Security and Safety, Vol. 1, 2022003 (2022)
https://doi.org/10.1051/sands/2022003
sands.edpsciences.org

Research Article OPEN ACCESS

Intelligent Transportation

Efficient poisoning attacks and defenses for unlabeled data in DDoS prediction of intelligent transportation systems

Zhong Li[1,2,3], Xianke Wu[1], and Changjun Jiang[2,3,*]

[1] *College of Information Science and Technology, Donghua University, Shanghai 201620, China*
[2] *Key Laboratory of the Ministry of Education for Embedded System and Service Computing, Department of Computer Science, Tongji University, Shanghai 201804, China*
[3] *Shanghai Network Financial Security Collaborative Innovation Center, Tongji University, Shanghai 201804, China*

Received: 29 December 2021 / Revised: 17 February 2022 / Accepted: 28 February 2022 / Published online: 22 June 2022

Abstract Nowadays, large numbers of smart sensors (*e.g.*, road-side cameras) which communicate with nearby base stations could launch distributed denial of services (DDoS) attack storms in intelligent transportation systems. DDoS attacks disable the services provided by base stations. Thus in this paper, considering the uneven communication traffic flows and privacy preserving, we give a hidden Markov model-based prediction model by utilizing the multi-step characteristic of DDoS with a federated learning framework to predict whether DDoS attacks will happen on base stations in the future. However, in the federated learning, we need to consider the problem of poisoning attacks due to malicious participants. The poisoning attacks will lead to the intelligent transportation systems paralysis without security protection. Traditional poisoning attacks mainly apply to the classification model with labeled data. In this paper, we propose a reinforcement learning-based poisoning method specifically for poisoning the prediction model with unlabeled data. Besides, previous related defense strategies rely on validation datasets with labeled data in the server. However, it is unrealistic since the local training datasets are not uploaded to the server due to privacy preserving, and our datasets are also unlabeled. Furthermore, we give a validation dataset-free defense strategy based on Dempster–Shafer (D–S) evidence theory avoiding anomaly aggregation to obtain a robust global model for precise DDoS prediction. In our experiments, we simulate 3000 points in combination with DARPA2000 dataset to carry out evaluations. The results indicate that our poisoning method can successfully poison the global prediction model with unlabeled data in a short time. Meanwhile, we compare our proposed defense algorithm with three popularly used defense algorithms. The results show that our defense method has a high accuracy rate of excluding poisoners and can obtain a high attack prediction probability.

Keywords Poisoning attacks, Defenses, Multi-step DDoS prediction, Unlabeled data, Intelligent transportation systems

Citation Li Z, Wu X and Jiang CJ. Efficient poisoning attacks and defenses for unlabeled data in DDoS prediction of intelligent transportation systems. Security and Safety 2022; **1**: 2022003. https://doi.org/10.1051/sands/2022003

* Corresponding author (email: cjjiang@tongji.edu.cn)

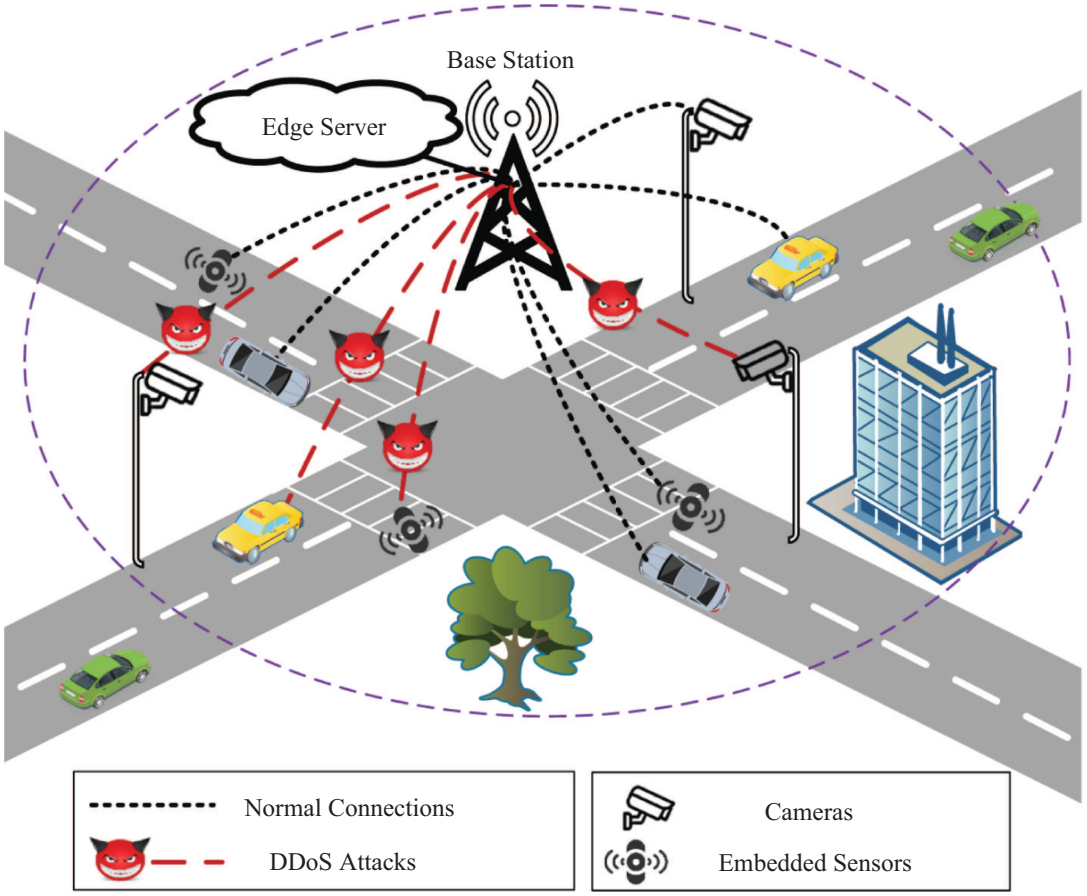

Figure 1. DDoS attacks to a base station in an intelligent transportation scenario

1 Introduction

With the development of intelligent transportation systems (ITS), more and more wireless sensors are used for information services. For example, road-side cameras, vehicular on-board sensors, induction coils and embedded road sensors are connected to base stations (BS) to do data processing or calculation as shown in Figure 1. As edge infrastructures, road-side base stations can provide convenient nearby services (such as road condition broadcast, driving alert support) to the mobile users, and meanwhile, they are the key bridges that connect thousands of sensors and backend servers. However, there are more and more smart sensors (*e.g.*, cameras) that can launch attacks to the base stations, especially distributed denial of services (DDoS) attacks. Since the number of sensors is very large, they are easy to become "zombies" to realize DDoS attacks. As illustrated in Figure 1, the wireless sensors in the communication range can establish normal connections with the base station. If there exist some malicious sensors that send a large amount of useless and interfering packets to the BS and finally launch DDoS attacks, the communications on the BS will be blocked and many normal connections will be disabled. Thus, the BS should have an ability of predicting whether DDoS attacks will happen in the future on the current BS so that it can take actions before a disaster occurs.

Up till now, there have been many research work about the detection of DDoS attacks, rather than prediction. Most of them considered DDoS as a type of single-step attack. However, in fact, a DDoS attack can be decomposed into multiple steps and executed in a series of strategic stages, posing a formidable security risk to the system [1]. Usually, a DDoS attack contains five steps [2]: (i) "IPsweep": sweeping the network IP from a remote site; (ii) "Sadmind Ping": making the attacker distinguish which hosts are vulnerable to intrude; (iii) "Break Into and Buffer Overflow": the attacker trying to break into the hosts which are under the sadmind service in the second step; (iv) "Installation Mstream Software": installing the trojan mstream DDoS software on hosts; (v) "Launch DDoS". If DDoS attacks can be

captured in the early stage, we can take actions in advance. This is what the defense system needs. Thus, researchers had proposed some machine learning (ML) based and correlation-based techniques to predict the probabilities of DDoS attacks, such as Bayesian network-based method [3], attack graph-based method [4] and hidden Markov model-based (HMM) method [5]. Although these methods have achieved good results by analyzing data packets, there still exist the following problems:

- First, the majority of such ML-based or correlation-based techniques are centralized methods, meaning that the BSs need to gather data from different areas and send them to the central cloud server. The requirement of large transmissions to the cloud server and privacy preservation awareness [6] propose challenges to the defense system if we directly apply previous methods to base stations in our city. A smart defense/attack prediction system on each road-side infrastructure and response with short delay are required.
- Second, even if an attack prediction system is put in the road-side base station, the BS has no feasibility of data aggregation in some areas with sparse communication flows due to little training data. Then, the model cannot be trained effectively, which further leads to the fact that the system would not predict DDoS attacks on time.

To solve the aforementioned problems, the federated learning (FL) occurs on us firstly. Therefore, based on the above previous studies, our paper adopts the latest and excellent hidden Markov model-based multi-step DDoS prediction model [5] combining FL, a type of decentralized ML paradigm. It can alleviate the pressure on BSs, and realize a secure and effective prediction system for determining whether there will be DDoS attacks on base stations. In FL, each BS can be seen as a participant. The participants can train their models (HMM) based on the local dataset and do not need to exchange the original privacy data with others. These participants upload their local model parameters to the cloud server, where the parameters will be aggregated to form a global model. The cloud server then distributes the global model to the participants and both sides repeat the above processes until the model converges.

