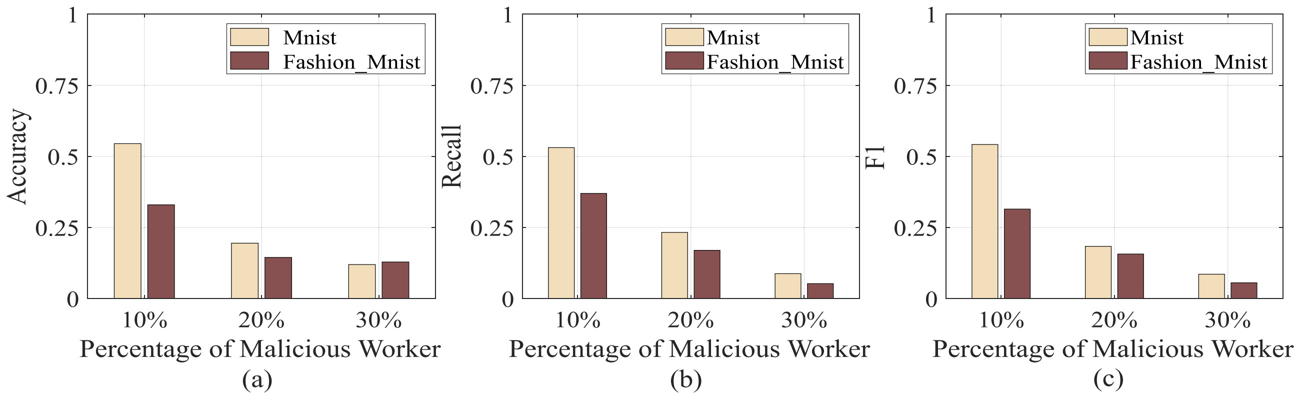


Fig. 6. (a) Accuracy, (b) Recall, and (c) F1 of global model under the different numbers of malicious participants.

V. Performance Evaluation

A. Experimental Setup

In this section, we first conduct a simulation experiment on the influence of the label flip-based poisoning attack on the federated learning system, and then we evaluate the proposed malicious model detection mechanism in the handwritten digit recognition task. All experiments are conducted using Tensorflow 2.3.1 and scikit-Learn 0.23.2 [30] on a Windows 10 Server working environment, which is Inter Core i5-7500 CPU, 8RAM.

In this experiment, we use the recognized data sets Mnist [31] and Fashion_Mnist [32]. The Mnist data sets are widely used handwritten digit recognition data sets, usually used for the performance evaluation of image classification algorithms in the computer vision field. There are ten number categories in this data set, from number 0 to number 9. The Mnist data set includes 70 000 grayscale images with a resolution of 28 $\times$ 28, a training set for training the model with 60 000 images, and a test set for evaluating the model with 10 000 images.

Fashion_Mnist is an extended version of Mnist. The Fashion_Mnist clothing data set contains 70 000 grayscale images, including a training set of 60 000 examples and a test set of 10 000 examples. Each example is a 28 $\times$ 28 grayscale image, including different types, such as T-shirt, dress, ankle boot, etc.

B. Experiment Results

We randomly divide the two data sets into 50 subdata sets, respectively, and sent them to each participant. The malicious participant performs label flipping on the local subdata sets before training the model. The poisoning attack is performed by flipping the label of the training data set, but the test data set is not flipped. We simulated the training results of malicious participants on two data sets, respectively. As shown in Figs. 3 and 4, as the number of training rounds increases, the loss of the malicious participant model will gradually decrease, and the model will converge. However, when testing on a local test data set, as the number of rounds increases, the model predicts that the loss of correct sample label pairs will gradually increase, and the prediction accuracy is almost close to 0.

