Computer Networks 190 (2021) 107952

Contents lists available at ScienceDirect

Computer Networks

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

An adaptive trust model based on recommendation filtering algorithm for the Internet of Things systems

Guozhu Chen [a], Fanping Zeng [a,b,*], Jian Zhang [c,d], Tingting Lu [a], Jingfei Shen [a], Wenjuan Shu [a]

[a] School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui, China
[b] Anhui Province Key Lab of Software in Computing and Communication, Hefei, Anhui, China
[c] State Key Laboratory of Computer Science, Institute of Software Chinese Academy of Sciences, Beijing, China
[d] University of Chinese Academy of Sciences, Beijing, China

ARTICLE INFO

Keywords:
Internet of Things
Trust model

ABSTRACT

The Internet of Things (IoT) is growing rapidly and brings great convenience to humans. But it also causes some security issues which may have negative impacts on humans. Trust management is an effective method to solve these problems by establishing trust relationships among interconnected IoT objects. In this paper, we propose an adaptive trust model based on recommendation filtering algorithm for the IoT systems. The utilization of sliding window and time decay function when calculating direct trust can greatly accelerate the convergence rate of trust evaluation.

We design a recommendation filtering algorithm to effectively filter out bad recommendations and minimize the impact of malicious objects. An adaptive weight is developed to better combine direct trust and recommendation trust into synthesis trust so as to adapt to the dynamically hostile environment. In the simulation experiments, we compare our adaptive trust model with three related models: TBSM, NRB and NTM. The experimental results indicate that our trust model converges fast and the mean absolute error is always less than 0.05 when the proportion of malicious nodes is from 10% to 70%. The comparative experiments further verify the effectiveness of our trust model in terms of accuracy, convergence rate and resistance to trust related attacks.

1. Introduction

The concept of Internet of Things (IoT) is to connect a large number of objects in the real physical world to the Internet based on standard communication protocols and unique addressing schemes [1]. These interconnected objects can be service providers offering services and sharing resources and information with each other. For the past few years, IoT has grown rapidly and a series of relevant services and applications including smart home, smart city and smart community [2] emerged. These services and applications bring great convenience to humans, but they also cause some security issues that may do harm to our lives. For example, a misbehaved object can perform various types of malicious attacks to destroy the integrity and availability of data and network resources. Trust management is an effective method to solve the above security issues by establishing trust relationships among objects and then excluding malicious objects. It allows multiple objects to share their opinions about the trust value of their companions [3].

Although trust management can effectively solve some of the security problems, there are still some challenges in building trust management systems. First, there are a large number of heterogeneous objects which have different functions and provide diverse services and applications. Consequently, an IoT trust model should be universal and capable of running on various types of objects. Second, most objects have limited capacities so that the existing trust models in P2P and social networks are no longer applicable. Third, many of the objects will be malicious for their own benefits and then carry out various malicious attacks in order to reduce the trust value of others or improve their own trustworthiness. As a result, IoT trust models should be resistant to those malicious attacks.

To meet the challenges discussed above, we propose an adaptive trust model to establish trust relationships among objects. Our trust model based on the recommendation filtering algorithm can effectively resist malicious attacks carried out by misbehaved objects and evaluate the trust value of target objects accurately. The major contributions of our paper are as follows:

- We propose a system architecture based on trust third parties (TTPs) which provides a secure and reliable trust computing

* Corresponding author at: School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui, China.
E-mail addresses: chengz18@mail.ustc.edu.cn (G. Chen), billzeng@ustc.edu.cn (F. Zeng).

https://doi.org/10.1016/j.comnet.2021.107952
Received 7 September 2020; Received in revised form 27 January 2021; Accepted 17 February 2021
Available online 22 February 2021

