Received April 10, 2021, accepted April 23, 2021, date of publication April 27, 2021, date of current version May 5, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3076118

Machine Learning Empowered Trust Evaluation Method for IoT Devices

WEI MA[1,2,3], XING WANG[4], MINGSHENG HU[1], AND QINGLEI ZHOU[2]
[1]School of Information Science and Technology, Zhengzhou Normal University, Zhengzhou 450044, China
[2]School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China
[3]School of Information Engineering, North China University of Water Resources and Electric Power, Zhengzhou 450046, China
[4]College of Electrical Engineering, Yuquan Campus, Zhejiang University, Hangzhou 310027, China

Corresponding authors: Wei Ma (wei.ma1222@gmail.com) and Qinglei Zhou (ieqlzhou@zzu.edu.cn)

This work was supported in part by the Henan Programs for Science and Technology Development under Grant 212102210100, in part by the Henan Key Research Projects of Universities under Grant 20A413008, Grant 21B520022, and Grant 20B520040; and in part by the Consulting Research Project of Chinese Academy of Engineering under Grant 2020YNZH7.

ABSTRACT With the rapid development of the Internet of Things (IoT), malicious or affected IoT devices have imposed enormous threats on the IoT environment. To address this issue, trust has been introduced as an important security tool for discovering or identifying abnormal devices in IoT networks. However, evaluating trust for IoT devices is challenging because trust is a degree of belief with regard to various types of trust properties and is difficult to measure. Thus, a machine learning empowered trust evaluation method is proposed in this paper. With this method, the trust properties of network QoS (Quality of Service) are aggregated with a deep learning algorithm to build a behavioral model for a given IoT device, and the time-dependent features of network behaviors are fully considered. Trust is also quantified as continuous numerical values by calculating the similarity between real network behaviors and network behaviors predicted by this behavioral model. Trust values can indicate the trust status of a device and are used for decision making. Finally, the proposed method is verified with experiments, and its effectiveness is described.

INDEX TERMS Internet of Things, trust evaluation, network behaviors, machine learning-based method.

I. INTRODUCTION

As an emerging type of network, the Internet of Things (IoT) has been under development for years. ''Things'' such as sensors, monitors, and mobile devices are connected via various network technologies. By interconnecting devices, the IoT can collect information and provide information services to the physical world. Many industries have benefitted from IoT technology, such as smart cities, smart grids, and the Internet of Vehicles, and a market with a value of more than a billion dollars has developed using the IoT. However, security issues hamper the further development of the IoT and its applications. Devices that lack security considerations or that have security vulnerabilities pose great risks to the entire IoT environment. In 2016, a DDoS attack was launched by IoT devices infected with Mirai [1], and many web services, such as Twitter, Netflix, and NY Times, were affected. As reported, nowadays more than 25% of infected devices in Botnet are IoT devices instead of traditional computers.

The associate editor coordinating the review of this manuscript and approving it for publication was Md. Arafatur Rahman.

Traditional security mechanisms such as cryptology and identity authentication [46] are adopted to address the security issues in IoT. However, constrained by the unclear network perimeter and weak computing capability of IoT devices, the traditional mechanisms are not applicable for all scenarios of IoT. Building trust is a useful method for security issues and can identify problematic, suspicious, or ''untrustworthy'' devices on a network. Trust mechanisms have been studied and applied in many fields, such as social network, e-commerce, and peer-to-peer network [36]. Trust mechanisms evaluate the trustworthiness of every entity in a network environment, and the trust level or quantified numerical trust value can be used for data fusion, decision making, and service management [2], [3]. Additionally, trust management in the IoT is an efficient security countermeasure. The concept of trust in the IoT is a degree of belief with regard to the behaviors of a specific entity and is based on experience or knowledge derived from interactions between IoT entities. In IoT environments, trust can cope with malicious nodes, misbehaving nodes or nodes affected by malicious software.

However, it is not easy to build trust for the IoT due to the complexity of the experiences, knowledge, and relationships between different devices, and it is not easy to calculate trust based on various types of trust properties. Thus, there are two key issues of trust evaluation in the IoT: trust metrics and trust computational methods [2]. Trust metrics provide a standard for evaluating trust, implying the information that the trustor (the subject of the trust relationship) can obtain from the trustee (the object of the trust relationship) and is used to evaluate the trustee [4]. Trust computational methods determine a trust level or numerical trust value of a trustee with specific trust metrics [5].

Trust metrics should be designed based on trust properties that can describe the status of the corresponding entity. As studied in the literature [6]–[8], trust metrics are based on social properties such as intimacy, honesty, and common interest derived from social relationships and on quality of service (QoS) properties such as energy consumption and latency derived from network status. There are also principles of selecting trust properties, which include choosing properties that are generic, easy and legitimate to obtain. Compared to social properties, QoS information is more generic and easier to obtain, which makes it more convenient for building trust in the IoT. Therefore, QoS properties are widely adopted in IoT trust studies. However, due to the diversity of IoT devices, only partial QoS properties that are exclusive to certain IoTs, such as wireless sensor networks (WSNs) or mobile ad hoc networks (MANETs), are discussed in the literature. With the development of the notion of edge computing, edge devices can comprehensively and generically collect a network's QoS information of IoT devices, which is a useful trust property for trust evaluation. Additionally, trust computational schemes are another important issue with respect to trust evaluation in the IoT and have also been widely investigated [5], [9]. Trust computational schemes concern an aggregation of trust properties and generate an assessment of the trustee. Certain schemes focus on discriminating untrustworthy devices from trusted ones, while other schemes focus on calculating a numerical trust value for each device. In contrast, the discrimination scheme is more arbitrary; however, the numerical value scheme is more flexible and can be applied in more cases. The most common numerical computational approaches for trust in recent studies are the weighted sum method and regression [9]–[11]; however, these methods only take advantage of limited trust properties instead of making full use of them. Certain important trust properties, such as time-dependent features, are not fully considered [36]. We thus believe that the trust evaluation for a device is not a static binary problem, and that it would be better to describe the trust status with a set of dynamic and continuous numerical values.

Because the proposed approach focusses on evaluating the trustworthiness of IoT devices based on network QoS properties with a numerical trust value, the contributions of this paper include the following:

- Network behaviors are generic and easy-to-obtain trust properties; thus, in this paper, trust metrics based on comprehensive network behaviors are adopted in trust evaluation.
- Different trust properties with different dimensions are integrated to compute numerical trust values.
- Time dependence is an important feature for network behaviors; thus, a time-series-based learning algorithm is adopted for computation;
- The effectiveness of this method is evaluated in a simulative environment.

The remainder of this paper is organized as follows. In Section II, a brief introduction to trust computation and management is given. In Section III, we describe the proposed trust evaluation method. Next, experiments and results are discussed in Section IV. Section V concludes this article.

II. RELATED WORK

The notion of trust derives from social science and has been investigated in other areas, such as economics and computer science. For computer science, research on trust always focuses on security issues. Trust and security are complementary to each other, and in most cases, trust is a concern about when and where security mechanisms should be deployed [3], [12]. Trust management in the IoT has been studied for years and from many perspectives. [13], [14] reviewed trust and reputation models in specific IoTs, such as WSNs and ad hoc networks. [15] reviewed the application of trust in WSNs and MANETs. In [16], trust metrics, attacks on trust, and performance in MANETs were discussed, and the author also summarized the research directions of trust management in MANETs. Trust metrics, trust aggregation and reputation were studied in [17], and trust computational schemes were reviewed in [18]. A comprehensive review of trust management in the IoT was given in [2], in which the roles played in trust management were identified. Based on trust computation schemes, trust models in the IoT were classified in [19] for trust-based services management. [20] summarized trust management in the IoT, and [3] provided thematic taxonomy for trust in the IoT, including trust metrics, trust properties, trust computation schemes, and trust applications.