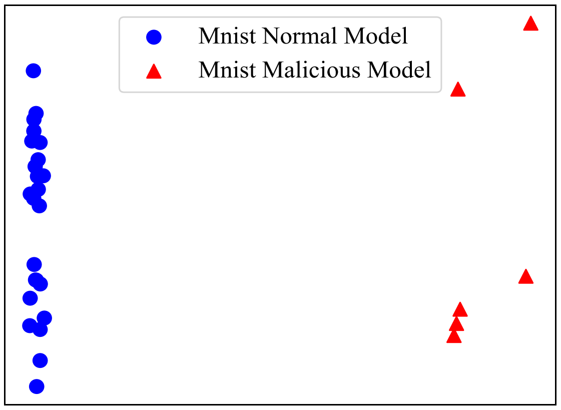


Fig. 7. Visual differences between normal and malicious model in Mnist.

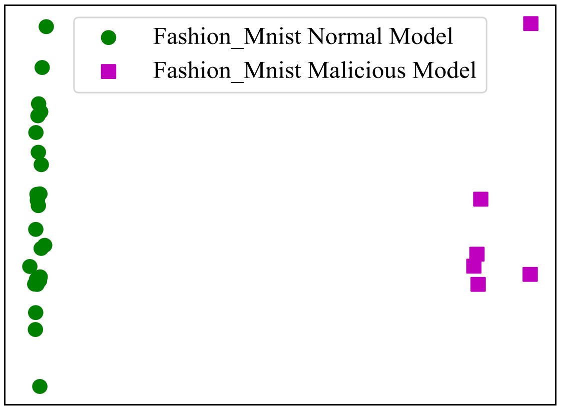


Fig. 8. Visual differences between normal and malicious model in Fashion_Mnist.

The server randomly selects participants and receives the local models uploaded by them, including 10% of the malicious participants. Malicious participants thirst for maximization of the success of the poisoning attack, they will expand the model parameters after training. We set the magnification of the malicious model to 5, 10, 15, and 20, respectively, to verify the influence of malicious models of different magnifications on the accuracy of the global model prediction. We verified on the Mnist data set. As shown in Fig. 5, as the magnification of the malicious model

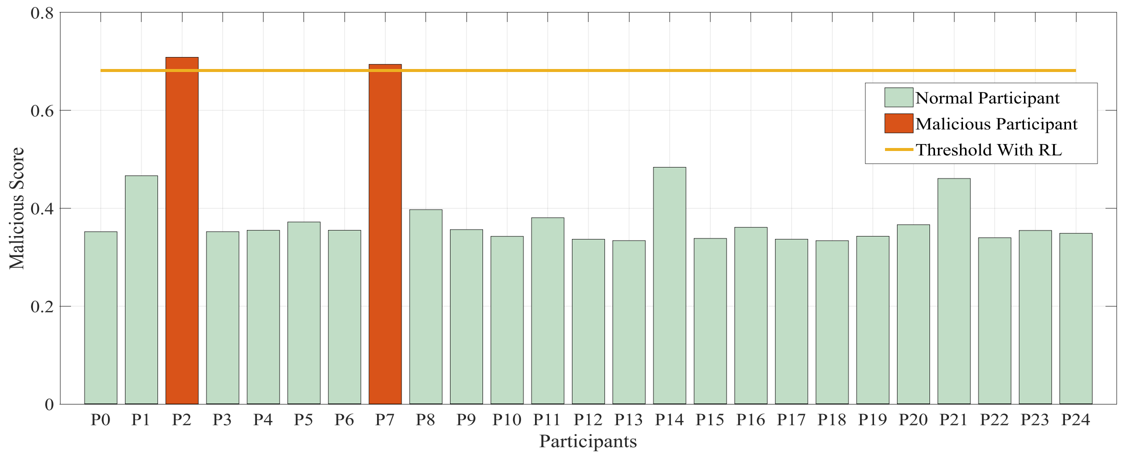


Fig. 9. Result of detection malicious model on the Mnist.