environment and hence saves storage and computing resources of IoT objects. Although in some previous work [4–7] and [8], the authors proposed hybrid architectures which are similar to ours, they did not specify what components are included in their proposed architecture and they did not explain how to apply their trust models to the architectures they proposed. Instead, we clarify the components included in our architecture and the functions of these components. Meanwhile, we explain the process of trust evaluation and the interaction process of these components in the architecture we proposed.
- Considering that the impact of past feedback will decrease over time, we introduce a sliding window to store feedback and use a time decay function to reduce the weight of the previous feedback. The differences from [5] and [6] are that we not only use the decay function to reduce the impact of previous feedback, we also propose a sliding window to save the feedback of the most recent period of time. The use of the sliding window can reflect the changes in the trust value of the IoT objects more quickly because of the fact that recent behaviors can better reflect the current trust status of IoT objects.
- We design a recommendation filtering algorithm based on k-means to filter out bad recommendations provided by malicious recommenders. Although a similar filtering algorithm was proposed in [5], we also introduce three important factors on the basis of our filtering algorithm. Even if the filtering algorithm cannot completely filter out the bad recommendations, the use of these three important factors can reduce the negative impact of the bad recommendations on the calculation of the recommendation trust as much as possible.
- We introduce an adaptive weight that can adjust automatically according to the dynamic environment to combine direct trust and recommendation trust. The experimental results indicate that our adaptive trust model enables fast and accurate trust evaluation and resists malicious attacks in the dynamically hostile environment. Compared with the fixed weight used in [9] and [10], our adaptive weight enables fast and accurate trust evaluation and resists malicious attacks in the dynamically hostile environment.

The remainder of this paper is organized as follows. In Section 2, we introduce the concept of trust and attack model in IoT. In Section 3, we survey the related work of IoT trust models. In Section 4, we propose the system architecture and the process of trust evaluation. In Section 5, we elaborate on our adaptive trust model and we give the experimental results and relevant analysis in Section 6. Finally, we summarize the paper and outline the future work in Section 7.

2. Background

In this section, we first introduce the concept of trust in IoT, the main participants in the trust model and the types of trust. Then, we list some trust related attacks that can break the trust management system. Finally, we introduce some common outlier detection methods which can be used to detect bad recommendations caused by those trust related attacks and filter out them from all the recommendations received by the trustor.

2.1. Trust in internet of things

In human society, trust usually indicates the degree of subjective belief between people. People are more likely to communicate with people they trust. Similarly, IoT objects are more willing to use services provided by trusted objects. Objects can evaluate the trust value of others through trust models before using their service.

There are three main participants in a trust model: trustor, trustee and recommender. A trustor is an object who wants to evaluate the trust value of others. Correspondingly, a trustee is an object who is evaluated by the trustor. If the trustor is satisfied with the service provided by the trustee, it will give the trustee a high trust rating. However, the trustor cannot interact with all trustees directly all the time. In this situation, the trustor needs recommendations from other objects that have interaction histories with trustees. Those objects who give recommendations to the trustor are called recommenders.

According to the above descriptions, we know that there are two types of trust relationships including direct trust and recommendation trust between a trustor and a trustee. The type of a given trust relationship depends on the way the trustor communicates with the trustee. If the trustor communicates with the trustee directly, this trust relationship is considered direct trust. Otherwise, we call the trust relationship recommendation trust. In our trust model, the trustor evaluates the trustee's trust value by synthesis trust that combines direct trust and recommendation trust by adaptive weight.

2.2. Attack model

A malicious object is dishonest and can perform malicious attacks such as providing bad service or recommending adverse trust information about trustees to the trustor. We call these attacks trust related attacks. The trust related attacks are summarized as follows:

- **On–off attacks**: A malicious object behaves well for a period of time and badly at other times. For example, a trustee can provide a trustor with good service that does not need many resources and prefers not to serve the trustor when the trustor needs too many resources.
- **Self promoting attacks**: A malicious object can promote its reputation by offering good recommendations about itself so that it can be selected as a service provider and then provides poor service. A service requester can hardly select good service providers under these attacks if the trust model does not ignore bad recommendations about the malicious object itself.
- **Bad mouthing attacks**: A malicious recommender can slander the reputation of a well-behaved trustee by providing the trustor with bad recommendations about that trustee. As a result, the trustee that is evaluated by the trustor with a low trust rating cannot be selected as a service provider.
- **Ballot stuffing attacks**: These attacks are similar to bad mouthing attacks. A badly-behaved trustee that cannot offer satisfying service will be highly rated by malicious recommenders that give opposite recommendations to the trustor. When multiple recommenders collaborate with each other to perform these attacks at the same time, they can boost the reputation of a bad trustee quickly.
- **Selective misbehavior attacks**: A malicious recommender provides the trustor with bad recommendations about some trustees and gives correct recommendations about others. In such a case, the trustor can hardly judge if the recommender is malicious because of its intermittent malicious behavior.

From the above description of trust related attacks, we know that trust models are under many security threats that can break the functionality of trust management systems. Therefore, trust models should consider multiple trust factors in order to evaluate trustees accurately. They should also take more defensive measures to avoid the negative effect of bad recommendations so as to improve the stability of trust evaluation in the dynamically hostile environment.