Though the FL framework takes advantage of the collaborative training and avoids the leakage of sensitive privacy data, there still exists the vulnerability on data reliability. During a federated learning process, even we assure that the uploaded parameters are protected by various cryptography-based tamper-proofing technologies, we cannot guarantee all participants are trustful. Participants may upload interference parameters/models to the server due to vicious intentions, *i.e.*, launching poisoning attacks. If one malicious participant uploads its interference parameter to the server continuously, the global model will be poisoned and cannot converge [7, 8]. Then, the protection of the whole network will be reduced due to the poisoning attacks. Therefore, the uploaded parameters should be guaranteed to be valid.

Nowadays, there are many studies about the poisoning attacks [9–13] and defense strategies [14–19]. With respect to poisoning, the latest poisoning attacks based on GAN mainly point to the classifier/classification model with labeled data. Its principle is to generate data through a discriminator and a generator, and then to assign wrong labels to these data. This also means that the poisoner knows how to process the data to have a negative effect on the model. However, the poisoning attacks based on GAN cannot poison the prediction model that has no labeled data, since the GAN poisoner cannot know how to process data and cannot poison it by flipping the labels. Besides, among various prediction models, the HMM-based prediction model is a widely used, popular and excellent prediction model for multi-step DDoS prediction [5]. Thus, it is very meaningful to focus on the poisoning attack for unlabeled data in the HMM-based prediction model in this paper.[1] With respect to the defense, many studies about secure aggregation must have an assumption that the server has a validation dataset with labeled data to judge whether the uploaded local model is poisoned. However, in practice, this assumption is unrealistic since the local training datasets are not uploaded to the server due to privacy preserving, and our datasets are also unlabeled.

Thus, this is the first paper that considers the poisoning attacks on an unlabeled prediction model, while taking the validation dataset-free as a condition to design a defense strategy for multi-step DDoS

[1] Although there were both data poisoning methods and model poisoning methods in previous studies, our aim is to poison the HMM parameters, which is different from the previous classification model's poisoning. Compared with directly flipping the labels in the classification model, it is very difficult to execute the model poisoning on the transition matrix since it will lose poisoning direction in this matrix. Thus, in our paper, we give a data poisoning attack method which is easy to launch by attackers.

prediction in ITS under a federated learning framework. The main contributions of our paper are listed as follows:

- A reinforcement learning-based poisoning algorithm is proposed, which can not only poison the prediction model with unlabeled data successfully, but also can use accumulated experiences to speed up the poisoning speed continuously.
- A validation dataset-free defense strategy based on D–S (Dempster–Shafer) evidence theory is given with avoiding anomaly aggregation. Three different types of evidences are designed to guarantee the robustness of the aggregated model under limited knowledge owned by the global server.
- Through extensive experiments, the results show that our reinforcement learning-based poisoning method can successfully poison the global prediction model with unlabeled data in a short time. Besides, we compare our defense algorithm with Krum, Trimmed Mean and Median three algorithms. The results show that our defense method can achieve the best performance on defending poisoning attacks and guarantee the attack prediction probability of global model above 0.96.

The paper is organized as follows. The related studies are reviewed in Section 2. System architecture and some preliminaries are given in Section 3. Then, a reinforcement learning-based poisoning algorithm for the HMM model is proposed in Section 4. Next, a validation dataset-free defense strategy based on D–S evidence theory is designed to guarantee the reliability of the aggregated global model in Section 5. Finally, we give experiments and analysis in Section 6 and conclude the paper in Section 7.

2 Related work

The previous studies related to the subject of our paper involve three aspects. The first is related to the methods of multi-step DDoS attack prediction (Section 2.1). The second is related to the poisoning attacks in FL (Section 2.2). The third is related to the secure aggregation and defense approaches to the poisoning attacks in FL (Section 2.3).

2.1 Prediction methods of multi-step DDoS attacks

There are many related studies about DDoS detection by utilizing supervised or unsupervised machine learning technologies. Some reviews [20, 21] can be referred. Here we do not repeat them. Recently, with respect to the research of multi-step DDoS prediction, there are also some classic studies about DDoS prediction. For example, Kavousi and Akbari [3] presented a new algorithm to mine attack behavior patterns and generated correlation rules from a large number of intrusion alerts without prior knowledge through using a Bayesian causality mechanism. Wang *et al.* [4] proposed an approach to generate a global attack graph based on a heuristic searching strategy which would improve the attack search process. Holgado *et al.* [5] proposed a method based on a hidden Markov model (HMM), in which intrusion alerts were used to predict the multi-step DDoS attacks. Though all above studies have high prediction accuracy of multi-step DDoS attacks, they still cannot avoid the problems of tremendous data transmission and privacy leakage brought about by the central training.

2.2 Poisoning attacks in FL

FL is vulnerable to the poisoning attacks since it is difficult to ensure the security of each participant. The poisoning attacks in FL can be categorized into two types: data poisoning and model poisoning [9]. The former can finally generate poisoned model updates through interfering the training data. The latter is to directly launch model poisoning by manipulating benign model updates. With respect to the research of data poisoning, for example, Xie *et al.* [10] and Fung *et al.* [11] described the poisoning attacks that flipped the labels of data, which would make the local training result inaccurate and further generate poisoned model updates. This finally leads to poor performance of the global model. With respect to the research of model poisoning, for example, Fang *et al.* [12] proposed a method to reduce the reliability of local models. The author made a process of crafting a local model as an optimization problem and then minimized the loss function in iterations to launch a poisoning attack.

Though different studies can launch poisoning attacks in FL, they usually assume that the attacker (the distrustful participant) knows the distribution of other local participants' datasets. It is impractical for attackers to observe the other local participants' datasets directly in FL because each participant trains its model secretly and only updates the model parameters. With later the generative adversarial network (GAN) appearing, Zhang *et al.* [13] utilized GAN to generate the same distribution samples as the other participants through the global model without knowing the prior knowledge of training dataset. Then the attacker assigns the incorrect labels to the generated samples and finally uploads its poisoning parameters for aggregation. As described in Section 1, the previous poisoning attacks mainly apply to the classifier/classification model with labeled data. It cannot poison our HMM prediction model without labeled data. In this paper, we propose a data poisoning method through a learning way to realize the poisoning for the prediction model.

2.3 Secure aggregation and defense approaches

The approaches against poisoning attacks can be traditionally classified into two types [14]. One is robust aggregation. The server side aggregates the local models according to a secure strategy instead of using an average-based aggregation method [15]. For example, Blanchard *et al.* [16] proposed a defense approach named Krum. This research selects the most representative parameters to update the global model. Chen *et al.* [17] aggregated the global model with a median value. Another solution is anomaly detection. This type of approach is based on similarity comparison among all the updated local models and then excludes some local models with great differences.

Nowadays, as more and more poisoning attacks appear, there are many new research works concentrating on how to build a robust FL framework. They do not strictly divide the approaches into the above two types. For example, Tan *et al.* [14] proposed a verify-before-aggregate (VBA) method that removed the model if it could not achieve a good result after verifying. However, it may lead to some reliable clients taken away because the training dataset is insufficient and the local model cannot converge in time. Kang *et al.* [18] described a concept of reputation in FL. It can distinguish the non-benign participants and select reliable ones to aggregate. However, when there is a large number of clients, it will have huge computational costs. What's more, it is difficult to determine the reputation threshold. Fang *et al.* [12] proposed two defense methods based on the error rate and the loss function to reject the suspicious update parameters by verifying them using a validation dataset. Taheri *et al.* [19] trained a GAN network through using a validation dataset to obtain an anomaly detector and set an anomaly threshold. If the anomaly value of a participant is higher than the threshold, the local model in this participant would be excluded in aggregation. However in fact, a validation dataset does not always exist in the server side, especially considering the privacy preserving. Unlike the aforementioned approaches, we propose in this paper a defense method by utilizing D–S evidence theory to aggregate a robust global model.

3 System architecture and preliminaries

As stated in Section 1, an anti-poisoning FL strategy for multi-step DDoS prediction on base stations in intelligent transportation systems will be proposed in our paper. Thus, first we give a system architecture of our proposed FL-based multi-step DDoS prediction (Section 3.1). Under this architecture, we also show how the poisoning attack can impact the DDoS prediction preliminarily. Second, under this architecture, we introduce the latest and novel study [5] about the hidden Markov model-based multi-step DDoS prediction into our FL-based architecture (Section 3.2). This introduction is necessary since we will show how this HMM-based DDoS prediction model cannot resist the poisoning attack in FL framework in the following sections.