Trust metrics and trust computation methods are two key factors of trust management in the IoT. Trust metrics concern "what" to evaluate, and trust computation methods concern "how" to evaluate. Both factors have been widely studied. Trust metrics are based on QoS properties or social properties [4]. The concept of QoS trust metrics is derived from QoS information of computer systems or networks. For example, [21] used energy consumption and the resent/delivery ratio in WSNs as trust metrics. The result of storing and transmitting data was chosen as a trust metric in [22]. [9] used the performance of IoT services to evaluate the trustworthiness of nodes. [23] considered QoS properties such as the forwarding ratio, resent ratio, and forward latency. With the notion of social network and social IoT, social properties were also adopted as trust metrics. The social properties of IoT devices are derived from the owners and users of the devices, such as honesty, colleague/friendship relations, intimacy, and common interests. Social trust properties have been studied extensively in recent years. Friendship, social connections, and common interests were used to rank the services provider in [6]. [25] utilized honesty, connections, intimacy, and selfishness to evaluate social trust level. Comprehensive aspects of service reputation and similarity were considered in [26]. The REK model (reputation-evidence-knowledge model) was proposed in [27], which considered both QoS properties and social properties, and classified them into three categories. Social properties are important and useful information for evaluating trust in the IoT; however, they are only applicable to the social IoT, while QoS properties are more generic and easier to access and collect.

Trust computational methods are another important factor of trust management in the IoT. Typically, there are three categories of major methods: weighted sum, inference approaches, and regression analysis [4]. Weighed sum is a common computational method, and the primary idea of weighted sum is that the more important the trust property is, the higher weight it will be assigned. [23] adopted the weighted sum method to compute direct/indirect trust. [28] used the weighted sum method to obtain an indirect trust value with recommendations and feedback. Certain inference approaches such as fuzzy logic and Bayesian inference are used to compute trust levels. [21] used fuzzy logic to obtain a numerical trust value in (0,1). [23], [28], [29] took advantage of Bayesian inference to express trust values as expectations of a beta distribution. In recent years, regression analysis or machine-learning-based methods have been adopted to measure trust with comprehensive trust properties based on statistical methods, particularly the behaviors of IoT nodes. LogitTrust [11] proposed a logic-regression-based method to predict service performance with the properties of energy consumption, service prices, and data transmission ratio. K-means and support vector machines were adopted in [30] to distinguish untrustworthy IoT nodes from trusted ones. Recently, reinforcement learning and multiclass classification technique-based trust computational methods were proposed in [31]–[33]. Regression or learning-based trust computational methods have shown great potential at determining a more accurate numerical trust value with more comprehensive trust properties. Also, in addition to the application of trust evaluation for the IoT, machine learning techniques have become a powerful tool with research on security or privacy issues in the IoT environment [42]. For example, federated learning was used on edge devices to reduce the computational overhead of IoT devices in [39], and privacy was also considered when edge devices collect data from IoT devices [40]. In [41], reinforcement learning was adopted to determine the offloading strategy for healthcare IoT devices to protect location privacy and usage pattern privacy. In [43], a reinforcement learning and blockchain based method was proposed to address selfish edge attacks in edge computing networks. In [44], data sharing problem in industrial Internet of Things was formulated as a machine learning problem and federated learning was adopted to ensure the data privacy was maintained when sharing the data.

III. LEARNING-BASED TRUST COMPUTATIONAL METHOD

A. SYSTEM MODEL

With the concepts of edge computing and fog computing [37], a layered architecture is formed for the IoT. As shown in Figure 1, the architecture is composed of 3 layers: the device layer, the edge and fog layer, and the cloud layer. Various sensing devices are deployed in the device layer and are finally connected to the cloud layer with edge and fog layer which consists of multiple base stations and access points.

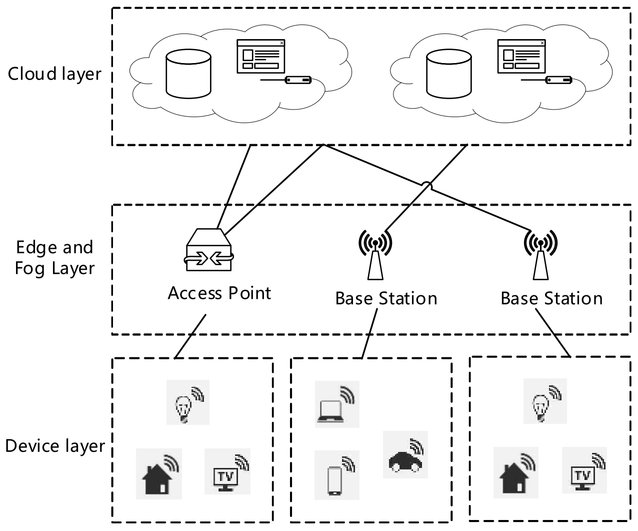


FIGURE 1. 3-layer architecture of IoT.

In this architecture, sensing data collected from the device layer will be transmitted to the edge and fog layer, and eventually stored and processed in the cloud layer. However, with more computational resources, energy, and capability, devices in the edge and fog layers can preliminarily process sensing data when it is transmitted to the edge and fog layers. Base stations and access points can obtain the whole information of network connections of the IoT devices connected to them. Thus, in the proposed method, the edge and fog layer plays an important role in collecting the network behaviors of IoT devices and deploying learning algorithms.

B. ATTACK MODEL

Because the proposed method aims to evaluate the trustworthiness of IoT nodes based on the metrics of network behaviors, the attack model is defined with the following capabilities and incapabilities of the attacker.

1) CAPABILITIES

The attacker or malicious node can attempt to connect to the C&C server (command & control) of the botnet using hard-coded network addresses and private protocols via the access point or base station.

The attacker or malicious node can launch network attacks by sending attack packets to the access point or base station.

2) INCAPABILITIES

The attacker or malicious node cannot compromise edge devices in other word, the access point or base station is trusted in the attack model.

The attacker or malicious node cannot bypass the access point or base station, namely all network communications of the IoT node must pass through edge devices.

C. FRAMEWORK OF THE PROPOSED METHOD

Because IoT devices are constantly connected to a network, network behaviors are critical to identify devices. The essential aspect of the method proposed in this paper is to predict the future behaviors of a specific device based on historical experience. A learning algorithm is used in this paper to capture not only the features of network behaviors but also the time-dependent features implied in the network behaviors. The similarity between predicted behaviors and real follow-up behaviors will be computed and used to obtain a numerical trust value. There are two phases of the proposed method: offline training and online computation. The framework of the proposed method is shown in Figure 2.

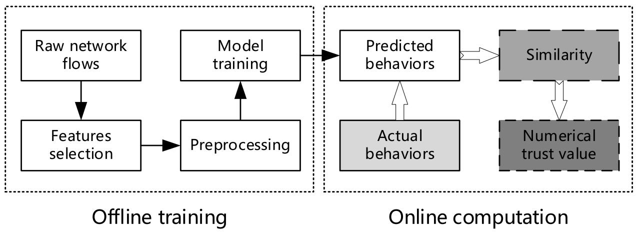


FIGURE 2. Framework of learning-based trust computational method.

There are several independent steps in each phase. In phase 1, the steps include gathering raw network flows, feature selection, preprocessing, and model training. In phase 2, there are steps of predicting behaviors, computing similarity, and calculating numerical trust value. Important symbols are summarized in Table 1.

D. GATHERING RAW NETWORK FLOWS AND SELECTING FEATURES

Utilizing the technique of sniffing, we can collect raw network flows of the targeted IoT device from edge devices. Considering the level of network activity of a device, a time threshold is defined that describes for how long a single network flow would be captured. The symbol TH_{flow} is used to represent the time threshold.

Network behaviors are profiled by statistical information derived from network flows. Thus, with the proposed method, we must also select proper features that can describe the behavior patterns of devices. Considering the significance of different features and based on [34], [35], we have chosen 14 features for the proposed method, as shown in Table 2.