2.3. Outlier detection methods

In Section 2.2, we have already introduced that malicious recommenders which perform some trust related attacks such as bad mouthing attacks and ballot stuffing attacks will provide bad recommendations to the trustor. If the trustor uses these bad recommendations, the accuracy of the recommendation trust evaluation will be reduced. In order to effectively avoid the negative impact of these trust related attacks, these bad recommendations can be regarded as outliers and detected by outlier detection methods. Therefore, the trustor can use a recommendation filtering algorithm based on outlier detection methods to eliminate these bad recommendations when evaluating the recommendation trust of trustees. In this subsection, we introduce some common outlier detection methods and then we will compare and analyze these methods in Section 5.2.1 so as to explain why we choose k-means to filter out bad recommendations.

- **Grubbs' test**: Grubbs' test which was proposed by Grubbs et al. [11] is a statistically based outlier detection method. It is used to detect outliers in one-dimensional data under the assumption that the data is generated by a Gaussian distribution. It calculates the z score of each data instance and compares the z score with the threshold. The z score is calculated by dividing the absolute value of the difference between the data instance and the average value of the data by the standard deviation of the data. A data instance whose z score greater than the threshold will be regarded as an outlier.
- **Box plot**: Box plot [12] is a simple statistical technique to detect outliers in one-dimensional and multi-dimensional data. It first calculates the Inter Quartile Range(IQR) which is the difference between the first quartile(Q_1) and the third quartile(Q_3). Then, data instances greater than $Q3 + 1.5 * IQR$ or less than $Q1 - 1.5 * IQR$ will be regarded as outliers.
- **Isolation forest**: Isolation forest was brought by Liu et al. [13] and can be viewed as the unsupervised counterpart of decision trees. An isolation tree is generated with a given sample set by recursively choosing one random attribute and one random split value of the data on every tree node until the height limit is reached or the terminal leaf contains one distinct data instance. The principle is that outliers have a higher chance of being isolated on an earlier stage than normal data instances. Hence, outliers are expected to have a shorter height in the isolation trees.
- **Local outlier factor(LOF)**: LOF [14] is a well-known approach that first introduced the concept of local outliers. The LOF score for a data instance is based on the average ratio of the instance's neighbors' density to the instance's density. For a normal instance lying in a dense region, its local density will be similar to that of its neighbors, while for an outlier, its local density will be lower than that of its neighbors. Hence, LOF scores of normal instances are close to 1 while outliers' LOF scores are much greater than 1.
- **DBSCAN**: DBSCAN [15] is a density-based clustering algorithm and can be used as an outlier detection method. It has two user-specified parameters that determine the density of the data and it autonomously determines the number of clusters. Users can determine which clusters of data instances are outliers according to the rules set in advance by themselves.
- **k-Means**: k-Means [16] is another clustering algorithm and can also be used for outlier detection. k is the number of clusters and needs to be specified by users in advance. Similar to DBSCAN, users can determine which clusters of data instances are outliers according to their own rules.

3. Related work

In this section, we survey recently proposed trust models for enhancing the security of IoT systems. Guo et al. [17] published a survey and presented a classification of trust models for IoT and this classification contains eight classes based on five trust design dimensions: trust composition, trust propagation, trust aggregation, trust update and trust formation. The trust model we propose also involves these five dimensions. Furthermore, they presented trust related attacks that can perturb the trust computation models: self promoting attacks, bad mouthing attacks, ballot stuffing attacks, selective misbehavior attacks and on–off attacks. We also explain these attacks in detail in Section 2.2 and our trust model can resist these trust related attacks effectively. In the next paragraph, we introduce some specific trust models and their advantages and limitations.

Chen et al. [18] clarified the concept of trust and reputation in IoT and proposed an IoT trust management model based on fuzzy theory. But in their model, a trustor cannot evaluate trustees without direct interactions. To solve this problem, our trust model adopts the recommendation trust evaluation to help the trustor calculate the trust value of trustees indirectly. Nitti et al. [19] proposed two types of trust models: subjective model and objective model. In the subjective model, each trustor calculates and stores the trust value of trustees itself. In the objective model, a distributed hash table is designed for storing the information of each node. But these two trust models are susceptible to malicious nodes in the network. Considering that the trust evaluation is sensitive to context, Saied et al. [20] designed a context-aware and multi-service approach to trust management. The model selects a certain number of historical trust values to calculate the current trust value. But it is difficult to quickly evaluate the trustworthiness when there is not enough trust related information. To solve this problem, Xia et al. [21] designed a kernel-based nonlinear multivariate gray prediction model to predict the direct trust which needs a small amount of historical information. Experimental results indicate the accuracy and convergence rate of the trust model. But, the proportion of malicious nodes is only 30% in their experiments. Our trust model is still accurate when the proportion of malicious nodes is as high as 70%.