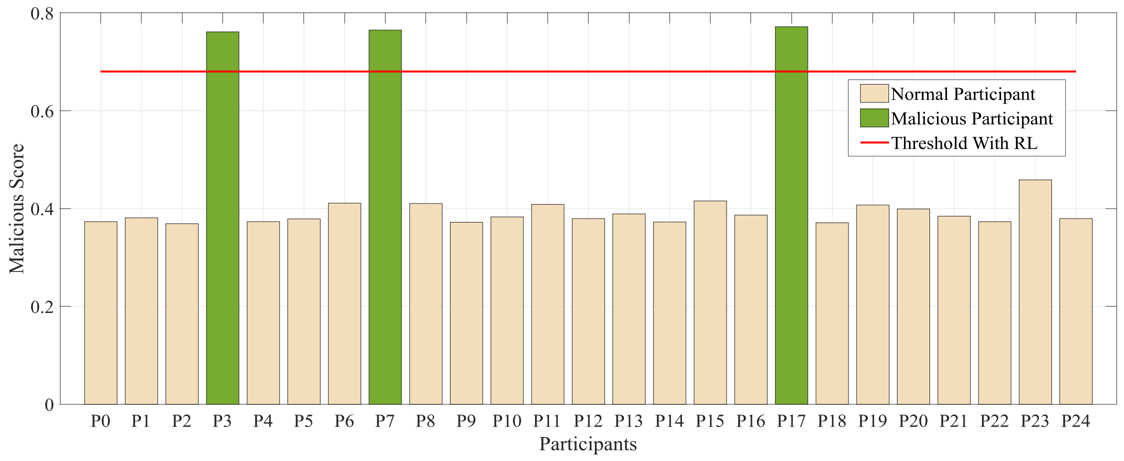


Fig. 10. Result of detection malicious model on the Fashion_Mnist.

increases, the accuracy when the global model converges will decrease.

Since the model set contains a malicious model, the aggregated global model will be affected in accuracy, recall, and F1. In order to verify the impact of the number of malicious participants on the accuracy of the global model convergence, we set the number of malicious participants to 10%, 20%, and 30% of the total number of participants during sampling and aggregation. As shown in Fig. 6(a)–(c), when there are 10% malicious participants in the federated learning system, the accuracy, recall, and F1 will converge to a lower when the global model reaches convergence. As the number of malicious participants increases to 20% and 30%, the accuracy, recall, and F1 of the global model decreases during convergence

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FN} + \text{FP}}, \tag{16}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{17}$$

$$F1 = \frac{2\text{TP}}{2\text{TP} + \text{FN} + \text{FP}}, \tag{18}$$

where TP is the number of predicting positive samples as positive labels; TN is the number of predicting negative samples as negative labels; FN is the number of predicting positive

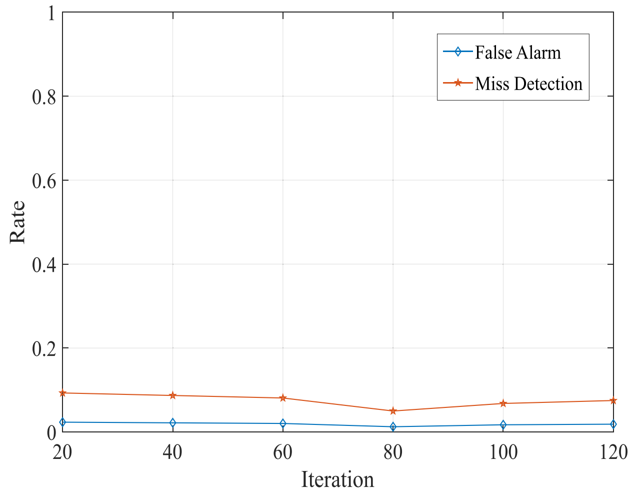


Fig. 11. False alarm rate and the missing detection rate.

samples as negative labels; and FP is the number of predicting negative samples as positive labels in the data set.