Thus, with the proposed scheme, one network flow stands for network information captured per TH_{flow} seconds. Using $x_i = [f_i^i, f_i^2, \ldots, f_i^{14}]^T$ to denote a single network flow, in which i indicates that it is the i-th network flow, the collected network flows of the device are represented as $X = \{x_1, x_2, \ldots, x_n\}$, where n is the total number of network flows.

TABLE 1. List of notations.

Category	Name	Description
Symbols in data prepossessing phase	TH_{flow}	Time threshold for a network flow
	x_i	i-th network flow
	f_i^j	j-th feature in x_i
	$\bar{x}, \bar{f}$	Normalized x and f
	Wn	Size of the sliding window
	s	Step size of the sliding window
	n	Number of network flows
Symbols in trust value computation phase	P, A	cluster of Predicted data points and Actual data points
	Cen_P, Cen_A	Central point of P and A
	$AvgDistance$	Average distance of a cluster
	D_{PA}	Euclidean distance between Cen_P and Cen_A
	$Diff$	Difference between two average distances
	$T(x)$	Trust value calculating function
	ΔT	Variance of trust value
Symbols in model training phase	F_t, I_t, O_t	Forget gate, input gate and output gate of LSTM
	W, b	Weight and bias value
	h_t	Output of the t-th Hidden layer
	C_t	Cell state of the t-th Hidden layer

TABLE 2. Selected features for trust properties.

Name	Description
Packet-sent	Number of packets sent by the device.
Packet-received	Number of packets received by the device.
Max-size-sent	Maximum size of packet sent by the device.
Min-size-sent	Minimum size of packet sent by the device.
Avg-size-sent	Average size of packet sent by the device.
Max-size-received	Maximum size of packet received by the device.
Min-size- received	Minimum size of packet received by the device.
Avg-size- received	Average size of packet received by the device.
Mean-time-sent	Mean time between two packets sent.
Mean-time-received	Mean time between two packets received.
Avg-byte-sent	Average bytes per packet sent.
Avg-byte-received	Average bytes per packet received.
Reconnection	Number of reconnections.
Ratio-sent-received	Ratio between packets sent and received.

E. DATA PREPROCESSING

During preprocessing, two tasks are performed: normalization and sequence generation.

Normalization: To eliminate the influence of different data scales and dimensions in raw network flows, normalization is used in the proposed method. Normalization adjusts numerical values measured on different scales to a common scale to reduce computational overhead and improve calculation accuracy. Considering the characteristics of the features selected, we use min-max normalization with the proposed method. For collected network flows X, normalization is performed following formula (1):

$$\bar{f}_i = \frac{f_i - \min(f_i)}{\max(f_i) - \min(f_i)} \tag{1}$$

With min-max normalization, the values of all features will be converted into the range of [0, 1]. The normalized network flows will be denoted as $\bar{X} = \{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n\}$, and the learning process will benefit from normalization.

Sequence Generation: With $\bar{X}$, we can build training sets for the learning algorithm. Note that network flows are performed by devices in a timed sequence. The data at a given sampling time point are likely related to the data at the previous and next sampling time points. Therefore, the time dependency of network flows is considered a hidden feature to profile device behaviors. The purpose of the proposed method is to train a model to predict future behaviors of devices based on collected historical behaviors; thus, a typical regression problem and supervised learning are required. It is assumed that the collected network flows $\bar{X} = \{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n\}$; we also use a sliding window to generate sequences for training sets, as shown in Figure 3. First, we set up a window size wn; in Figure 3, $wn = 3$. The window will slide from the beginning of $\bar{X}$ with a step size of s, as shown in Figure 3 $s = 2$. The network flows falling in the window, such as $\bar{x}_1$, $\bar{x}_2$ and $\bar{x}_3$, become a sequence of the training set, and the next flow right after this sequence $\bar{x}_4$ is considered the "label" of this sequence.

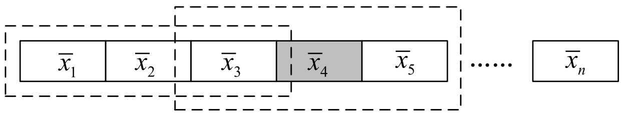


FIGURE 3. Sliding window for generating sequences.

With the sliding window mechanism, a training set with $\lfloor (n - wn)/s \rfloor + 1$ labeled records will be generated. This training set will be used to train the model.

F. MODEL TRAINING

To cope with sequences that exhibit time dependencies, a long short-term memory (LSTM) neural network structure is adopted. LSTM can efficiently predict the objects of a series of inputs, and it is widely used in security scenarios such as threats hunting in IoT and malware detection [45]. The input layer of LSTM is associated with a time series, and LSTM can evaluate the impact of different time steps and influence the final output, which is the next network flow after the input sequence in this paper. The structure of LSTM used in the proposed method is shown in Figure 4.

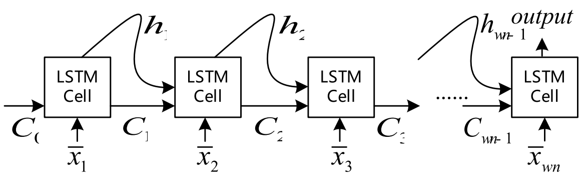


FIGURE 4. LSTM structure.

The input of the LSTM is a $14 \times wn$ matrix, and the output is a 14-dimension vector. The output indicates the next network flow after flows in the input. For every LSTM cell, the current input $\bar{x}_t (t \in [1, wn])$ is calculated with three "gates" (a forget gate, an input gate, and an output gate) and a cell state. This process is described as follows.

First, the forget gate takes $\bar{x}_t$ and h_{t-1}, the output of the hidden layer, as the input to calculate as follows:

$$F_t = \sigma\left(W_f \cdot [h_{t-1}, \bar{x}_t] + b_f\right) \tag{2}$$

where $\sigma(\cdot)$ is the sigmoid function. The first step obtains a value to determine how much information within cell state C_{t-1} is reserved.

Next, the input gate decides what information to update with $\bar{x}_t$ and h_{t-1}, and a new cell state information $\tilde{C}_t$ will be generated for updating:

$$I_t = \sigma\left(W_i \cdot [h_{t-1}, \bar{x}_t] + b_i\right) \tag{3}$$

$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, \bar{x}_t] + b_C\right) \tag{4}$$

With the forget and input gates, the LSTM cell updates the cell state C_t with C_{t-1} and $\tilde{C}_t$ as follows:

$$C_t = F_t * C_{t-1} + I_t * \tilde{C}_t \tag{5}$$

The output gate will then decide what to output based on x_t and h_{t-1}:

$$O_t = \sigma\left(W_o\,[h_{t-1} + b_o]\right) \tag{6}$$

As a hidden layer, the output will be:

$$h_t = O_t * \tanh(C_t)$$

In this process, W and b are learnable parameters, which are determined during training. The size of the sliding window wn and the step size s are hyperparameters for the model.

G. PREDICTION, COMPUTATION OF SIMILARITY AND TRUST VALUE

With the trained model, we can predict the network behaviors that the device is likely to perform in the future. As shown in algorithm 1, the next k-step network behaviors will be predicted for future analysis.

We can inspect the status of the device by checking the similarity between predicted and collected real behaviors.

Algorithm 1 PredictNetworkBehaviors

```
Input: Current network behaviors Queue Current, steps to
    predict k
Output: Predicted network behaviors Predict
 1: iter = 0
 2: while (TRUE) do
 3:     if iter <= k then
 4:         next = LSTM(Current)
 5:         Current dequeue
 6:         Current enqueue next
 7:     else
 8:         Predict = Current
 9:         return Predict
10:     end if
11: end while
```

Note that the predicted and real behaviors are represented as two clusters of 14-dimension data points, and a scheme is designed to compute the similarity.