Some work brings social attributes to the IoT. A comprehensive model was proposed in [22] and used the social relations of users on the real social platform to establish the social relationship among nodes so as to make the experimental results more persuasive. Chen et al. [9] divided trust into three types based on social attributes: honesty, cooperation and community-interest. The trust model separately calculates the three types of trust and combines them according to the actual scenario. However, it needs a large number of experiments to determine the best weight. When the trustor and the trustee do not interact with each other directly, recommendations are important to trust evaluation. Xia et al. [23] proposed a trust model that divides recommendations into direct recommendations and indirect recommendations and uses direct trust and similarity value to calculate the weight of the two types of recommendations. But their work lacked security analysis of their model. To avoid the impact of bad recommendations, a trust model with clustering technique was proposed in [5] to dynamically filter out attacks related to bad recommendations. Similarly, Chen et al. [6] developed a trust management system that adopts distributed collaborative filtering to select feedback and uses social contacts as filters. However, they did not illustrate how to establish social contacts between nodes. Same as above related work, our model adopts a recommendation filtering algorithm to filter out bad recommendations provided by malicious recommenders. Besides, our model considers three important factors: direct trust, similarity value and confidence level to further reduce the impact of bad recommendations.

Machine learning based trust models have been proposed in recent years. A trust model based on SVM and k-means was presented in [24] to classify the extracted trust features and combine them to produce a final trust value, whereas it is only valid in some situations. Caminha et al. [25] proposed a smart trust management method that can detect on–off attacks. However, this method cannot resist collusion attacks such as bad mouthing attacks. A trust evaluation method based on usage scenarios was presented in [26]. The authors believed that the trustworthiness of the service provided by the target node varies according to the scenario in which the service is used and they used neural network training to obtain the trustworthiness of the service. Alshehri et al. [7] proposed a clustering-driven intelligent method that can filter out dishonest recommenders. In addition, Boudagdigue

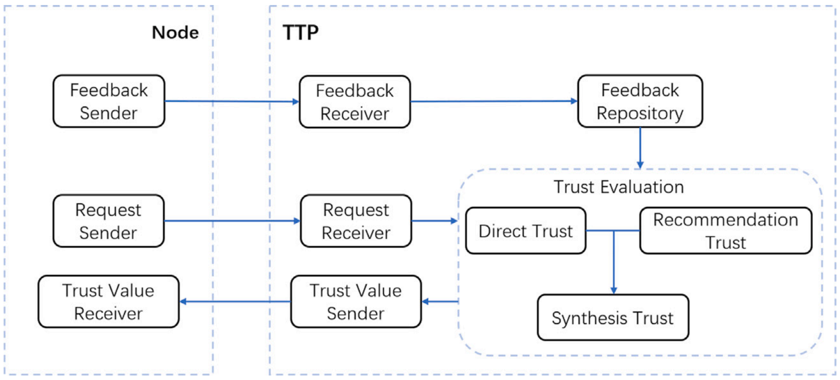


Fig. 1. Architecture of the trust model.

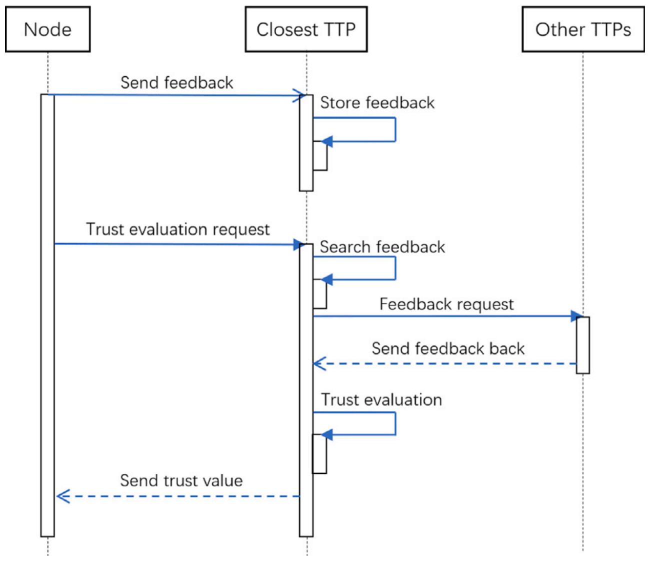


Fig. 2. Process of trust evaluation.