3.1 FL-based architecture for multi-step DDoS prediction on base stations in intelligent transportation systems

Figure 2 describes our FL-based multi-step DDoS prediction architecture. The architecture can be divided into two parts: participants and a central server. We assume that different BSs have different scales of

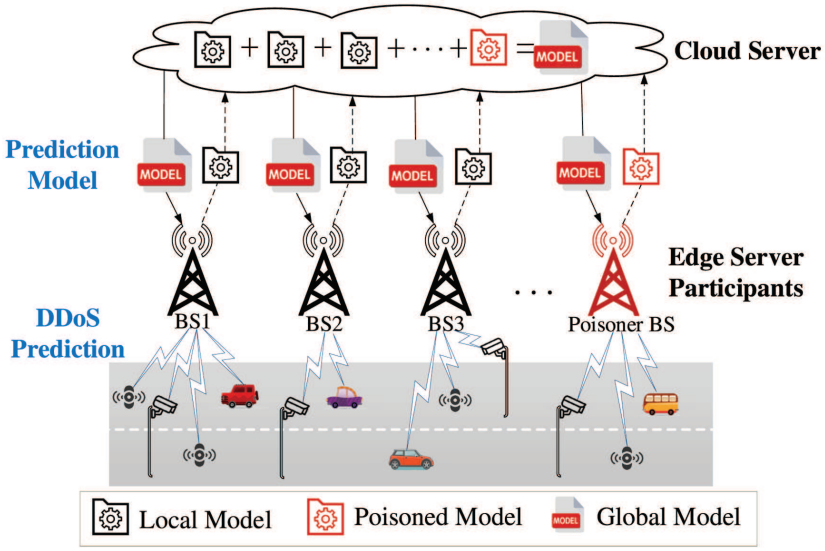

Figure 2. FL-based model for multi-step DDoS prediction

training datasets and each participant (*i.e.*, BS) cannot know the models of other participants in the federated learning framework. The BS collects the sensor-BS's communication traffics continuously and trains its local prediction model to monitor whether it will suffer DDoS attacks in its communication range. Then all the participants in different geographical areas update their local models to the central server. The server aggregates these local models to a global model and further distributes the global model to each BS to realize the DDoS prediction at BSs. After predicting the DDoS attacks, each BS updates cumulative data into its training dataset through DDoS observation subsequently. Besides, we assume that the server has no validation dataset.

Under this architecture, we need to clarify two issues.

- First, it is related to the prediction model deployed on each BS. In this paper, we use an HMM-based multi-step DDoS prediction method according to literature [5], which will be described briefly in Section 3.2. It requires each BS to be equipped with a usually used packet sniffer, named "Snort",[2] which can detect abnormal packets and produce relevant alerts. These alerts provide a common language to describe the suspicious events and finally correspond to some alert tags. These alert tags include "Information", "Remote Users", "Remote Attackers", "Denial of Service", and "Buffer Overflow". Each BS utilizes these generated alerts to train its local model.
- Second, it is related to the poisoning attacks. As shown in Figure 2, if a participant updates a poisoned local model, the global model will become inaccurate, which further leads to poor prediction performance for each BS. Next, in Section 4, we show a designed poisoning method which can realize the poisoning for HMM prediction model in detail. Further, in Section 5, we give a defense method to complete a robust aggregation in the server.

3.2 An HMM-based multi-step DDoS prediction method

According to literature [5], an HMM-based multi-step DDoS prediction model is built as follows.

A discrete first-order HMM is defined as a five tuple λ, having

$$\lambda = (S, \Sigma, E, Q, \Pi). \tag{1}$$

These parameters, which describe the HMM and their correspondence to our purposes, are explained below:

[2] https://www.snort.org.

- S describes a set of HMM states corresponding to a DDoS attack's several sub-steps. As stated in Section 1, there are five types of hidden states: "IPsweep" (s_1), "Sadmind Ping" (s_2), "Break Into and Buffer Overflow" (s_3), "Installation Mstream Software" (s_4), and "Launch DDoS" (s_5). Thus, we have

$$S = \{s_1, s_2, s_3, s_4, s_5\}. \tag{2}$$

- Σ is a set of observations based on the alerts, which are defined by five types of alert tags described in Section 3.1: "Information" (σ_1), "Remote Users" (σ_2), "Remote Attackers" (σ_3), "Denial of Service" (σ_4), and "Buffer Overflow" (σ_5). We have

$$\Sigma = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5\}. \tag{3}$$

- E is a transition probability matrix describing the transitions between the states in set S. We define a 5×5 transition probability matrix shown below:

$$E = \{\{e_{ij}\}_{5\times5} | e_{ij} = P(x_t = s_j | x_{t-1} = s_i)\}, \quad \forall i, j \in [1, 5], \tag{4}$$

  where a hidden process, which is represented by random variable x_t and corresponds to a state at time t in the Markov chain. Notation e_{ij} denotes the transition probability from state s_i to state s_j.
- Q is an observation probability matrix, with size of 5×5. We have

$$Q = \{\{q_{ik}\}_{5\times5} | q_{ik} = P(o_t = \sigma_k | x_t = s_i)\}, \quad \forall k \in [1, 5], \tag{5}$$

  where an observation process, which is represented by random variable o_t and corresponds to alert tags sent from Snort. Each element q_{ik} in Q denotes the probability of observation symbol σ_k while in state s_i.
- Π represents an initial distribution vector for each HMM state. It indicates the probability that the intrusion can start in each state. We have

$$\Pi = \{\pi_1, \pi_2, \pi_3, \pi_4, \pi_5\}. \tag{6}$$

  Here, we set each element in Π to 0.2 uniformly.

At the beginning of HMM training, the observation sequences and the hidden states are used to calculate the transition probability matrix E and observation probability matrix Q through supervised training. That is to say, it calculates the frequency of each observation in each HMM state and adopts the maximum likelihood method to obtain HMM parameters on λ.

Let O represent a series of observations with variable length, whose elements belong to the set of observations Σ. After obtaining the HMM parameters of λ, we can use Forward or Backward algorithm to calculate the observation sequence probability $P(O|\lambda)$. Also, when given the λ and observation sequence, we can get the most likely corresponding state using Viterbi algorithm. According to literature [5], a final attack probability PA is calculated based on the number of alerts triggered by Snort. It reflects the progress of the current DDoS attack. The probability PA is given below:

$$\mathrm{PA} = \frac{\sum_{t=0}^{\mathrm{Num}} \max[P(x_t = s_i | O, \lambda)]}{\mathrm{Num}}, \quad \forall i \in [1, 5], \tag{7}$$

in which Num is calculated by using the following Equation (8):

$$\mathrm{Num} = \sum_{j=0}^{5} f(\omega_j), \tag{8}$$

$$f(\omega_j) = \begin{cases} n_j, & \text{if } j \geq \text{the index } i \text{ of current state or } n_j > \mu_j, \\ \mu_j, & \text{otherwise,} \end{cases} \tag{9}$$

where n_j is the number of alerts at state s_j in the training dataset, and μ_j is the counter of number of alerts for the attack in progress detected in each state which can dynamically adjust the number of alerts based on the current intrusion.

The DDoS attack has the multi-step characteristic. It means that the DDoS attacker needs to make a series of preparations to launch the final DDoS attack. In Equation (7), the numerator denotes the sum of the probability of the most likely state when receiving the alert at each time, while the denominator denotes the total alert length of a DDoS attack. If the HMM model can predict the state accurately, then when each alarm comes, the value of probability will be close to 1. As the number of alerts increases, the DDoS attack is approaching to the final DDoS launching stage. Therefore, the attack probability PA indicates the degree/extent from the current time when the alert is received to the final DDoS launching stage. A high attack probability PA means that it is closer to the final stage of a DDoS attack, and the base station needs to be prepared for defense.

4 Reinforcement learning-based poisoning attacks for HMM-based multi-step DDoS prediction

As stated in Section 1, the federated learning could suffer from poisoning attacks and lead to the global model with low accuracy. In this section, we present a reinforcement learning-based poisoning algorithm in detail. First, we give the threat model and assumptions (Section 4.1). Then, we define a reinforcement learning model of poisoning attacks (Section 4.2). Finally, we design a reinforcement learning-based poisoning algorithm for HMM-based multi-step DDoS prediction (Section 4.3).

4.1 Threat model and assumptions

In the above FL-based multi-step DDoS prediction architecture, one or more poisoners would choose some BSs to invade and participate in the federated learning as normal participants. These poisoners not only make their local BSs unable to predict DDoS attacks normally, but also want to poison the global model which can further impact other BSs' predictions. To avoid the model deviating from the original one suddenly, a poisoner tries to poison its local model by injecting abnormal alert data into the BS's local training dataset. A poisoner can inject different alerts, which would change the transition probability matrix in HMM. When these alerts are used for training, the prediction model may not be accurate.

4.2 Reinforcement learning model of poisoning attacks

In each participant, its local training dataset consists of a sequence of alert tags. The five types of alert tags have been described in Equation (3). As a poisoner does not know how the alert tags impact the model and wants to poison the local data with a best policy, our poisoning method can be formulated as a reinforcement learning model.

As shown in Figure 3, we regard a poisoner as an agent that interacts with the environment and takes an action according to the current state and reward. A poisoner can obtain the uploaded parameters so that it can evaluate its action under the current state and further can feedback a reward to guide the policy learning. After multiple iterations, the agent can make actions that maximize accumulated rewards based on the current state and learn the optimal policy. We give definitions of three important elements as follows.