We will verify our malicious model detection mechanism. The intuition of our mechanism is that the model trained using the flipped label data set must be an outlier to the normally trained model. As shown in Figs. 7 and 8, the model uploaded

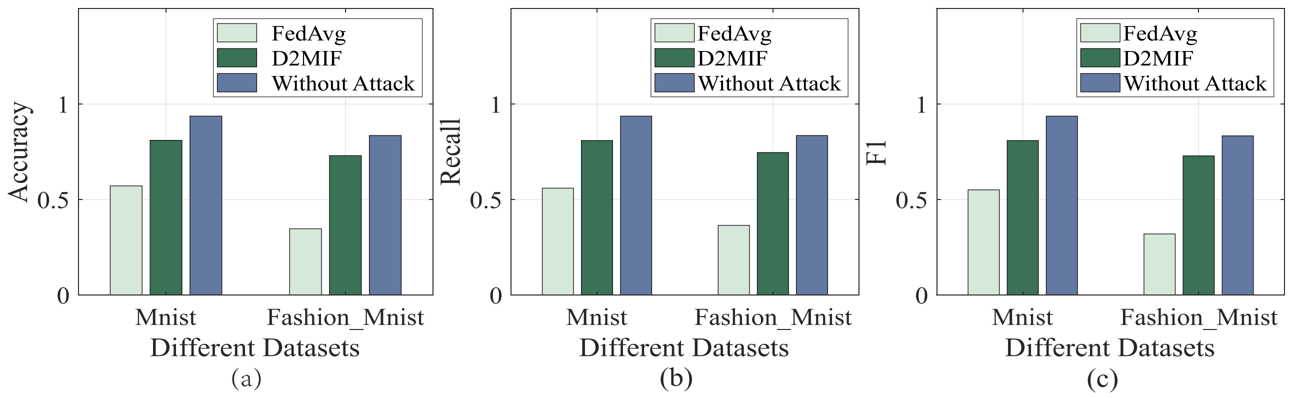


Fig. 12. (a) Accuracy, (b) Recall, and (c) F1 of global model with different approaches.

by 30 participants was randomly selected from the Mnist data set and Fashion_Mnist data set, respectively, to visually verify our conjecture on the premise of maintaining its original data characteristics. For the high numerical density of the normal model, the malicious value is an outlier that can be easily isolated.

We will conduct simulation for the malicious model detection scheme via iforests on Mnist data sets and Fashion_Mnist data sets. There are two vital parameters in iforests: 1) the number of itrees and 2) the number of samples to construct each itree. In our method, we set the former to 100, while the latter corresponds to the number of models in the model set received by the server, which is 10.

At the beginning of the federated aggregation, we selected 25 participants and received the model uploaded by them. We calculate the malicious score for each model by establishing an iforest. The threshold adjustment algorithm based on RL selects the threshold for detecting the malicious model. As shown in Figs. 9 and 10, the proposed method can effectively detect the malicious model in the model set, where the malicious score of malicious models is all greater than the threshold based on the RL, which is consistent with our hypothesis.

We conducted experiments on the false alarm rate and the missing detection rate of our proposed D2MIF on the Mnist data set. The false alarm rate R_F and the missing detection rate R_M are expressed as the following formula:

$$R_F = \frac{N_{\text{False}}}{N_{\text{Normal participants}}}, \tag{19}$$

$$R_M = \frac{N_{\text{Miss}}}{N_{\text{Malicious participants}}}, \tag{20}$$

where N_{False} is the number of normal participants detected as malicious participants; N_{Miss} is the number of malicious participants not detected; $N_{\text{Normal participants}}$ is the number of all normal participants; and $N_{\text{Malicious participants}}$ is the number of all malicious participants. As shown in Fig. 11, we set 10% of malicious participants in the federated learning system. As the number of iterations increases, the false alarm rate of our proposed D2MIF for malicious participants is about 7%, and the missing detection rate is about 3%.