et al. [27] proposed a distributed advanced analytical trust model based on a Markov chain which can effectively resist bad mouthing attacks and ballot stuffing attacks. But they do not explain how to select suitable nodes as recommenders. Wang et al. [4] proposed a novel trust mechanism based on a multilayer structure that solves energy consumption problems. Trust models based on machine learning may require large amounts of data to ensure the performance of trust evaluation. On the contrary, our model uses adaptive weight to combine direct trust and recommendation trust according to the current environment and only requires some necessary information to rapidly evaluate trustees. In addition, the introduction of TTPs can reduce the energy consumption of IoT objects and extend their lifetime.

4. System overview

In this section, we first present the architecture of our trust model and specify the role of each component in the architecture. Then, we give the process of trust evaluation in our trust model so as to explain how the components work together to establish trust relationships among objects in a dynamically hostile IoT environment.

4.1. The proposed system architecture

Fig. 1 illustrates the system architecture of our trust model. There are two main entities in it: nodes and trust third parties (TTPs). Trustor, trustee and recommender are all nodes. Meanwhile, a node can play different roles according to different requirements. Nodes are usually IoT objects with limited capabilities and resources so that they can hardly perform complex computing all the time. To solve such problems, TTPs that provide safe and reliable trust computing environment are introduced into our trust model. We divide the nodes into multiple groups and each group has a TTP responsible for assisting the node in trust evaluation. Each node sends feedback about the services it has received from service providers to its closest TTP in the process of trust evaluation. Hence, our system architecture is a hybrid architecture and TTPs in our architecture play supporting roles. The nodes in the trustor roles really need to perform trust evaluation to evaluate the trust value of trustees. With the help of TTPs to evaluate the trust value of trustees, nodes can save energy as much as possible and thus extend their lifetime.

There are three components in a node, which are feedback sender, request sender and trust value receiver. The details of these three components are as follows.

- Feedback sender: It sends feedback that is provided by nodes to TTPs. If a node is satisfied with the service it has received from the service provider, it will give positive feedback to its closest TTP through the feedback sender.
- Request sender: If a node wants to learn about the trust value of others, it will send a request for trust evaluation to its closest TTP through the request sender.
- Trust value receiver: It receives the trust value of trustees that is sent by TTPs.

There are five components in a TTP, which are feedback receiver, feedback repository, request receiver, trust evaluation module and trust value sender. The details of these five components are as follows.

- Feedback receiver: It is a component that receives feedback from nodes and then sends the feedback to the feedback repository.
- Feedback repository: It is a place where stores feedback from nodes. The feedback in the feedback repository will be used to evaluate the trust value of trustees later.
- Request receiver: It receives the request for trust evaluation from the trustor and then notifies trust evaluation module to evaluate the specific trustee's trust value.
- Trust evaluation module: This module computes the direct trust, recommendation trust and synthesis trust of trustees through feedback from feedback repository and the trust model we proposed.
- Trust value sender: It sends trust value that is evaluated by trust evaluation module to the trustor sending trust request before.

4.2. Process of trust evaluation

In this subsection, we elaborate on how the components mentioned above cooperate with each other in the trust management system in order to implement trust evaluation. Fig. 2 illustrates the process of trust evaluation and the detailed description is as follows:

(1) Each node periodically sends feedback about the services it has received from service providers to its closest TTP via its feedback sender.
(2) Each feedback receiver of the TTP receives feedback from nodes and uploads the feedback to its feedback repository.
(3) A trustor will use request sender to a send trust evaluation request to its closest TTP when it wants to obtain the trust value of the target trustee.
(4) When a TTP receives a trust evaluation request from the trustor, it first searches whether there is feedback about the target trustee in its feedback repository. If not, it will request feedback about that trustee from other TTPs. The TTP which stores the required feedback will send them back.
(5) The TTP utilizes the feedback and its trust evaluation module to evaluate the direct trust, recommendation trust and synthesis trust of the target trustee.
(6) After the work of the trust evaluation module, the TTP sends the target trustee's trust value to the trustor through the trust value sender.
(7) Finally, the trustor receives the trust value of the trustee and then decides whether to receive services provided by the trustee.

5. The proposed trust model

In this section, we propose the concrete methods used in the trust model that can evaluate the trust value accurately and steadily in the dynamically hostile environment.