It is assumed that the predicted cluster of data points $P = \{p_1, p_2, \ldots, p_k\}$ and the real cluster of data points $A = \{a_1, a_2, \ldots, a_k\}$; every point in them has 14 dimensions. First, we must determine the central points of A and P with the mean value of the points in the clusters:

$$\begin{cases} Cen_P = \frac{\sum_{i=1}^{k} p_i}{k} \\ Cen_A = \frac{\sum_{i=1}^{k} a_i}{k} \end{cases} \tag{7}$$

Next, the average distances between every two points in every cluster are calculated as algorithm 2:

Algorithm 2 Calculate Average Distance

```
Input: Cluster of data points C
Output: Average distance of points in C AvgDistance
1: d = 0
2: for every Pair of Points P(p1,p2) in C do
3:     d = d + Euclidean Distance(p1,p2)
4: end for
5: AvgDistance = 2*d/Length(C)*(Length(C)-1)
6: return AvgDistance
```

Second, with the central point and the average distance in the cluster, we can determine the scale of the clusters. The similarity between two clusters can also be determined with the central point and average distance. We define the difference between the average distance of A, the average distance of P and the distance between central points of the two clusters with (8):

$$\begin{cases} Diff = AvgDistance_A - AvgDistance_P \\ D_{PA} = EuclideanDisance(Cen_P, Cen_A) \end{cases} \tag{8}$$

where the further the distance between two central points of two clusters, the less similar they are. With a short distance between central points, if the average distances vary greatly, the two clusters are also not similar. Using this metric, the trust value computation scheme is calculated with a recurrent function as follows:

$$T(x) = T(x-1) + \Delta T \tag{9}$$

where x is the number of sampling points, $T(0) = T_0$ is the initial trust value assigned to the device, and ΔT is the variance of the trust value:

$$\Delta T = \begin{cases} 1, & Diff < 0 \wedge D_{PA} < AD_A \\ -\tanh(|Diff| + D_{PA}) * t_0, & otherwise \end{cases} \tag{10}$$

where AD_A indicates the average distance of the cluster of real behaviors, and D_{PA} is the distance between Cen_P and Cen_A. In this scheme, we conduct a penalty mechanism and a rewarding mechanism simultaneously. The penalty mechanism indicates that when there is a significant deviation between predicted behaviors and real behaviors, the trust value would decay based on the similarity between two clusters of behaviors. A hyperbolic tangent function is used to determine the decayed trust value, as shown in (9). The rewarding mechanism indicates that if the predicted behaviors are nearly identical to the real behaviors, the trust value would increase marginally as a reward for the device. Because a prediction occurs every k steps of behaviors, the trust value is updated every $k \times TH_{flow}$ seconds. Thus, a series of dynamic trust values will be generated to depict the status of the evaluated device.

IV. EVALUATION AND DISCUSSION

A. DATE COLLECTION

Experiments were conducted to verify the proposed method. First, network flows were collected in an experimental environment, as shown in Figure 5.

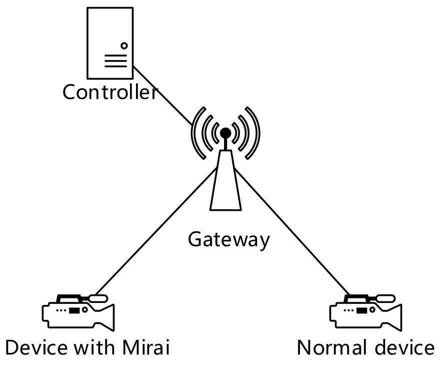


FIGURE 5. Experimental environment.

In the experimental environment, two IoT devices (two smart home cameras in this case) were connected to a gateway. One of the devices is affected with Mirai, and both devices send data packets to the controller. Network flows were captured via the sniffer deployed on the controller. To learn the behavior patterns of the devices in the normal state, we collected the network flows of both devices without activating Mirai. The collected data were used for offline training. In the proposed experiment, TH_{flow} was set to 180 seconds, and 30,000 network flows were collected for each device, which were denoted as $Data^{A}_{benign}$ and $Data^{N}_{benign}$.

After collecting training data, we also collected validation data. We manually set the affected device into an abnormal state by activating Mirai right after capturing the training data capture, and the next 100 network flows were captured as validation data. For contrast, the same 100 network flows of the normal device were also collected. The validation data were denoted as V^{A}_{Bot} and V^{N}_{benign}. Figure 6 shows the composition of the training data and validation data.

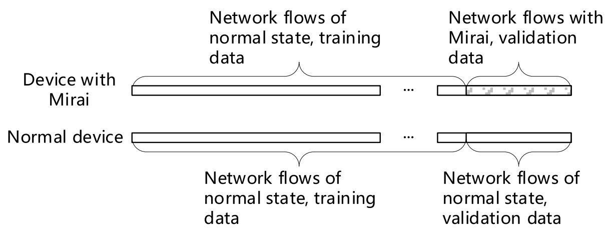

Network flows of normal state, training data
Network flows with Mirai, validation data
Device with Mirai
Normal device
Network flows of normal state, training data
Network flows of normal state, validation data

FIGURE 6. Composition of collected data.

B. EXPERIMENTS AND EVALUATION

1) EXPERIMENT SETUP

Except for TH_{flow}, other hyperparameters were also determined before training the model, as shown in Table 3.

TABLE 3. Setup of hyperparameters.

Hyperparameter	Description	Value
TH_{flow}	Time-threshold for capturing network flows	180
n	Number of network flows	30,000
wn	Size of the sliding window	99
s	Step size of the sliding window	1
batch_size	Number of training examples utilized in one iteration	500
epoch_size	Number of training data passes through the neural network.	5
learning_rate	Parameter that determines the step size at each iteration.	0.001
Activation function	Function that determines the output of a neuron.	ReLU
Loss Function	Function that measures the trained model.	Mean Square Error (MSE)

With the chosen hyperparameters, a training set with 29,902 records was generated. Experiments were conducted on a desktop server with an Intel i9-5820k CPU and 32 GB of RAM. Keras with TensorFlow was used as the backend along with scikit-learn in Python. At the beginning of the experiment, we compared several different machine learning models to achieve better performance, such as logistic regression (LR) in [11], support vector machine (SVM) in [30], and LSTM in Section III. And not only the traditional LSTM was adopted, but also the variants of LSTM, including stacked LSTM and bidirectional LSTM (Bi-LSTM), wherein stacked LSTM is a neural network with multiple hidden LSTM layers and has greater model complexity, which can create a more complex feature representation. Bi-LSTM considers input data twice to train itself from two directions, which can improve learning long-term dependencies and thus improve model accuracy. In the proposed experiment, we used a 3-layer stacked LSTM.

2) EXPERIMENT RESULTS AND COMPARISON ANALYSIS: TRAINING PERFORMANCE

To analyze the performance in the training phase, the models were trained independently for comparison. We trained the models independently with 20%, 50%, 80%, and 100% of the training set. The results are illustrated in Figure 7.

We compared the accuracy and training time of all models in the experiments. As shown in Figure 7(a), the performance of linear models was not as good as LSTM based models. With the increase of the data scale, the accuracy of the linear models hasn't changed significantly while the accuracy of the LSTM based models has increased as well. Note that the stated problem is not a binary classification problem, and the time-dependent feature should be captured. Therefore, LSTM-based models significantly outperformed linear models such as logistic regression and support vector machine.

In Figure 7(b), as the scale of the model parameters increased, the training time also increased. The LSTM-based models required more time to be trained than the linear models. However, because the training phase is offline, the extra time consumption would not affect the overall performance of the proposed method.