As shown in Fig. 12(a)–(c), we set 10% of malicious participants in the federated learning system to evaluate the average accuracy, average recall, and average F1 of the global model of our D2MIF and the baseline approach FedAvg on the Mnist data set and Fashion_Mnist data set, respectively. When the global model converges, due to the existence of poisoning models generated by the poisoning data set training uploaded by malicious participants, which will seriously affect the performance of the global model. The accuracy of FedAvg can only reach 54.1% and 34.7%, while D2MIF can reach 81.1% and 72.9%; the recall of FedAvg can only reach 55.9% and 36.5%, while D2MIF can reach 80.9% and 74.6%; and F1 of FedAvg can only reach 55.1% and 32.0%, while D2MIF can reach 80.8% and 72.9%. Because of the existence of false alarm and missing detection, the accuracy, recall, and F1 of the global model cannot completely converge to the situation where there is no attack.

VI. Conclusion

In this article, we proposed a malicious model detection mechanism based on the iforest, named D2MIF, for the federated learning-empowered AIoT. In D2MIF, the model uploaded by each participant is regarded as the leaf node of the itree, and the corresponding malicious score is calculated for each node. Then, we use the RL algorithm to set a dynamically adjusted threshold for filtering malicious models. If the malicious score of a model is greater than the threshold, then this model is a malicious one and the corresponding participant will be eliminated from the federated learning process. The experiments show that the proposed D2MIF can effectively detect malicious models and significantly improve the global model accuracy in federated learning-empowered AIoT.

References

[1] H. Nguyen, K. Tran, X. Zeng, L. Koehl, P. Castagliola, and P. Bruniaux, "Industrial Internet of Things, big data, and artificial intelligence in the smart factory: A survey and perspective," in *Proc. ISSAT Int. Conf. Data Sci. Bus. Financ. Ind.*, Da Nang, Vietnam, 2019, pp. 72–76.

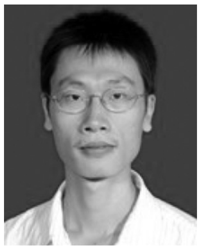
[2] J. Varghese, S. K. Vargheese, and E. Peter, "A study on artificial intelligence of things: Techniques and applications," *J. Compos. Theory*, vol. 8, no. 3, pp. 888–896, 2020.

[3] J. Zhou, Y. Wang, K. Ota, and M. Dong, "AAIoT: Accelerating Artificial Intelligence in IoT Systems," *IEEE Wireless Commun. Lett.*, vol. 8, no. 3, pp. 825–828, Jun. 2019.