5.1. Direct trust

We adopt a Bayesian inference model [28] based on beta probability density function to evaluate the direct trust of the trustee. Eq. (1) shows the direct trust of trustor i about trustee j.

$$DT_{ij}^{(t)} = \frac{\alpha_{ij}^{(t)} + 1}{\alpha_{ij}^{(t)} + \beta_{ij}^{(t)} + 2} \tag{1}$$

In Eq. (1), $DT_{ij}^{(t)}$ represents the direct trust of trustor i about trustee j at time t. It is a real number in the range of $[0, 1]$ where 1 indicates complete trust, 0.5 indicates uncertainty and 0 indicates complete distrust. $\alpha_{ij}^{(t)}$ denotes the total number of positive feedback given by trustor i about trustee j from the beginning of trust evaluation to current time t. Similarly, $\beta_{ij}^{(t)}$ is the total number of negative feedback. If the services provided by the trustee can meet the requirements, the trustor will give positive feedback to the trustee. On the contrary, the trustor will give negative feedback.

We consider the influence of feedback is blunted over time because feedback from past interactions cannot accurately reflect the current status of the trustee. So the weight of previous feedback should be reduced. To achieve this, we introduce a time decay function whose value will decrease constantly over time, and adopt a sliding window which only stores and updates the feedback from recent interactions. The sliding window has m time slots in order from its left side to the right side. Each time slot stores the amount of positive and negative feedback during an interaction and the corresponding time when this interaction happened. The rightmost time slot stores the latest feedback that has the most important influence to the direct trust evaluation. Eq. (2) shows the calculation of positive feedback and negative feedback.

$$\begin{aligned} \alpha_{ij}^{(t)} &= \sum_{i=1}^{m} e^{-\lambda(t-t_i)} * \alpha_{ij}^{(t_i)} + pf \\ \beta_{ij}^{(t)} &= \sum_{i=1}^{m} e^{-\lambda(t-t_i)} * \beta_{ij}^{(t_i)} + nf \end{aligned} \tag{2}$$

In Eq. (2), $\alpha_{ij}^{(t_i)}$ denotes the amount of positive feedback provided by trustor i about trustee j at time t_i and $\beta_{ij}^{(t_i)}$ denotes the amount of negative feedback. $e^{-\lambda(t-t_i)}$ is a time decay function and λ is a decay factor that affects the decay rate of the time decay function. m is the size of the sliding window. pf and nf are the amount of positive and negative feedback at time t, respectively.

Another problem we need to solve in the direct trust evaluation is to migrate the risk of on–off attacks. We use a penalty factor to amplify the influence of negative feedback and the trust value of the trustee will decrease faster if it provides the trustor with bad service. Trustor will give negative feedback about the trustee and the weight of negative feedback will be greater with the influence of the penalty factor. Eq. (3) is the final formula to evaluate the direct trust.

$$DT_{ij}^{(t)} = \frac{\alpha_{ij}^{(t)} + 1}{\alpha_{ij}^{(t)} + \beta_{ij}^{(t)} * PF + 2} \tag{3}$$

In Eq. (3), PF is the penalty factor. The calculation of $\alpha_{ij}^{(t)}$ and $\beta_{ij}^{(t)}$ can be found in Eq. (2).

5.2. Recommendation trust

When the trustor does not interact with the trustee directly, it lacks essential information to evaluate the trustee's direct trust. At this time, the trustor needs to request recommendations from recommenders who have interacted with the trustee before and then uses these recommendations to calculate the recommendation trust of the trustee. Under the trust related attacks, the trustor may receive some bad recommendations. To avoid the influence of these attacks, we propose a recommendation filtering algorithm based on k-means to filter out malicious recommenders. For the recommendations provided by remaining recommenders after filtering, some important factors are applied to ensure the accuracy of the recommendation trust.

5.2.1. The choice of k-means

We have already discussed why we need a recommendation filtering algorithm based on outlier detection methods in Section 2.3. Now we analyze the applicability of these outlier detection methods according to the characteristics of bad recommendations and explain why we finally propose a recommendation filtering algorithm based on k-means instead of other outlier detection methods. The bad recommendations about the trustee provided by malicious recommenders are often opposite to the ground truth of the trustee. For example, if the ground truth of a well-behaved trustee is 1, malicious recommenders are likely to give recommendations less than 0.5 to reduce the recommendation trust of the trustee. These behaviors performed by malicious recommenders are called bad mouthing attacks. Ballot stuffing attacks are just the opposite of these behaviors.