4.2.1 State space definition

Since a tremendous state space is not friendly for an agent to learn, we select several representative environment characteristics to describe the current state. Here, a state can be defined as

$$\widehat{s}_t = \{\widehat{sa}_1, \widehat{sa}_2, \widehat{sa}_3, \widehat{sa}_4, \widehat{sa}_5, \widehat{sc}\}_t. \tag{10}$$

Notation $\widehat{sa}_j$, $j \in [1, 5]$, indicates the proportion of j-th type of alert tags to the total number of alert tags in the training dataset. The value of $\widehat{sa}_j$ under the current t-th step can be calculated as

$$\widehat{sa}_j = \frac{n_j}{\sum_{i=1}^{5} n_i}. \tag{11}$$

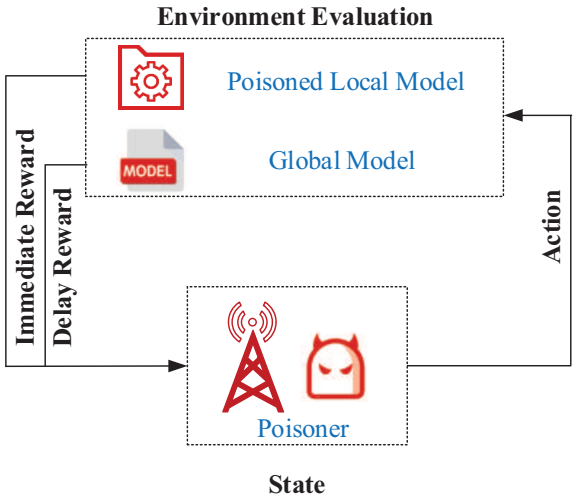

Figure 3. Poisoning model based on reinforcement learning

Besides, notation $\widehat{sc}$ denotes the last alert tag in our alert sequence of the training data. Since the poisoning attack needs to affect the HMM model ultimately and the observation sequence of HMM is a time-dependent sequence, the poisoning agent needs to try to inject interferential alerts into the tail of the observation sequence based on the last alert tag in the current observation sequence. At different states, the last alert tag of the current observation sequence will be different. Based on the last alert tag $\widehat{sc}$, the poisoning agent can choose two actions which are related to $\widehat{sc}$. The detailed work process is shown in the following action space definition. Thus, we set $\widehat{sc}$ as a component of the state vector in Equation (10).

4.2.2 Action space definition

We define an action space as $\{a_{\text{keep}}, a_{\text{transit}}\}$. Let notation a_t denote an agent selecting one of the actions in the above action space at time step t. In our paper, to speed up the process of poisoning, when an agent performs an action, 10 single/10 pairs of alerts will be injected to the local alert sequence of the training data. Action a_{keep} represents the agent injecting the same 10 alerts as the last alert (*i.e.*, $\widehat{sc}$) at the end of the alert sequence. Action a_{transit} denotes transiting to the other alert tag. The agent randomly chooses one of the remaining four alert tags except the last alert tag $\widehat{sc}$ as a transiting alert. Then the agent injects 10 pairs of alert tags with the last alert tag $\widehat{sc}$ and the transiting alert in an alternating manner at the end of the alert sequence. For example, the local alert sequence of the training data is $\{\sigma_1, \sigma_1, \sigma_2, \sigma_3\}$. If the agent selects action a_{transit} the transiting alert is randomly chosen as σ_4. Then, the poisoned training data become $\{\sigma_1, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_3, \sigma_4, \ldots, \sigma_3, \sigma_4\}$. The above two types of actions would finally change the transition probability matrix in HMM.

4.2.3 Reward function definition

To verify whether the current state and action have an impact on model poisoning, we should judge whether the alert injection takes effect. Since a poisoner owns the local training dataset in the base station it can utilize this original dataset to verify the performance of model poisoning. Considering that the global model is aggregated and distributed periodically (*i.e.*, it will take some time), we use the local model and the global model together to formalize two reward functions so as to speed up convergence.

Immediate Reward: Before a global model is distributed, a poisoner can use its local model to judge whether a certain action takes effect under the current state. We take the final attack probability introduced in Equation (7) as the calculation criterion for evaluating the poisoning performance. An immediate reward RI_t can be calculated as follows:

$$\text{RI}_t = \text{PA}_{t-1}^I - \text{PA}_t^I, \tag{12}$$

Algorithm 1 Reinforcement Learning-Based Poisoning Algorithm

Input:
The original training dataset in each local base station
The descending degree tar of model performance
Output:
A trained neural network mode that can output policy π_{DDQN}

Initialize action-value functions Q and target action-value functions $\bar{Q}$ with random weights
Initialize replay memory to capacity M
repeat
  $t \leftarrow 0$, $R_t \leftarrow 0$
  With probability ε select a random action a_t
  With probability $1-\varepsilon$ select action $a_t = \arg\max_a Q(\widehat{s}_t, a; \theta)$
  Execute action a_t and obtain $\widehat{s}_{t+1}$
  Use the original training dataset to calculate RI_t according to Equation (12)
  if the agent obtains a global model at the current time step **then**
    Calculate RD_t according to Equation (13)
  end if
  Obtain R_t according to Equation (14)
  Store transition$(\widehat{s}_t, a_t, R_t, \widehat{s}_{t+1})$
  Sample random mini-batch transitions $(\widehat{s}_j, a_j, R_j, \widehat{s}_{j+1})$ from the replay memory
  Perform a gradient descent step with loss function $L(\theta)$ according to Equation (15)
  Update $\widehat{\theta} \leftarrow \theta$ for every τ steps
until $\sum_t \mathrm{RD}_t > tar$
Output a trained neural network model

where PA_{t-1}^I and PA_t^I are the attack probabilities calculated by the local model at time step $t-1$ and local model at the current time step t after taking a poisoning action.

Delayed Reward: When the server aggregates local models, it will distribute a global model to each participant. After a poisoner obtains the global model, the impacts of actions during this period can be measured. A delayed reward RD_t is defined as follows:

$$\mathrm{RD}_t = \mathrm{PA}_{t-1}^D - \mathrm{PA}_t^D, \tag{13}$$

where PA_{t-1}^D and PA_t^D are the attack probabilities calculated by the global model at time step $t-1$ and global model at the current time step t after taking a poisoning action. If the global model is not distributed at the current time step, we set $\mathrm{PA}_t^D = \mathrm{PA}_{t-1}^D$.

Finally, the total reward is defined as:

$$R_t = w_i \mathrm{RI}_t + w_d \mathrm{RD}_t, \tag{14}$$

where w_i, w_d are the weights of different rewards, and here we set $w_i = 0.5$, $w_d = 0.5$. When either the global model or local model is poisoned, the attack probabilities will decrease, and then the reward R_t will be positive. Both the immediate reward and the delayed reward increasing will optimize an agent's action choice.

4.3 Reinforcement learning-based poisoning algorithm for HMM-based multi-step DDoS prediction

Based on the above-mentioned reinforcement learning model, we can see that the state space is continuous and large. As a type of reinforcement learning method, double deep Q-network (DDQN) [22] is fit for tremendous state space and action space, which can accelerate the convergence speed. Thus, in this section, we give pseudo-codes of our reinforcement learning-based poisoning algorithm by utilizing DDQN in Algorithm 1.

In DDQN, two deep neural networks with the same structure are set up and used to approximate the action-value function. Notation $Q(\widehat{s}_t, a_t; \theta)$ with weight θ represents an online neural network, and notation $\bar{Q}(\widehat{s}_t, a_t; \bar{\theta})$ with $\bar{\theta}$ represents a target neural network. Then, we use M to denote the maximum capacity of the replay memory in DDQN. The replay memory contains an agent's experiences that will be used for optimizing the weights of Q and $\bar{Q}$.

In Algorithm 1, to describe the poisoning level uniformly, a threshold tar, $tar \in [0, 1]$, is set to denote the descending degree of model performance. For an agent, it takes the original training dataset in its local base station and threshold tar as the inputs of algorithm. First, we initialize relevant parameters of Q, $\bar{Q}$, and M (Lines 1–2). Next, at each step, the agent selects a random action a_t using ε-greedy method based on policy $\pi_{DDQN(\widehat{s}_t)}$ to balance the exploration and exploitation [23] (Lines 3–5). After the agent executes action a_t, the next state $\widehat{s}_{t+1}$ can be obtained (Line 6). Then, the agent can calculate the reward about the poisoning performance according to Equations (12)–(14) (Lines 7–11). Then the transition $(\widehat{s}_t, a_t, R_t, \widehat{s}_{j+1})$ will be stored in the replay memory (Line 12). In the next step, the mini-batch gradient descent method is used to optimize the weight θ of online neural network $Q(\widehat{s}_t, a_t; \theta)$ with a loss function $L(\theta)$ (Lines 13–15). In the mini-batch gradient descent method, the agent randomly selects a part of transitions from M for training. Assuming that the size of the training set is M_s. $L(\theta)$ is calculated as:

$$L(\theta) = \frac{1}{M_s} \sum_{j=1}^{M_s} \left(z_j - Q(\widehat{s}_j, a_j; \theta)\right)^2, \tag{15}$$

$$z_j = R_j + \gamma \bar{Q}(\widehat{s}_j, a_{\max}; \bar{\theta}). \tag{16}$$

where $a_{\max} = \arg\max_a Q(\widehat{s}_j, a; \theta)$. Notation γ is a discount factor, and the value is between 0 and 1. The parameter $\bar{\theta}$ is updated by cloning the value of θ for every τ steps [22].