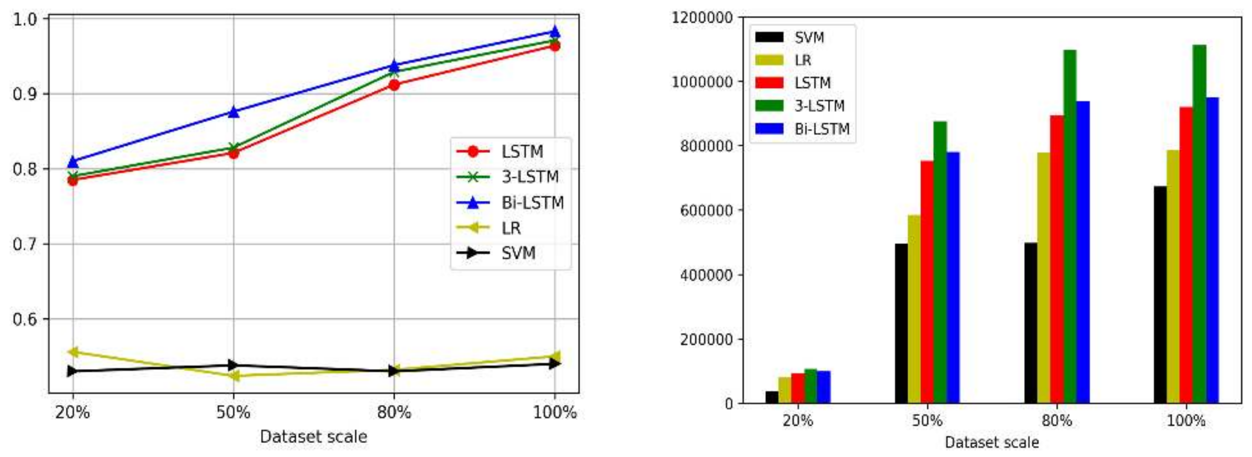
3) EXPERIMENT RESULTS AND COMPARISON ANALYSIS: REGRESSION PERFORMANCE

We also calculated the mean square error (MSE) and R-squared (R^2) values of the models with different data scales to evaluate the performance of regression using (11) and (12):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{11}$$

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y}_i)^2} \tag{12}$$

The symbols y_i, $\hat{y}_i$ and $\bar{y}_i$ represent the real value, predicted value and mean value, respectively. The results are summarized in Table 4. Those numerical results also indicate that the regression performance of LSTM-based models is better than that of linear models.



(a) Model accuracy **(b) Training time**

FIGURE 7. Training performance comparison of LSTM models and linear models.

TABLE 4. Regression performance of models.

Model	Dataset scale	Accuracy	MSE	R^2	Training Time
LR	20%	55.6%	10.9	-0.66	81931
	50%	52.4%	11.2	-0.73	585327
	80%	53.2%	11.2	-0.73	778090
	100%	55%	10.7	-0.67	787439
SVM	20%	53%	11.21	-0.72	37968
	50%	53.8%	10.9	-0.72	494955
	80%	53%	11.2	-0.72	497781
	100%	54%	10.9	-0.71	673303
LSTM	20%	78.5%	0.022	0.51	92113
	50%	82.1%	0.018	0.59	752168
	80%	91.2%	0.014	0.69	893741
	100%	96.4%	0.008	0.81	920325
3-LSTM	20%	79%	0.019	0.52	106551
	50%	82.8%	0.018	0.59	875545
	80%	92.9%	0.008	0.82	1096818
	100%	97.1%	0.006	0.86	1112398
Bi-LSTM	20%	81%	0.018	0.56	99249
	50%	87.6%	0.016	0.62	780436
	80%	93.8%	0.008	0.84	937453
	100%	98.3%	0.005	0.88	949918

TABLE 5. Trust values in different sample point.

Sample point	LR		SVM		Our method	
	N	A	N	A	N	A
0th	20	20	20	20	20	20
5th	4.7	0.4	5.2	7.8	21.2	19.4
10th	0.3	0.1	0.3	3.2	21.1	17.2
15th	<0.1	<0.1	1.0	0.3	20.8	3.5
20th	<0.1	<0.1	0.1	0.1	19.5	0.8
25th	<0.1	<0.1	0.2	<0.1	20.7	0.1
30th	<0.1	<0.1	0.2	<0.1	18.6	<0.1

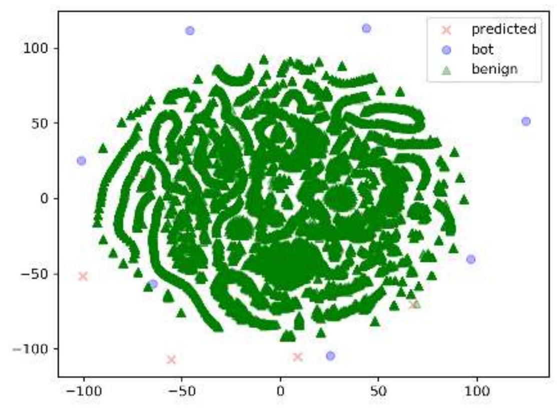
*N=Normal device, A=Affected device with Mirai.

4) EXPERIMENT RESULTS AND COMPARISON ANALYSIS: ADVANCED PERFORMANCE COMPARISON AMONG LSTM-BASED MODELS

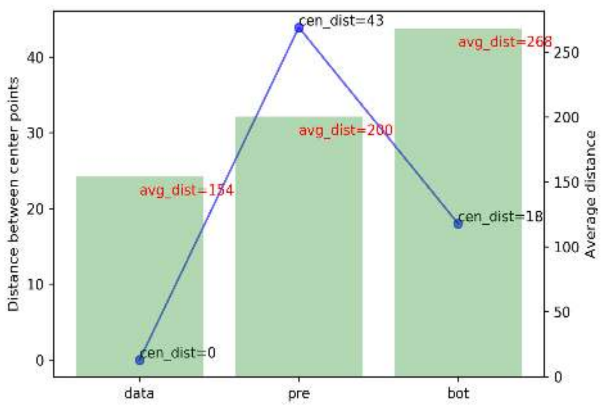
Because the accuracy of LSTM-based models is similar, we conducted experiments to compare the performance of LSTM, stacked LSTM, and Bi-LSTM. We also applied the method on V_{Bot}^{A} with Algorithm 1 and Algorithm 2 to compute $AvgDistance_a$, D_{PA}, and $AvgDistance_p$ by predicting the next 10 network flows. Note that data were made of 14-dimension points, and t-SNE was used to reduce the dimensions for visualization. The results are shown in Figure 8. Results showed that the Bi-LSTM model achieved the best performance. With the LSTM model, the benign cluster, the predicted cluster, and the bot cluster were not distinctly separated. With stacked LSTM, although the edges of the three clusters were distinguishable, the distance between the predicted cluster and the benign cluster was greater than the distance between the bot cluster and the benign cluster, which indicates that the prediction shifted from normal behaviors. Only Bi-LSTM showed that it was able to predict behaviors following the learned pattern. Thus, Bi-LSTM was used to compute the numerical trust value in this study.

5) EXPERIMENT RESULTS AND COMPARISON ANALYSIS: TRUST VALUE COMPUTATION

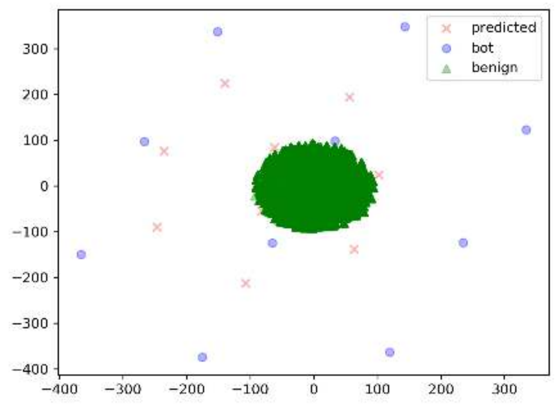
We conducted an experiment to calculate the trust value with the trained Bi-LSTM model. The experiment was designed as follows. First, a time slot was set, and in a single time slot, a certain number of network flows was captured. Network flows with the same number are predicted with the trained model. Next, the similarity between the captured network flows and predicted flows was calculated to determine the trust value of the device. This process was repeated to evaluate the device and generate a trust value in every time slot, which allowed the model to learn the trends of the trust value. Note that in every time slot, the data used for prediction were the network flows that were actually captured in the last time slot instead of the predicted flows. Conversely, the experiment was conducted with both devices (the normal device and the device with Mirai). In the experiment, there were 30 time slots with 10 network flows captured per time slot, and we activated Mirai on the affected device in the 10th time slot.