When the proportion of malicious recommenders is relatively small, most of the recommendations received by the trustor are close to the ground truth of the trustee. The six outlier detection methods introduced above can all effectively detect bad recommendations in such a case. Then, the trustor can filter out these outliers based on the detection results. However, when the proportion of malicious recommenders increases, the proportion of bad recommendations will also increase. Not all of these outlier detection methods are effective in this situation. The average value of all recommendations is no longer close to the average value of good recommendations, but a value between good recommendations and bad recommendations. The z scores of all recommendations will be less than the fixed threshold and thus grubbs' test cannot detect bad recommendations as outliers. Similarly, the first quartile will fall among bad recommendations instead of good recommendations, resulting in all recommendations being within the specified range of the box plot. Therefore, box plot cannot detect bad recommendations either. Both isolation forest and LOF treat data instances in the sparse area as outliers. The difference is that isolation forest is based on the global distribution of data instances while LOF is based on the local density of data instances. When the proportion of malicious recommenders increases, a bad recommendation will also be in a dense area with other bad recommendations surrounding it. Hence, it is difficult to judge whether the bad recommendation is an outlier or not based on its height in the isolation tree. Similarly, the LOF score of a bad recommendation will be close to 1 because its local density is almost the same as its neighbors'. Obviously, neither isolation forest nor LOF can effectively detect bad recommendations when the proportion of malicious recommenders increases.

DBSCAN and k-means are both clustering-based outlier detection methods. The former autonomously determines the number of clusters based on the density of data instances. The latter determines the number of clusters according to the user-specified parameter k and divides data instances into clusters based on the distance between them and the centroids. The same is that both of them need to determine which clusters of data instances are outliers according to the user-specified rules. In the recommendation trust evaluation, we can take the direct trust of the trustor about the recommenders and the recommendations provided by the recommenders as data instances. The reason behind that is the average value of the direct trust of the trustor about good recommenders is larger. The recommendations in the cluster which centroid's first value is the largest can be regarded as good recommendations and others will be deemed to be bad. Then, the trustor can filter out bad recommendations from all recommendations it received based on the clustering results. Whether it is DBSCAN or k-means, good recommendations and bad recommendations will be divided into different clusters even if the proportion of malicious recommenders increases. Therefore, both of them are effective for detecting bad recommendations according to the rules we set. The space complexity of DBSCAN is $O(m)$ where m is the number of data instances. The space complexity of k-means is $O(m+k)$ where k is the number of clusters specified by users. In the recommendation trust evaluation, we set k to 2. As a result, the space complexity of k-means is approximately equal to the space complexity of DBSCAN. The time complexity of DBSCAN is $O(mlogm)$. The time complexity of k-means is $O(Ikm)$ where I is the number of iterations specified by users. Because I and k are much smaller than m, the time complexity of k-means can be regarded as $O(m)$. We can conclude that k-means is more efficient than DBSCAN in terms of time complexity. Based on the above comparative analysis, we can conclude from both effectiveness and efficiency that the choice of k-means to detect bad recommendations is better than other outlier detection methods. Consequently, we propose a recommendation filtering algorithm based on k-means.

5.2.2. Recommendation filtering algorithm based on k-means

We take the direct trust of the trustor about the recommenders and the recommendations provided by the recommenders as vectors which will be taken by the recommendation filtering algorithm as inputs. Through the filtering algorithm, the vectors will be divided into two clusters. The cluster whose centroid is larger will be selected and the corresponding recommenders will be regarded as recommenders after filtering.

Algorithm 1 illustrates the recommendation filtering algorithm in detail. The inputs are a list of recommenders $R = \{r_1, r_2, \dots, r_m\}$ and the max number of interactions $Iterations^{max}$. The outputs are a list of recommenders $R' = \{r'_1, r'_2, \dots, r'_n\}$ that remain after filtering. For each recommender r_k, the algorithm constructs a vector of two values: the direct trust of the trustor about the recommender and the recommendation of the recommender about the trustee. At the beginning of filtering, the algorithm randomly selects two vectors as initial centroids of clusters. The core part of the filtering algorithm is based on the k-means clustering algorithm (Lines 7–29). It separately calculates the Euclidean distance of each vector and the centroid of two clusters. Then, each vector will be added to the cluster closer to it (Lines 10–18). The centroid of each cluster will be recalculated after all vectors are added to the corresponding clusters until it does not change any more.