The agent will not terminate a loop until the poisoning level tar is reached or the target network $\bar{Q}(\widehat{s}_t, a_t; \bar{\theta})$ converges (Line 16). Finally, we can get a trained neural network (Line 17) and use it to guide the action when a current state $\widehat{s}_t$ is given. The values of relevant parameters in this algorithm are given in Table 2 in the experiment.

5 A validation dataset-free defense strategy based on D–S evidence theory

From Section 4, a reinforcement learning-based poisoning can impact the global model for HMM-based multi-step DDoS prediction. Since the central server has no validation dataset, it is difficult to judge which participant is harmful. We only can use the uploaded information and the knowledge owned by the central server to infer the suspicious poisoners and obtain a robust global model. D–S evidence theory [24] proposed by Dempster and Shafer is an effective method to infer uncertain conception which is based on the synthetic formulas that integrate different evidences. For this reason, we utilize this theory to design our defense algorithm. Note that, our proposed strategy is under the condition that the number of poisoners is less than 1/3 of the total participants. It has been proved that only under this condition, a secure defense strategy can be found. Here, the server will execute the defense strategy and aggregate the global model periodically. We give the pseudo-codes of our proposed defense strategy in Algorithm 2. The input is each participant's local model. The output is an aggregated global model. In each round of aggregation, we first utilize D–S evidence theory to do evidence quantization (Lines 2–3). The details are shown as follows.

First, four tuple (Ω, EV, BPA, BEL) is defined under the framework of D–S evidence theory as follows.

- Ω denotes an identification frame, also named hypothetical space which contains all the propositions. Here, we define $\Omega = \{Benign, Harmful\}$, where $Benign$ presents a participant is not a poisoner, while $Harmful$ denotes a participant is a poisoner.
- EV denotes a set of evidences, which is an essential part of D–S evidence theory. The elements of the set are independent of each other. We define three evidences from different perspectives that can present the reliability level of the participants. The detailed definitions about the three evidences are given in the following Section 5.1.
- BPA describes a set of basic probability assignment functions. We assume L is the number of participants, and l denotes one of them. Then the BPA is given by

$$\text{BPA} = \{m_1^l(Benign), m_1^l(Harmful), \ldots, m_3^l(Benign), m_3^l(Harmful)\}. \tag{17}$$

  Notation $m_i^l(Benign)$ and $m_i^l(Harmful)$ represent probability assignment functions of the $Benign$ and $Harmful$ propositions, respectively, for the l-th participant under the i-th evidence.

Algorithm 2 A Validation Dataset-Free Defense Strategy Based on D–S Evidence Theory

Input:
Each participant's local model λ_l
Output:
An aggregated global model
for each round of aggregation **do**
  Use K-means to cluster the participants' local models
  Calculate three types of evidences according to Section 5.1
  Aggregate different evidences using Equation (25)
  for $\{Bel^l(Benign)\}$, $1 \leq l \leq L$ **do**
    Sort these values in a descending order and calculate the average value and the median
  end for
  Set two pointers: $trustful\ pointer$ and $distrustful\ pointer$
  Compare the average value with the median;
  Assign the large one to the trustful pointer and the small one to the distrustful pointer
  Aggregate the models with $Bel^l(Benign) > trustful\ pointer$ to obtain a global model
end for

- BEL denotes a set of belief functions. For the BEL of l-th participant, we have

$$\mathrm{BEL} = \{\mathrm{Bel}^l(Benign), \mathrm{Bel}^l(Harmful)\}. \tag{18}$$

Each element in the above set represents a total belief function for the $Benign$ or $Harmful$ proposition through fusion of the three different evidences for the l-th participant.

5.1 Evidence definitions

Here, we give three evidences, including clustering, upload frequency of local model and historical trust.

(1) **Evidence of Clustering**

In the federated learning, each participant will upload its parameter/model λ to the central server. For normal participants, their uploaded parameters will be similar. Thus, firstly we utilize a clustering method to group the uploaded local models. There are many different clustering algorithms, such as the hierarchical clustering, the density-based clustering algorithm and K-means algorithm. Compared with other algorithms, K-means algorithm can achieve good clustering effect with the advantage of easy implementation and low time complexity $O(N)$ [25]. Here, N denotes the number of samples in the dataset. Thus, the K-means algorithm is selected to cluster the participants' local models here.

Note that, λ is five tuple and it contains the element of matrix. Since the dimension of the matrix for each participant is the same, the Euclidean distance of the matrix is defined as the sum of the Euclidean distances of the corresponding elements. Besides, it is very important to determine the appropriate K value in the K-means algorithm. We refer to the latest improved K-means algorithm proposed by Zhang *et al.* [26]. Through using the maximum weight product method mentioned in this paper, we can obtain the optimal K value and initial cluster centers. Here, we do not repeat them.

After obtaining the clustering results, we can observe that every local model will be grouped into a cluster. Since we assume that the normal participants account for the majority, the normal participants are easy to be grouped together and the abnormal participants may be scattered in the clustering results. Therefore, we use the size of a cluster where a participant is located as the first evidence. Then, we have the following probability assignment functions:

$$m_1^l(Benign) = \frac{C_l}{\sum_l^L C_l}, \tag{19}$$

$$m_1^l(Harmful) = 1 - m_1^l(Benign). \tag{20}$$

The notation C_l represents the size of a cluster where the l-th participant is located.

(2) **Evidence of Upload Frequency**

A poisoner requires to upload the latest local model to evaluate the impact of the current state and action on the final poisoning target, resulting in the upload frequency higher than other normal participants. Thus, we use the upload frequency as the second evidence. Based on this evidence, we have the following probability assignment functions for the l-th participant:

$$m_2^l(Harmful) = \frac{f_l}{\sum_{l}^{L} f_l}, \tag{21}$$

$$m_2^l(Benign) = 1 - m_2^l(Harmful). \tag{22}$$

Here, f_l denotes the number of uploads in this round of aggregation.

(3) **Evidence of Historical Trust**

In the server side, it records whether a participant is a poisoner in each aggregation round. Therefore, we use the number of times a participant is judged to be a benign participant as the third evidence. Thus, we have the following probability assignment functions for the l-th participant:

$$m_3^l(Benign) = \frac{d_l}{D}, \tag{23}$$

$$m_3^l(Harmful) = 1 - m_3^l(Benign). \tag{24}$$

Here, notation D denotes the total number of aggregations and d_l represents the number of times the l-th participant is judged to be a benign participant historically.

Importantly, some excellent references [27–29] use the clustering method to defend the poisoning attacks. However, the clustering method is only one evidence of our proposed D–S evidence theory-based defense strategy in our paper. We have other evidences to strengthen the defense against poisoning attacks, and we also have an ablation study in the experiment (Table 3) to verify that our multi-evidences, not just clustering, are effective to defend the poisoning attacks in our paper.

5.2 D–S evidence fusion and robust model aggregation

After obtaining the above probability assignment functions for three evidences, we need to fuse these evidences and obtain the final belief function $\mathrm{Bel}^l(Benign)$ for the l-th participant (in Algorithm 2 Line 4). Since our two propositions are incompatible, the Bel function can be obtained by:

$$\begin{aligned} Bel^l(Benign) &= m_1^l(Benign) \oplus m_2^l(Benign) \oplus m_3^l(Benign) \\ &= (m_1^l(Benign) \oplus m_2^l(Benign)) \oplus m_3^l(Benign). \end{aligned} \tag{25}$$

Furthermore, we have

$$m_1^l(Benign) \oplus m_2^l(Benign) = \xi \cdot m_1^l(Benign) \cdot m_2^l(Benign), \tag{26}$$

where the coefficient ξ indicates the degree of conflict between evidences, having

$$\xi = (1 - m_1^l(Benign) \cdot m_2^l(Harmful) - m_1^l(Harmful) \cdot m_2^l(Benign))^{-1}. \tag{27}$$

Then, performing fusion with the third evidence is the same process as that in the above steps.