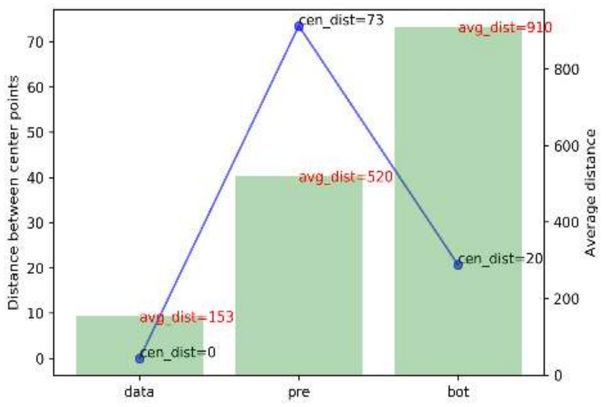
(a) Distribution of data points in the experiment with LSTM



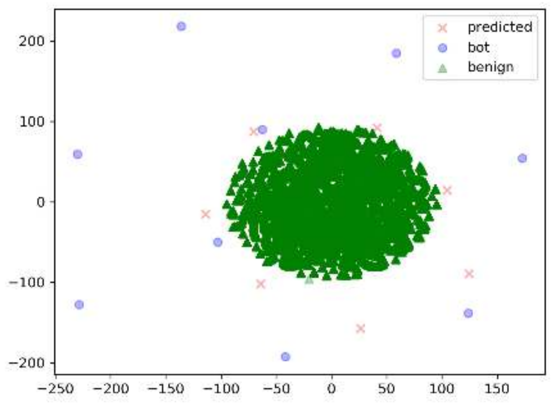
(b) Values of distances in the experiment with LSTM



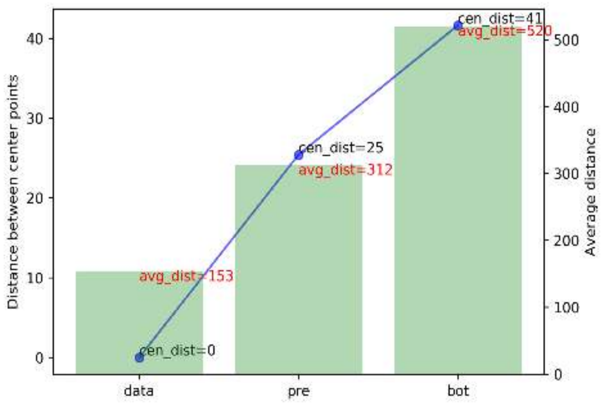
(c) Distribution of data points in the experiment with 3-LSTM



(d) Values of distances in the experiment with 3-LSTM



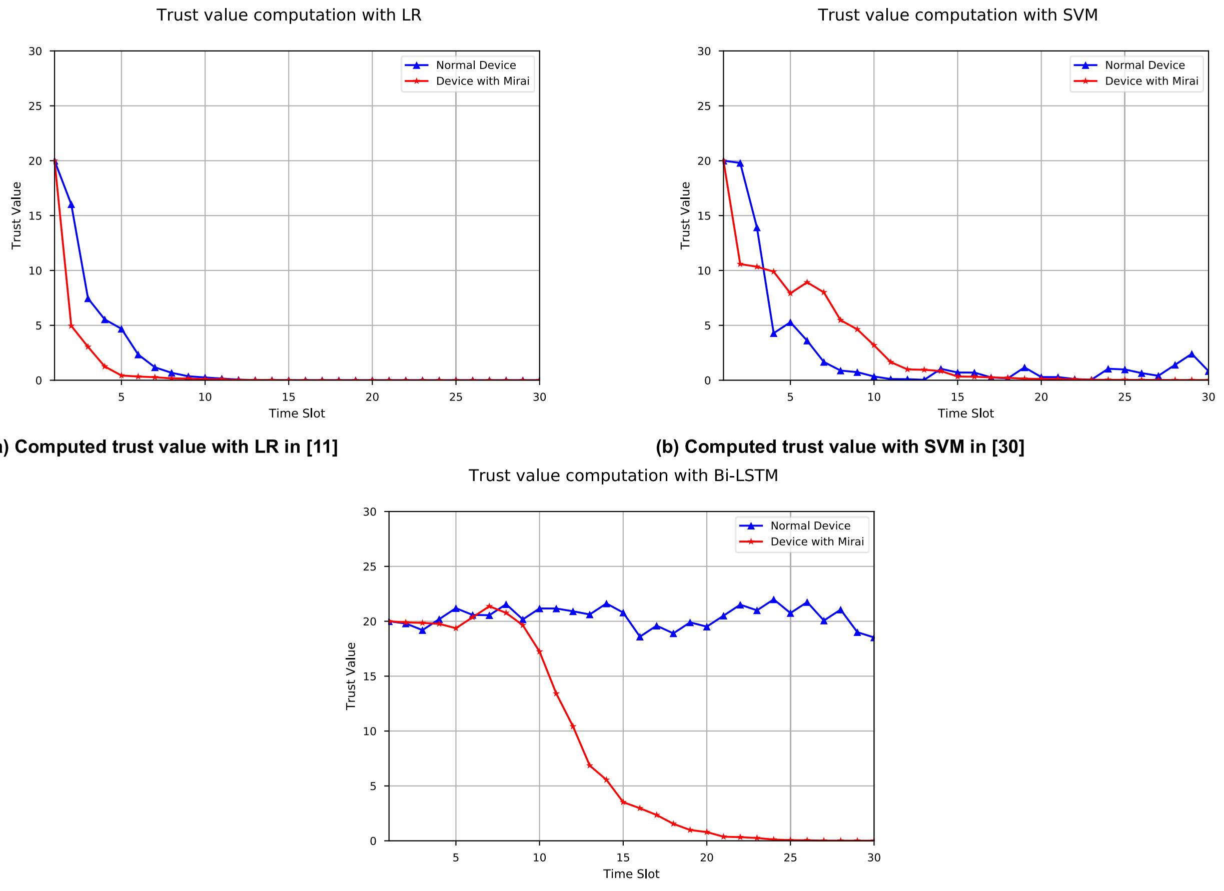
(e) Distribution of data points in the experiment with Bi-LSTM



(f) Values of distances in the experiment with Bi-LSTM

FIGURE 8. Method verification with 3 models, LSTM, stacked LSTM and Bi-LSTM, wherein cen_dist indicates the distance between the central point of the cluster of the training data and other clusters, the predicted data points and the bot data points. With LSTM, the perimeters of three data clusters were unclear and the computed distances also demonstrate that the model cannot tell the difference between predicted data and bot data. The perimeters were clearer with 3-LSTM and Bi-LSTM while only the values of distances calculated by Bi-LSTM were able to distinguish the bot data.

The initial trust value T_0 was set as 20. The trust values in different sample points are summarized in Table 5. And the global changes of the trust value in the 30 time slots were observed, as shown in Figure 9, in which Figure 9(c) is the



(a) Computed trust value with LR in [11]

(b) Computed trust value with SVM in [30]

(c) Computed trust value with our method

FIGURE 9. Computed trust values in the experiment.

TABLE 6. Qualitive comparison with associated methods.

	Trust metrics	Trust discrimination	Algorithm employed	Time dependence
Non-learning methods	QoS metrics or social metrics	Binary classification or numerical value	Non-learning algorithms, e.g., weighted sum	×
Method in [11]	QoS metrics	Binary classification	LR	×
Method in [30]	QoS metrics and social metrics	Binary classification	SVM	×
Our method	QoS metrics	Numerical value	LSTM-based	√

trust values calculated with our method and Figure 9(a) and Figure 9(b) are the results with LR and SVM respectively.