[4] M. Al-Rubaie and J. M. Chang, "Privacy-preserving machine learning: Threats and solutions," *IEEE Security Privacy*, vol. 17, no. 2, pp. 49–58, Mar./Apr. 2019.
[5] U. Butt *et al.*, "A review of machine learning algorithms for cloud computing security," *Electronics*, vol. 9, no. 9, p. 1379, 2020, doi: 10.3390/electronics9091379.
[6] H. Lin, S. Garg, J. Hu, X. Wang, M. J. Piran, and M. S. Hossain, "Privacy-enhanced data fusion for COVID-19 applications in intelligent Internet of medical things," *IEEE Internet Things J.*, early access, Oct. 22, 2020, doi: 10.1109/JIOT.2020.3033129.
[7] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016. [Online]. Available: arXiv:1610.05492.
[8] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5986–5994, Jul. 2020.
[9] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE J. Sel. Areas Commun.*, early access, Dec. 2, 2020, doi: 10.1109/JSAC.2020.3041404.
[10] S. Li, Y. Cheng, W. Wang, Y. Liu, and T. Chen, "Learning to detect malicious clients for robust federated learning," 2020. [Online]. Available: arXiv:2002.00211.
[11] H. Lin, J. Hu, W. Xi, M. F. Alhamid, and M. J. Piran, "Towards secure data fusion in industrial IoT using transfer learning," *IEEE Trans. Ind. Informat.*, early access, Nov. 17, 2020, doi: 10.1109/TII.2020.3038780.
[12] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Min.*, Pisa, Italy, 2008, pp. 413–422.
[13] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. 35th Int. Conf. Mach. Learn.*, Stockholm, Sweden, 2018, pp. 1587–1596.
[14] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, Los Angeles, CA, USA, 2017, pp. 118–128.
[15] D. Cao, S. Chang, Z. Lin, G. Liu, and D. Sun, "Understanding distributed poisoning attack in federated learning," in *Proc. IEEE 25th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Tianjin, China, 2019, pp. 233–239.
[16] Y. Zhao, J. Chen, J. Zhang, D. Wu, M. Blumenstein, and S. Yu, "Detecting and mitigating poisoning attacks in federated learning using generative adversarial networks," *Concurrency Comput. Pract. Exp.*, to be published, doi: 10.1002/cpe.5906.
[17] C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," 2018. [Online]. Available: arXiv:1808.04866.
[18] S. Li, Y. Cheng, Y. Liu, W. Wang, and T. Chen, "Abnormal client behavior detection in federated learning," 2019. [Online]. Available: arXiv:1910.09933.
[19] L. Zhao *et al.*, "Shielding collaborative learning: Mitigating poisoning attacks through client-side detection," *IEEE Trans. Dependable Secure Comput.*, early access, Apr. 14, 2020, doi: 10.1109/TDSC.2020.2986205.
[20] J. Tan, Y.-C. Liang, N. C. Luong, and D. Niyato, "Toward smart security enhancement of federated learning networks," *IEEE Netw.*, vol. 35, no. 1, pp. 340–347, Jan./Feb. 2021.
[21] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *IEEE Wireless Commun.*, vol. 27, no. 2, pp. 72–80, Apr. 2020.
[22] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Proc. 25th Eur. Symp. Res. Comput. Security (ESORICS)*, Guildford, U.K., 2020, pp. 480–501.
[23] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Stat.*, Lauderdale, FL, USA, 2017, pp. 1273–1282.
[24] P. Kairouz *et al.*, "Advances and open problems in federated learning," 2019. [Online]. Available: arXiv:1912.04977.
[25] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proc. 36th Int. Conf. Mach. Learn.*, Los Angeles, CA, USA, 2019, pp. 634–643.
[26] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. 23th Int. Conf. Artif. Intell. Stat.*, Palermo, Sicily, Italy, 2020, pp. 2938–2948.
[27] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Trans. Knowl. Discov. Data*, vol. 6, no. 1, pp. 1–39, 2012.
[28] V. Francois-Lavet, H. Peter, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," 2018. [Online]. Available: arXiv:1811.12560.
[29] J. Wang, J. Hu, G. Min, A. Y. Zomaya, and N. Georgalas, "Fast adaptive task offloading in edge computing based on meta reinforcement learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 242–253, Jan. 2021.
[30] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.
[31] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[32] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017. [Online]. Available: arXiv:1708.07747.

Wenxin Liu received the bachelor's degree in information security from Xi'an University of Posts and Telecommunications, Xi'an, China, in 2019. He is currently pursuing the master's degree with the School of Mathematics and Informatics, Fujian Normal University, Fuzhou, China.

His research interests include deep learning, cyber security, and blockchain.



Hui Lin received the Ph.D. degree in computing system architecture from the College of Computer Science, Xidian University, Xi'an, China, in 2013.

He is a Professor with the College of Mathematics and Informatics, Fujian Normal University, Fuzhou, China, where he is currently a M.E. supervisor. He has published more than 50 papers in international journals and conferences. His research interests include mobile cloud computing systems, blockchain, and network security.

Xiaoding Wang received the Ph.D. degree from the College of Mathematics and Informatics, Fujian Normal University, Fuzhou, China, in 2016.

He is an Associate Professor with Fujian Normal University. His main research interests include network optimization and fault tolerance.

Jia Hu received the B.Eng. and M.Eng. degrees in electronic engineering from Huazhong University of Science and Technology, Wuhan, China, in 2006 and 2004, respectively, and the Ph.D. degree in computer science from the University of Bradford, Bradford, U.K., in 2010.