After obtaining $\mathrm{Bel}^l(Benign)$, we have to decide whether the l-th participant's local model can be aggregated into a robust global model. For $\{\mathrm{Bel}^l(Benign)\}$, $1 \le l \le L$, we sort these values in a descending order and calculate the average value and the median (Lines 5–7). Then, we set two pointers. One is called trustful pointer, the other is called distrustful pointer (Line 8). Through comparing the average value and the median, we assign the large one to the trustful pointer and the small one to the distrustful pointer

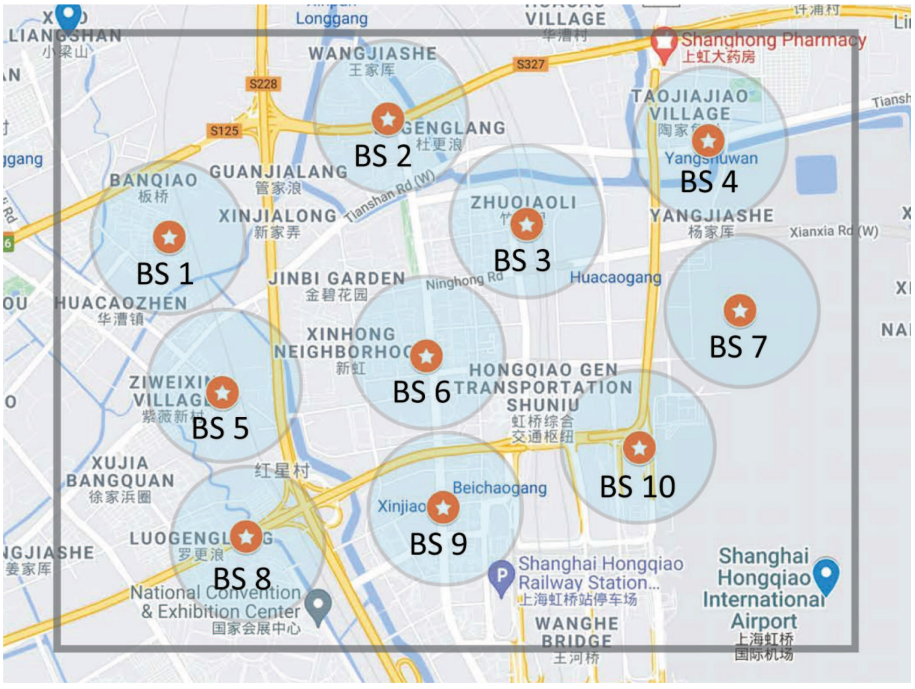

Figure 4. Experiment area

(Line 9). Finally, we exclude the participants whose $\mathrm{Bel}^l(Benign) \leq$ distrustful pointer and aggregate the models with $\mathrm{Bel}^l(Benign) \geq$ trustful pointer (Line 10). The participants' models with distrustful pointer $<\mathrm{Bel}^l(Benign) <$ trustful pointer are seen as suspicious. Thus, we have the final aggregated global model $\widetilde{\lambda}$:

$$\widetilde{\lambda} = \sum_l \frac{\mathrm{Bel}^l(Benign)}{\sum_l \mathrm{Bel}^l(Benign)} \lambda_l, \text{with } l \text{ satisfying } \mathrm{Bel}^l(Benign) \geq \text{trustful pointer}. \tag{28}$$

6 Performance evaluation and analysis

In this section, we evaluate the proposed poisoning and defense methods in detail. First, we give the simulation settings in Section 6.1. Then, we test the performance of our designed reinforcement learning-based poisoning attack method in Section 6.2. Finally, we evaluate the aggregation effect of our D–S evidence theory-based defense strategy in Section 6.3, and meanwhile, observe the performance of predicting DDoS attacks for global model in Section 6.4.

6.1 Simulation settings

We select an experiment area nearby Shanghai Hongqiao Airport, shown in Figure 4. The longitude is between [121.29031, 121.33821], and the latitude is between [31.19216, 31.2241]. We randomly set 10 base stations (participants of the federated learning) in the area. The detailed information of base stations is provided in Table 1. The communication ranges are marked by grey circles in Figure 4. In this experiment area, we use Python to generate 2000 random points to simulate the smart sensors (*e.g.*, cameras, embedded road sensors). Meanwhile, we use a simulator named SUMO to generate 1000 vehicles. The vehicles' traces are restricted to the road topology. Then, we utilize a dataset named DARPA2000[3] which contains multi-step DDoS attacks to carry out experiments. There are 420 alerts in the dataset. Under FL framework, we randomly select a number of attack alerts and assign them to the

[3] https://www.ll.mit.edu/ideval/data/2000data.html.

Table 1. Information of 10 base stations

Name	Longitude	Latitude	Communication radius (m)
Base Station 1	31.21177	121.29668	300
Base Station 2	31.21815	121.31052	300
Base Station 3	31.21247	121.3193	300
Base Station 4	31.21698	121.33077	300
Base Station 5	31.20336	121.30005	300
Base Station 6	31.20534	121.31294	300
Base Station 7	31.2078	121.33279	300
Base Station 8	31.19558	121.30155	300
Base Station 9	31.19717	121.31402	300
Base Station 10	31.20039	121.32642	300

Table 2. Parameters used in DDQN

Parameter	Value
The parameter ε of ε-greedy algorithm	0.8
Discount factor γ	0.9
Learning rate α	0.01
Maximum number of transitions M in replay memory	30,000
The size of mini-batch training set M_s	64
Descending degree tar	0.1
τ steps for updating the target network	200

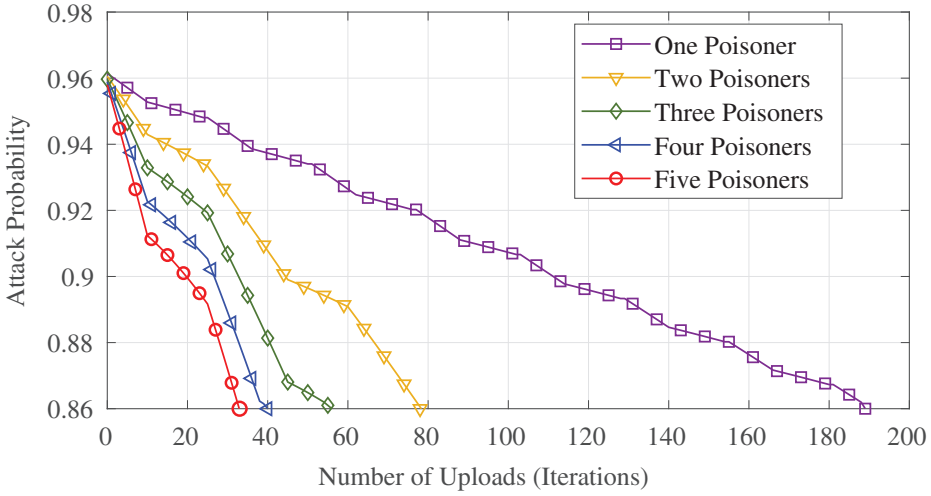

Figure 5. The different numbers of uploads under different numbers of poisoners when reaching to $tar = 0.1$

simulated sensors and vehicles through sampling with replacement. Each base station collects the alerts from sensors and vehicles within its communication range, which makes the training data of each base station different. We conduct our experiments on a 64-bit Win10 OS server equipped with Intel Core CPU i9-10920X 3.50 GHz processor, 64 GB of RAM.

6.2 Performance of poisoning attacks

The parameters used in DDQN are given in Table 2. Here, we set the poisoning target, that is, the descending degree tar of model performance as 0.1 to balance the poisoning efficiency and the poisoning effect. To verify the performance of our RL-based poisoning method is effective during the federated learning process, we show the different numbers of uploads that achieve the poisoning goal under different numbers of poisoners. The experiment results are shown in Figure 5. The x-axis is the number of uploads

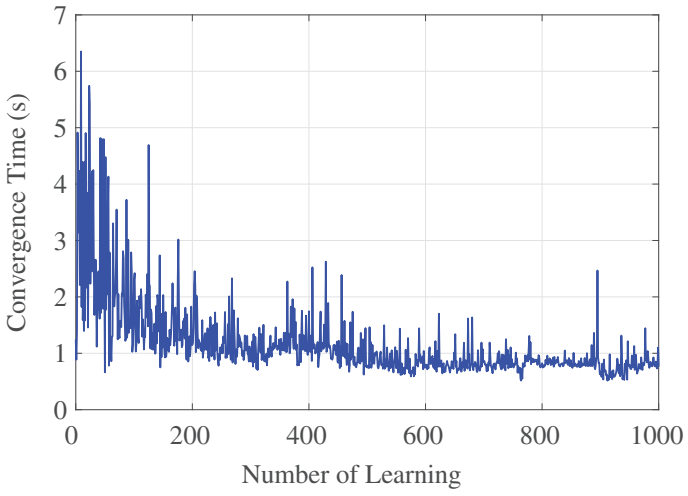

Figure 6. The convergence time of our poisoning method

Table 3. The attack probabilities of four defense methods under different numbers of poisoners

Defense methods	Number of poisoners				
	One	Two	Three	Four	Five
Trimmed mean [15]	0.9608	0.9324	0.9134	0.8942	0.8721
Krum [16]	0.9576	0.9576	0.9576	0.8610	0.8520
Median [17]	0.9423	0.9423	0.9423	0.9024	0.8512
Variant of our defense strategy (without evidence of clustering)	0.9615	0.9584	0.9460	0.9233	0.9135
Variant of our defense strategy (without evidence of upload frequency)	0.9621	0.9593	0.9463	0.9245	0.9087
Variant of our defense strategy (without evidence of historical trust)	0.9623	0.9597	0.9477	0.9217	0.9063
Our defense strategy (three evidences)	0.9627	0.9632	0.9640	0.9631	0.9512

of local model(s). The y-axis is the attack probability of the aggregated global model. In Figure 5, when there is only one poisoner, the number of uploads (iterations) is 185, which can make the attack probability decrease from 0.96 to 0.86 (*i.e.*, $tar = 0.1$). When the poisoners are two, three, four and five, the number of uploads are 76, 53, 38 and 32 in a gradual downward trend.