For the normal device, there was no significant change in the trust value across the 30 time slots. Marginal volatility was found in the trend of the trust value; however, the value remained near T_0. However, for the affected device, the trust value began to decay rapidly in the 10th time cycle and dropped to near zero in the 20th time cycle. According to (9) and (10) in Section III, the trust value never goes below zero; however, the downward trend and a near-zero value can indicate the status of the device. For comparison, we also conducted experiments with LR in [11] and SVM in [30] The trust value computed with LR and SVM decayed rapidly at the beginning of the experiments regardless of whether it was the normal device or the affected device due to the lower accuracy of the two models.

C. DISCUSSION

With the results of the experiments, it is clear that the proposed method can compute a series of numerical trust values for an IoT device. Considering certain related studies, we present the qualitative comparison of the proposed method in Table 5. Compared to the methods used in [11] and [30], the proposed method takes advantage of comprehensive QoS metrics, which are generic and easy to obtain. With edge computing architecture, collecting network QoS information is convenient, legitimate and able to describe the real states of devices to defend against trust attacks such as on-off attacks and self-promoting attacks [4]. By capturing time-dependent features, model performance in the proposed method yields a higher accuracy.

Instead of simply classifying one device as ‘‘trusted’’ or ‘‘untrusted’’, the primary output of the proposed method is intended to depict the trust status of the device continuously with continuously-changing numerical trust values. Thus, the feature of time dependence is important and is also the primary reason why the linear models did not perform well. The proposed method is thus beneficial for an IoT environment in which devices are in a constant, stable state.

Note that although the features we use with this method are primarily from the TCP/IP enabled transport layer protocol, we have a strong belief that the idea implied in the method is generic and flexible. With different trust metrics, trust evaluation can be performed from different perspectives. Thus, this method can be used in other IoT scenarios with different network architectures if there are sufficient collectable features. For example, it is assumed that an IoT environment wherein IoT devices connect to a gateway with the LoRa protocol, and the information in the MAC layer and PHY layer of the LoRa protocol stack can be collected by the gateway, such as the frame port, frame control flags, device address and frame payload. In this scenario, collected data are also time-dependent. Thus, with proper processing, these data can be utilized as features for LSTM training with the proposed method to learn the normal network behavioral patterns. The trust evaluation could be performed based on the trained model. In another scenario, it is assumed that a service-oriented IoT network with a feedback mechanism with which a service receiver can rate the received service based on the quality of service. Using sp_k to denote the k-th service provider and r_i^t to denote the rating score of the i-th IoT device that belongs to sp_k at time t, the matrix $sp_k = \begin{bmatrix} r_1^0 & \cdots & r_1^t \\ \vdots & \ddots & \vdots \\ r_i^0 & \cdots & r_i^t \end{bmatrix}$ is used to describe the service status of sp_k in a time period. This data format is not appropriate for LSTM. However, when training with appropriate models, such as convolutional LSTM neural network or graph-based models, the quality of service of sp_k can be predicted and evaluated. Although the algorithms used are different, the essence of the proposed method is the same.

In future work, we plan to focus on two research topics. First, note that the process of calculating trust values is based on a specific edge computing architecture of the IoT, as shown in Figure 1; thus, we plan to investigate how to improve the proposed method to be applicable in more flexible environments. Second, QoS metrics are practical with the proposed method, and social metrics are useful to describe the social relations between IoT entities. Third, although the data collected with the proposed method from IoT devices are not private, the privacy issue remains to be investigated [39]. Thus, in future work, we plan to consider taking advantage of more metrics, including QoS metrics and social metrics, and will consider the privacy of training data.

V. CONCLUSION

In this paper, we propose a trust evaluation method for IoT devices. The proposed method is empowered by the LSTM neural network to learn network behaviors patterns and time-dependent relations. We first extract network behaviors from the raw network flows of a specific device to build a training set. The training set is then used to train a behavioral model, which is used to predict the future behaviors of the device. Next, the similarity between the predicted and real behaviors is calculated, and numerical trust values are computed with similarity. Finally, with experiments, this method is demonstrated to produces promising results.

REFERENCES

[1] G. O. Young, ‘‘Synthetic structure of industrial plastics,’’ in *Plastics*, vol. 3, 2nd ed., J. Peters, Ed. New York, NY, USA: McGraw-Hill, 1964, pp. 15–64.

[2] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, ‘‘DDoS in the IoT: Mirai and other botnets,’’ *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[3] Z. Yan, P. Zhang, and A. V. Vasilakos, ‘‘A survey on trust management for Internet of Things,’’ *J. Netw. Comput. Appl.*, vol. 42, pp. 120–134, Jun. 2014.

[4] A. I. A. Ahmed, S. H. Ab Hamid, A. Gani, S. Khan, and M. K. Khan, ‘‘Trust and reputation for Internet of Things: Fundamentals, taxonomy, and open research challenges,’’ *J. Netw. Comput. Appl.*, vol. 145, Nov. 2019, Art. no. 102409.

[5] F. Bao and I.-R. Chen, ‘‘Dynamic trust management for Internet of Things applications,’’ in *Proc. Int. Workshop Self-Aware Internet Things*, 2012, pp. 1–6.

[6] A. Jøsang, R. Ismail, and C. Boyd, ‘‘A survey of trust and reputation systems for online service provision,’’ *Decis. Support Syst.*, vol. 43, no. 2, pp. 618–644, Mar. 2007.

[7] I.-R. Chen, J. Guo, and F. Bao, ‘‘Trust management for SOA-based IoT and its application to service composition,’’ *IEEE Trans. Services Comput.*, vol. 9, no. 3, pp. 482–495, May 2016.

[8] C. V. L. Mendoza and J. H. Kleinschmidt, ‘‘Mitigating on-off attacks in the Internet of Things using a distributed trust management scheme,’’ *Int. J. Distrib. Sensor Netw.*, vol. 11, no. 11, Nov. 2015, Art. no. 859731.

[9] S. Namal, H. Gamaarachchi, G. M. Lee, and T.-W. Um, ‘‘Autonomic trust management in cloud-based and highly dynamic IOT applications,’’ *J. Int. Bus. Res. Marketing*, vol. 1, no. 5, pp. 26–32, 2016.

[10] M. Nitti, R. Girau, and L. Atzori, ‘‘Trustworthiness management in the social Internet of Things,’’ *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 5, pp. 1253–1266, May 2014.

[11] P. Martinez-Julia and A. F. Skarmeta, ‘‘Beyond the separation of identifier and locator: Building an identity-based overlay network architecture for the future Internet,’’ *Comput. Netw.*, vol. 57, no. 10, pp. 2280–2300, Jul. 2013.

[12] Y. Wang, ‘‘LogitTrust: A logit regression-based trust model for mobile ad hoc networks,’’ in *Proc. 6th ASE Int. Conf. Privacy, Secur., Risk and Trust*, Boston, MA, USA, 2014, pp. 1–10.

[13] W. Harwood, ‘‘The logic of trust,’’ Ph.D. dissertation, Dept. Comput. Sci., Univ. York, Heslington, U.K., 2012.

[14] M. Momani and S. Challa, ‘‘Survey of trust models in different network domains,’’ 2010, *arXiv:1010.0168*. [Online]. Available: http://arxiv.org/abs/1010.0168

[15] M. A. Azer, S. M. El-Kassas, A. W. F. Hassan, and M. S. El-Soudani, ‘‘A survey on trust and reputation schemes in ad hoc networks,’’ in *Proc. 3rd Int. Conf. Availability, Rel. Secur.*, Mar. 2008, pp. 881–886.

[16] H. Yu, Z. Shen, C. Miao, C. Leung, and D. Niyato, ‘‘A survey of trust and reputation management systems in wireless communications,’’ *Proc. IEEE*, vol. 98, no. 10, pp. 1755–1772, Oct. 2010.

[17] J.-H. Cho, A. Swami, and I.-R. Chen, ‘‘A survey on trust management for mobile ad hoc networks,’’ *IEEE Commun. Surveys Tuts.*, vol. 13, no. 4, pp. 562–583, 1st Quart., 2011.