He is a Senior Lecturer of Computer Science with the University of Exeter, Exeter, U.K. He has published over 80 research papers within these areas in prestigious international journals and reputable international conferences. His research interests include edge-cloud computing, resource optimization, applied machine learning, and network security.

Dr. Hu has received the Best Paper Awards at IEEE SOSE'16 and IUCC14. He serves on the editorial board of *Computers & Electrical Engineering* (Elsevier) and has guest-edited many special issues on major international journals, including IEEE INTERNET OF THINGS JOURNAL, *Computer Networks*, and *Ad Hoc Networks*. He has served as the General Co-Chair of IEEE CIT'15 and IUCC'15, and the Program Co-Chair of IEEE ISPA'20, ScalCom'19, SmartCity'18, CYBCONF'17, and EAI SmartGIFT'2016.

Georges Kaddoum (Senior Member, IEEE) received the Ph.D. degree (Hons.) in signal processing and telecommunications from the National Institute of Applied Sciences, Toulouse, France, in 2008.

He published over 200 journal and conference papers and two pending patents.

Dr. Kaddoum is a recipient of the "Research Excellence Award of the Université du Quebec, 2018" and the "Research Excellence Award-Emerging Researcher" from ÉTS, in 2019. He is a co-recipient of the Best Papers Awards of IEEE PIMRC 2017 and IEEE WiMob 2014. He received the "Exemplary Reviewer Award" from IEEE TRANSACTION ON COMMUNICATION twice in 2015 and 2017. He is currently serving as an Associate Editor for the IEEE TRANSACTION ON INFORMATION FORENSICS AND SECURITY and the IEEE COMMUNICATION LETTERS. He held the ÉTS Research Chair in physical-layer security for wireless networks.

Md. Jalil Piran (Senior Member, IEEE) received the Ph.D. degree in electronics engineering from Kyung Hee University, Seoul, South Korea, in 2016.

Then, he continued his research carrier as a Post-Doctoral Fellow in Information and Communication Engineering with the Networking Laboratory, KyungHee University. He is an Assistant Professor with the Department of Computer Science and Engineering, Sejong University, Seoul. Prof. Piran published a substantial number of technical papers in well-known international journals and conferences in the area of *Intelligent Information and Communication Technology (IICT)*, specifically in the fields of *Wireless Communications and Networking*, *5G/6G*, *Internet of Things (IoT)*, *Data Science*, *Machine Learning*, and *Security*.

Dr. Piran received the IAAM Scientist Medal of the year 2017 for Notable and Outstanding Research in the field of New Age Technology & Innovation, in Stockholm, Sweden. Moreover, he has been recognized as the "Outstanding Emerging Researcher" by the Iranian Ministry of Science, Technology, and Research in 2017. In addition, his Ph.D. dissertation has been selected as the "Dissertation of the Year 2016" by the Iranian Academic Center for Education, Culture, and Research in the field of Electrical and Communications Engineering. In the worldwide communities, he is an Active Delegate from South Korea in the Moving Picture Experts Group (MPEG) since 2013, and an Active Member of the International Association of Advanced Materials (IAAM) since 2017.

Atif Alamri (Member, IEEE) received the Ph.D. degree in computer science from the School of Information Technology and Engineering, University of Ottawa, Ottawa, ON, Canada, in 2010.

He is currently a Full Professor with the Software Engineering Department, College of Computer and Information Sciences (CCIS), King Saud University (KSU), Riyadh, Saudi Arabia. He is one of the founding members of the Chair of Pervasive and Mobile Computing with CCIS, KSU, successfully managing its research program, which transformed the chair as one of the best chairs of research excellence in the college. His research interests include multimedia-assisted health systems, ambient intelligence, service-oriented architecture, multimedia cloud, sensor cloud, the Internet of Things, big data, mobile cloud, social networks, and recommender systems.