Besides, we test the convergence time that a poisoner completes a successful poisoning attack. The result is shown in Figure 6. The x-axis is the number of learning. A point in x-axis means a complete learning convergence process, *i.e.*, the attack probability is reduced by 10%. We can see that the poisoning method pays much time to achieve the poisoning goal at the beginning. Subsequently, the convergence time becomes low and steady due to cumulative learning experiences.

6.3 Performance of our defense strategy

In terms of the attack probability, we compare our proposed defense method with three other defense algorithms[4] (Trimmed Mean [15], Krum [16], Median [17]) under different numbers of poisoners. The experiment results are shown in Table 3.

From Table 3, we can find that our proposed defense strategy performs best than Trimmed Mean [15], Krum [16] and Median [17]. When there is only one poisoner, the three comparison defense methods can work effectively. The result of the Trimmed Mean is better than Krum and Median, as it can exclude the worst model and then take the mean. When there are two or three poisoners, the Krum and Median methods are well, and the values are 0.9576 and 0.9423, respectively. Trimmed Mean has a performance degradation since it takes mean and aggregates some poisoned parameters. When there are more than

[4] Trimmed Mean algorithm uses the average value to obtain a global model with pruning the largest and smallest local models; Krum algorithm is based on similarity to aggregate local models; Median algorithm utilizes the median value to obtain a global model.

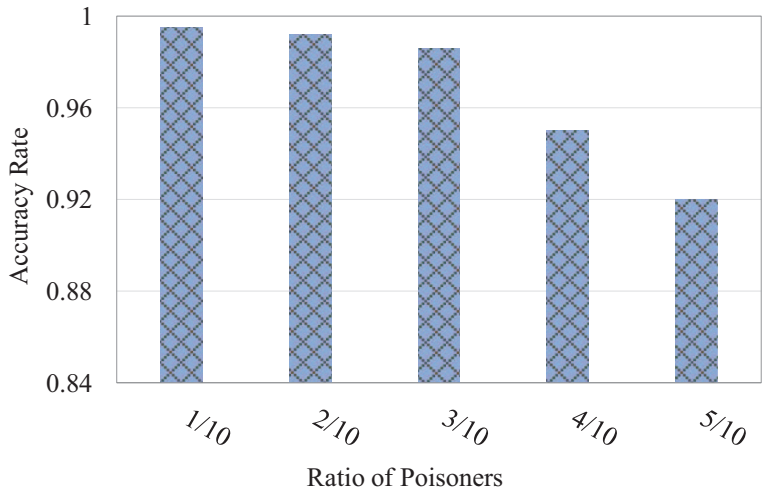

Figure 7. The accuracy rate of excluding poisoners in our proposed defense method

three poisoners, the attack probabilities of Krum, Median, and Trimmed Mean are bad at predicting the DDoS attacks. However, our defense method still has good results when the number of poisoners is less than 1/2, since our aggregation criteria considers many different evidences.

We also do an ablation study, in which our defense strategy retains any two evidences (*i.e.*, containing three variants). The results are given in the last four lines of Table 3. We can see that the attack probabilities of our defense strategy are also higher than the other three variants. When the number of poisoners is less than 3, the attack probabilities are close to the results that retaining all the three evidences. When the number of poisoners is more than 3, we find that selecting any two evidences performs worse than selecting all the three evidences. The ablation study verifies that selecting three evidences can achieve better results since it considers more factors to make the defense more rigorous.

Besides, we test our defense method's average accuracy of excluding poisoners in the federated learning when there are different ratios of poisoners. The results are shown in Figure 7. From Figure 7, we note that when the number of poisoners is less than 3/10 of participants, the accuracy rate of excluding poisoners is close to 100%. The gap to 100% is because the poisoned local model has little effect on the global model when the poisoning just starts, so that it cannot be directly eliminated. As the poisoning process progresses, our defense method can successfully detect and exclude it. When there are more than 3/10 poisoners, the accuracy rate of excluding poisoners decreases. Although the accuracy rate of excluding poisoners drops, the attack prediction probability in Table 3 remains relatively steady. This is because our defense method aggregates the local models cautiously. Through observing the experimental data, we find that our defense method will not select more than one-half of the participants for aggregation when the poisoners cannot be accurately detected.

6.4 Performance of predicting DDoS attacks for global model

In this paper, we implemented HMM to predict the intrusion probability of DDoS attacks as the number of alerts increases.

We set three performance metrics to evaluate the intrusion prediction performance. The first metric is the prediction accuracy rate (PAR). It is used to measure the proportion of the number of states that the model cumulatively predicts correctly. We have

$$\text{PAR} = \frac{\text{The Number of Correct Predictions}}{\text{Alerts}}, \tag{29}$$

where Alerts denotes the total number of incoming alerts. The second metric is the average state prediction probability (ASPP). It is used to evaluate how high the average probability that the model can predict

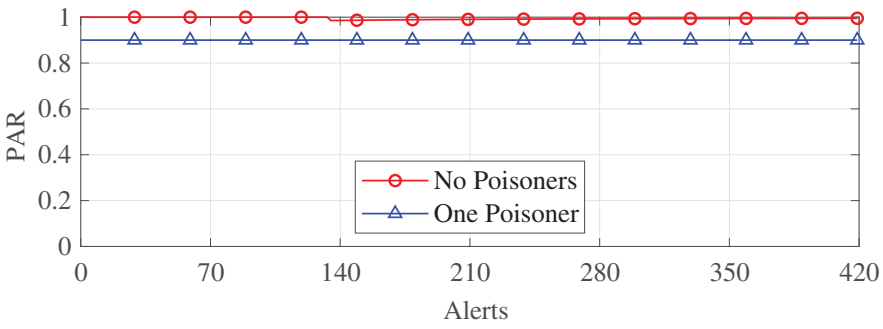

Figure 8. The performance of the prediction accuracy rate

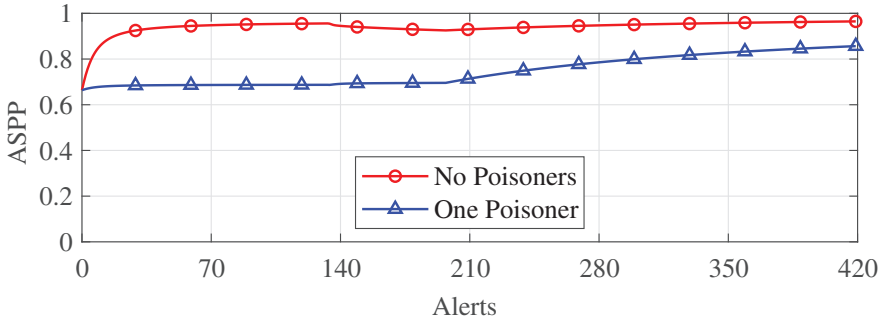

Figure 9. The performance of the average state prediction probability

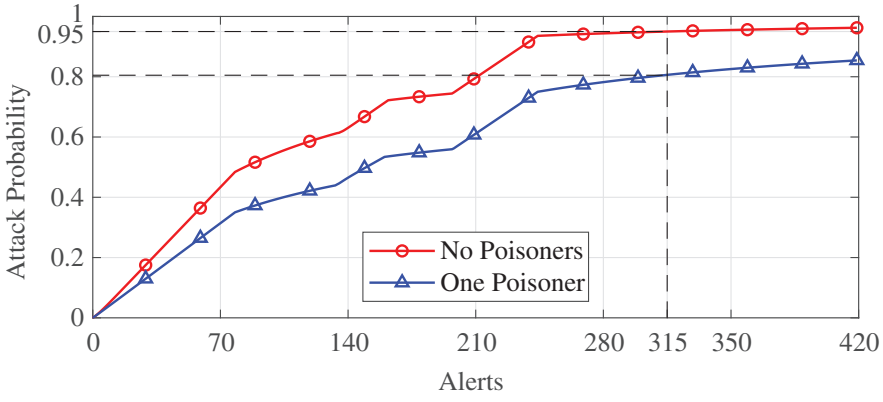

Figure 10. The performance of the attack probability

the current state based on the current alerts. We have

$$\text{ASPP} = \frac{\sum_{t=0}^{\text{Alerts}} \max[P(x_t = s_i|O, \lambda)]}{\text{Alerts}}. \tag{30}$$

The third metric is the attack probability, defined in Equation (7). It is used to indicate the progress of the current DDoS attack based on the current alerts.

We compare our aggregated global model without poisoners and the global model with one poisoner, to observer the performance changes. The experiment results are shown in Figures 8–10.