[18] K.-D. Chang, "A survey of trust management in WSNs, Internet of Things and future Internet," *KSII Trans. Internet Inf. Syst.*, vol. 6, no. 1, 2012.
[19] O. Khalid, S. U. Khan, S. A. Madani, K. Hayat, M. I. Khan, N. Min-Allah, J. Kolodziej, L. Wang, S. Zeadally, and D. Chen, "Comparative study of trust and reputation systems for wireless sensor networks," *Secur. Commun. Netw.*, vol. 6, no. 6, pp. 669–688, Jun. 2013.
[20] J. Guo, I.-R. Chen, and J. J. P. Tsai, "A survey of trust computation models for service management in Internet of Things systems," *Comput. Commun.*, vol. 97, pp. 1–14, Jan. 2017.
[21] I. Ud Din, M. Guizani, B.-S. Kim, S. Hassan, and M. Khurram Khan, "Trust management techniques for the Internet of Things: A survey," *IEEE Access*, vol. 7, pp. 29763–29787, 2019.
[22] D. Chen, G. Chang, D. Sun, J. Li, J. Jia, and X. Wang, "TRM-IoT: A trust management model based on fuzzy reputation for Internet of Things," *Comput. Sci. Inf. Syst.*, vol. 8, no. 4, pp. 1207–1228, 2011.
[23] L. Gu, J. Wang, and B. Sun, "Trust management mechanism for Internet of Things," *China Commun.*, vol. 11, no. 2, pp. 148–156, Feb. 2014.
[24] Y.-B. Liu, X.-H. Gong, and Y.-F. Feng, "Trust system based on node behavior detection in Internet of Things," *J. Commun.*, vol. 35, no. 5, pp. 8–15, 2014.
[25] Chen, Ingray, and Jia Guo, "Dynamic hierarchical trust management of mobile groups and its application to misbehaving node detection," *Proc. Adv. Inf. Netw. Appl.*, 2014, pp. 49–56.
[26] Y. Ben Saied, A. Olivereau, D. Zeghlache, and M. Laurent, "Trust management system design for the Internet of Things: A context-aware and multi-service approach," *Comput. Secur.*, vol. 39, pp. 351–365, Nov. 2013.
[27] U. U. K. Jayasinghe, "Trust evaluation in the IoT environment," Ph.D. dissertation, Dept. Comput. Sci., Liverpool John Moores Univ., Liverpool, U.K., 2018.
[28] W. Abdelghani, "Trust management in social Internet of Things: A survey," in *Proc. Conf. E-Bus., E-Services E-Soc.*, 2016, pp. 430–441.
[29] S. Ganeriwal, L. K. Balzano, and M. B. Srivastava, "Reputation-based framework for high integrity sensor networks," *ACM Trans. Sensor Netw.*, vol. 4, no. 3, pp. 1–37, May 2008.
[30] S. Ismail and A. Josang, "The beta reputation system," in *Proc. Bled Conf.*, 2002, pp. 2502–2511.
[31] U. Jayasinghe, G. M. Lee, T.-W. Um, and Q. Shi, "Machine learning based trust computational model for IoT services," *IEEE Trans. Sustain. Comput.*, vol. 4, no. 1, pp. 39–52, Jan. 2019.
[32] F. Boustanifar and Z. Movahedi, "A trust-based offloading for mobile M2M communications," in *Proc. Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People, Smart World Congr.*, Jul. 2016, pp. 1139–1143.
[33] W. Li, W. Meng, L.-F. Kwok, and H. H. S. Ip, "Enhancing collaborative intrusion detection networks against insider attacks using supervised intrusion sensitivity-based trust management model," *J. Netw. Comput. Appl.*, vol. 77, pp. 135–145, Jan. 2017.
[34] A. Bolster and A. Marshall, "Analytical metric weight generation for multi-domain trust in autonomous underwater MANETs," in *Proc. IEEE 3rd Underwater Commun. Netw. Conf.*, Aug. 2016, pp. 1–5.
[35] G. Kou, G.-M. Tang, S. Wang, H.-T. Song, and Y. Bian, "Using deep learning for detecting BotCloud," *J. Commun.*, vol. 37, no. 11, pp. 114–128, 2016.
[36] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant, "Botnet detection based on traffic behavior analysis and flow intervals," *Comput. Secur.*, vol. 39, pp. 2–16, Nov. 2013.
[37] J. Wang, X. Jing, Z. Yan, Y. Fu, W. Pedrycz, and L. T. Yang, "A survey on trust evaluation based on machine learning," *ACM Comput. Surveys*, vol. 53, no. 5, pp. 1–36, Oct. 2020.
[38] W. Yu, F. Liang, X. He, W. Grant Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.
[39] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Gener. Comput. Syst.*, vol. 115, pp. 619–640, Feb. 2021.
[40] L. Malina, "A privacy-enhancing framework for Internet of Things services," in *Proc. Int. Conf. Netw. Syst. Secur.* Cham, Switzerland: Springer, 2019, pp. 1–8.
[41] M. Min, X. Wan, L. Xiao, Y. Chen, M. Xia, D. Wu, and H. Dai, "Learning-based privacy-aware offloading for healthcare IoT with energy harvesting," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4307–4316, Jun. 2019.
[42] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for Internet of Things data analysis: A survey," *Digit. Commun. Netw.*, vol. 4, no. 3, pp. 161–175, Aug. 2018.
[43] L. Xiao, Y. Ding, D. Jiang, J. Huang, D. Wang, J. Li, and H. Vincent Poor, "A reinforcement learning and blockchain-based trust mechanism for edge networks," *IEEE Trans. Commun.*, vol. 68, no. 9, pp. 5460–5470, Sep. 2020.
[44] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4177–4186, Jun. 2020.
[45] A. Yazdinejad, H. HaddadPajouh, A. Dehghantanha, R. M. Parizi, G. Srivastava, and M.-Y. Chen, "Cryptocurrency malware hunting: A deep recurrent neural network approach," *Appl. Soft Comput.*, vol. 96, Nov. 2020, Art. no. 106630.
[46] C. Thirumalai, S. Mohan, and G. Srivastava, "An efficient public key secure scheme for cloud and IoT security," *Comput. Commun.*, vol. 150, pp. 634–643, Jan. 2020.

WEI MA received the B.S. degree from Henan Normal University, Xinxiang, China, in 2008, and the Ph.D. degree in information security from Beijing Jiaotong University, in 2016. He is currently a Postdoctoral Researcher with Zhengzhou University and Zhengzhou Normal University, and a Teacher with the North China University of Water Resources and Electric Power. He has published 20 articles in international journals and conference papers. His research interests include trusted computing, the IoT security, and cloud computing.

XING WANG was born in Taiyuan, Shanxi, China. He received the B.S. and Ph.D. degrees in information security from Beijing Jiaotong University, in 2009 and 2018, respectively. He is currently a Postdoctoral Researcher with Zhejiang University. He has published more than ten articles in IEEE journals and conferences. His research interests include machine learning and the IoT security.

MINGSHENG HU was born in Zhengzhou, Henan, China, in 1973. He received the B.S degree in computer software from Central China Normal University, in 1997, the M.S. degree in software engineering from Information Engineering University, in 2005, and the Ph.D. degree in control engineering from the Huazhong University of Science and Technology, in 2007. Since 1997, he has been working with Zhengzhou Normal University, where he is currently a Professor. He is an Awarder of the National Science Fund of China. He has published more than 20 papers and one monograph. His research interests include information security and complex networks.

QINGLEI ZHOU was born in Xinxiang, Henan, China, in 1962. He is currently a Professor and a Ph.D. Supervisor. He is also the Executive President of the School of Information Engineering, Zhengzhou University, China, and the Director of the China Computer Federation. He has published over 50 articles in the field of computer science. His major research interests include information security, automaton theory, and computational complexity theory. He was a recipient of the Science and Technology Expert of Outstanding Youth in Henan, China.