In Figure 8, we can see that the prediction accuracy rate can keep close to 1 steadily and the average value reaches to 0.995. However, when there is a poisoner in participants, the PAR of global model decreases and only has 0.9 approximately. It shows that the sever aggregates the poisoned model, which has an impact on the global model.

In Figure 9, the initial values of ASPP in both cases are only 0.664. The reason is when there are few alerts, the observation sequence is too short to accurately calculate the current state probability. As the number of incoming alerts increases, the ASPP without poisoners increases rapidly and reaches to 0.965, while the ASPP with one poisoner is lower and finally only reaches to 0.855. Combining Figures 8 and 9, we can see our global model without poisoners can accurately predict the current state with a high probability.

From Figure 10, we can find the attack probability rises as the number of incoming alerts increases. The attack probability represents how far a DDoS attack is from reaching the final target. A point of view [2] is that there should be some reactions when the attack probability arrives at 0.95. In our experiment, the final attack probability of the global model without poisoners can finally reach up to 0.963. When the attack probability arrives at 0.95, the number of incoming alerts is 315, much less than total alerts 420. It means that the BS has enough time to make active defenses. However, the attack probability of global model with one poisoner increases slowly. When the number of alerts reaches to 315, the attack probability is only 0.805, which is far less than 0.95.

7 Conclusions

An FL-based robust HMM prediction method is proposed for predicting multi-step DDoS attacks and defending the poisoning attacks in ITS. Also, a poisoning attack based on reinforcement learning is designed to poison the global model, and meanwhile, a validation dataset-free defense strategy is given to avoid anomaly aggregation. In the reinforcement learning-based poisoning algorithm, the poisoner can learn the best strategy to poison the global model and can speed up the poisoning process according to the accumulated experiences. For the validation dataset-free defense strategy, we analyze the characteristics of the poisoning attacks and propose three evidences to select benign participants. In the future work, we will consider a more complex attacking scenario, where multiple adversaries can work cooperatively to launch cooperative reinforcement learning-based poisoning attack. This will be a more challenging work for us to continue to study.

Conflict of Interest

The authors declare that they have no conflict of interest.

Data Availability

We make data available on request through sending an email to the authors.

Authors' Contributions

Zhong Li wrote and constructed this paper. Xianke Wu mainly surveyed the related work and helped do some experiments. Changjun Jiang discussed the recent development and jointly wrote this paper.

Acknowledgements

We thank all editors and reviewers who helped us improve the paper.

Funding

This research was supported in part by the National Key Research and Development Project (2018YFB2100801), in part by the National Natural Science Foundation of China (61972080), in part by the Shanghai Rising-Star Program (19QA1400300), in part by the Open Research Project from the Key Laboratory of the Ministry of Education for Embedded System and Service Computing (ESSCKF2021-01).

References

[1] Chadza T, Kyriakopoulos KG and Lambotharan S. Analysis of hidden Markov model learning algorithms for the detection and prediction of multi-stage network attacks. Future Gener Comput Syst 2020; **108**: 636–49.

[2] Sendi AS, Dagenais M and Jabbarifar M et al. Real time intrusion prediction based on optimized alerts with hidden Markov model. J Netw 2012; **7**: 311–21.

[3] Kavousi F and Akbari B. Automatic learning of attack behavior patterns using Bayesian networks. In: Proc. IEEE IST, Tehran, Iran, November 2012, 999–1004.

[4] Wang S, Tang G and Kou G et al. An attack graph generation method based on heuristic searching strategy. In: Proc. IEEE ICCC, Paris, France, July 2016, 1180–5.
[5] Holgado P, Villagrá VA and Vázquez L. Real-time multistep attack prediction based on hidden Markov models. IEEE Trans Dependable Secure Comput 2020; **17**: 134–47.
[6] Lyu L, Yu H and Yang Q. Threats to federated learning. Berlin, Germany: Springer, 2020.
[7] Hayes J and Ohrimenko O. Contamination attacks and mitigation in multi-party machine learning. In: Proc. NIPS, Montreal, QC, Canada, December 2018, 6604–15.
[8] Zhang J, Chen J and Wu D et al. Poisoning attack in federated learning using generative adversarial nets. In: Proc. IEEE TrustCom/BigDataSE, Rotorua, New Zealand, August 2019, 374–80.
[9] Lim WYB, Luong NC and Hoang DT et al. Federated learning in mobile edge networks: a comprehensive survey. IEEE Commun Surv Tutorials 2020; **22**: 2031–63.
[10] Xie C, Koyejo S and Gupta I. Slsgd: Secure and efficient distributed on-device machine learning. In: Proc. ECML PKDD, Würzburg, Germany, September 2019, 213–28.
[11] Fung C, Yoon CJ and Beschastnikh I. The limitations of federated learning in sybil settings. In: Proc. RAID, San Sebastian, Spain, October 2020, 301–16.
[12] Fang M, Cao X and Jia J et al. Local model poisoning attacks to byzantine-robust federated learning. In: Proc. USENIX Security, Boston, MA, USA, August 2020, 1605–22.
[13] Zhang J, Chen B and Cheng X et al. Poisongan: generative poisoning attacks against federated learning in edge computing systems. IEEE Internet Things J 2020; **8**: 3310–22.
[14] Tan J, Liang YC and Luong NC et al. Toward smart security enhancement of federated learning networks. IEEE Netw 2020; **35**: 340–7.
[15] Yin D, Chen Y and Kannan R et al. Byzantine-robust distributed learning: towards optimal statistical rates. In: Proc. PMLR ICML, Stockholm, Sweden, July 2018, 5650–9.
[16] Blanchard P, El Mhamdi EM and Guerraoui R et al. Machine learning with adversaries: byzantine tolerant gradient descent. In: Proc. NeurIPS, Long Beach, CA, USA, December 2017, 118–28.
[17] Chen Y, Su L and Xu J. Distributed statistical machine learning in adversarial settings: byzantine gradient descent. In: Proc. ACM SIGMETRICS, Irvine, California, June 2018, 1–28.
[18] Kang J, Xiong Z and Niyato D et al. Reliable federated learning for mobile networks. IEEE Wireless Commun 2020; **27**: 72–80.
[19] Taheri R, Shojafar M and Alazab M et al. Fed-IIoT: a robust federated malware detection architecture in industrial IoT. IEEE Trans Ind Inform 2020; **17**: 8442–52.
[20] Doshi R, Apthorpe N and Feamster N. Machine learning DDoS detection for consumer Internet of things devices. In: Proc. IEEE SPW, Gothenburg, Sweden, May 2018, 29–35.
[21] Agrawal N and Tapaswi S. Defense mechanisms against ddos attacks in a cloud computing environment: state-of-the-art and research challenges. IEEE Commun Surv Tutorials 2019; **21**: 3769–95.
[22] Van Hasselt H, Guez A and Silver D. Deep reinforcement learning with double q-learning. In: Proc. AAAI, Phoenix, Arizona, USA, February 2016, 2094–100.
[23] Sutton RS and Barto AG. Reinforcement Learning: An Introduction. Cambridge, MA, USA: MIT Press, 2018.
[24] Bogler PL. Shafer-Dempster reasoning with applications to multisensor target identification systems. IEEE Trans Syst Man Cybern 1987; **17**: 968–77.
[25] Xu R and Wunsch D. Survey of clustering algorithms. IEEE Trans Neural Netw 2005; **16**: 645–78.
[26] Zhang G, Zhang C and Zhang H. Improved k-means algorithm based on density canopy. Knowl-Based Syst 2018; **145**, 289–97.
[27] Xiang Z , Miller DJ , Kesidis G. A benchmark study of backdoor data poisoning defenses for deep neural network classifiers and a novel defense. In: Proc. MLSP, Pittsburgh, USA, October 2019, 1–6.
[28] Chen B, Carvalho W and Baracaldo N et al. Detecting backdoor attacks on deep neural networks by activation clustering. In: Proc. AAAI, Honolulu, Hawaii, January 2019, 1–8.
[29] Li D, Wong WE and Wang W et al. Detection and mitigation of label-flipping attacks in federated learning systems with KPCA and K-means. In: Proc. DSA, Yinchuan, China, August 2021, 551–9.

Zhong Li received the Ph.D. degree from Tongji University, Shanghai, China, in 2015. She is an Associate Professor with Donghua University, Shanghai. Her current research interests include wireless networking, network security, and intelligent transportation systems.

XianKe Wu is currently pursuing the M.S. degree from the College of Information Science and Technology in Donghua University, Shanghai, China. His research interests include security and privacy issues in vehicular networks.

Changjun Jiang received the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 1995. He was a Post-Doctoral Researcher with the Institute of Computing Technology, Chinese Academy of Sciences, in 1997. He is currently a Professor with the Department of Computer Science and Engineering, Tongji University, Shanghai. His current areas of research are concurrent theory, Petri net, and formal verification of software, concurrency processing, and intelligent transportation systems. He is a Council Member of the China Automation Federation and the Artificial Intelligence Federation, the Vice-Director of the Professional Committee of Petri Net of the China Computer Federation, the Vice-Director of the Professional Committee of Management Systems of the China Automation Federation, and an Information Area Specialist of the Shanghai Municipal Government.