Algorithm 1 Recommendation Filtering Algorithm

```
Input: $R = \{r_1, r_2, \dots, r_m\}$, $Iterations^{max}$
Output: $R' = \{r'_1, r'_2, \dots, r'_n\}$
1: for each $r_k \in R$ do
2:     $Construct\ vector\ (DT_{ir_k}; DT_{r_k j})$
3: end for
4: Initialize:
5:     $randomly\ select\ two\ vectors\ (DT_{ir_x}; DT_{r_x j}), (DT_{ir_y}; DT_{r_y j})$
6:     $Iterations = 1,\ R' = \emptyset$
7: repeat
8:     $C_1 = \emptyset,\ C_2 = \emptyset$
9:     $Flag = false$
10:    for each $(DT_{ir_k}; DT_{r_k j})$ do
11:        $dist_1 = \sqrt{(DT_{ir_k} - DT_{ir_x})^2 + (DT_{r_k j} - DT_{r_x j})^2}$
12:        $dist_2 = \sqrt{(DT_{ir_k} - DT_{ir_y})^2 + (DT_{r_k j} - DT_{r_y j})^2}$
13:        if $dist_1 <= dist_2$ then
14:            $C_1 = C_1 \cup (DT_{ir_k}, DT_{r_k j})$
15:        else
16:            $C_2 = C_2 \cup (DT_{ir_k}, DT_{r_k j})$
17:        end if
18:    end for
19:    $(DT'_{ir_x}; DT'_{r_x j}) = (\frac{1}{|C_1|} \sum_{DT_{ir_k} \in C_1} DT_{ir_k}; \frac{1}{|C_1|} \sum_{DT_{r_k j} \in C_1} DT_{r_k j})$
20:    $(DT'_{ir_y}; DT'_{r_y j}) = (\frac{1}{|C_2|} \sum_{DT_{ir_k} \in C_2} DT_{ir_k}; \frac{1}{|C_2|} \sum_{DT_{r_k j} \in C_2} DT_{r_k j})$
21:    if $(DT'_{ir_x}; DT'_{r_x j}) \neq (DT_{ir_x}; DT_{r_x j})$ then
22:        $(DT_{ir_x}; DT_{r_x j}) = (DT'_{ir_x}; DT'_{r_x j})$
23:        $Flag = true$
24:    end if
25:    if $(DT'_{ir_y}; DT'_{r_y j}) \neq (DT_{ir_y}; DT_{r_y j})$ then
26:        $(DT_{ir_y}; DT_{r_y j}) = (DT'_{ir_y}; DT'_{r_y j})$
27:        $Flag = true$
28:    end if
29: until $Iterations > Iterations^{max}$ or $Flag == false$
30: if $DT_{ir_x} >= DT_{ir_y}$ then
31:     $C_{filter} = C_1$
32: else
33:     $C_{filter} = C_2$
34: end if
35: for each $(DT_{ir_k}; DT_{r_k j}) \in C_{filter}$ do
36:     $R' = R' \cup r_k$
37: end for
38: return $R'$
```

After the clustering algorithm, the vectors are divided into two clusters. One of the clusters includes vectors corresponding to the good recommenders and the centroid's first value of it is larger because the average direct trust of the good recommenders is larger. We consider the first cluster as the trustworthy one. Finally, the recommenders corresponding to each vector in the trustworthy cluster are subsequently used for the recommendation trust evaluation. Through the recommendation filtering algorithm, the impact of trust related attacks such as bad mouthing attacks and ballot stuffing attacks can be minimized by filtering out the malicious recommenders. But the recommendations provided by the remaining recommenders may not be all used. We need to consider some important factors that affect the accuracy of the recommendation trust.

5.2.3. Evaluation of recommendation trust

Although the recommendation filtering algorithm can effectively resist some trust related attacks, it may not filter out all the malicious recommenders. In addition, the well-behaved recommenders may not evaluate the trustee accurately due to insufficient interactions and thus cannot provide precise recommendations. To solve those problems that the filtering algorithm cannot deal with, we apply three important factors in the evaluation of the recommendation trust.

In human society, we usually trust information provided by people who we believe. Similarly, in the trust model, the trustor tends to use the recommendations provided by recommenders who are highly rated by the trustor. As a result, the direct trust of the trustor about the recommender is needed in the evaluation of the recommendation trust. The second important factor is the similarity of the direct trust evaluation. Generally speaking, the trustor is more willing to receive recommendations from recommenders who have similar views with it. The similar views mean that the trustor and recommenders give similar evaluation of the direct trust to the trustee who provides the same service. Eq. (4) shows how to calculate the similarity of the direct trust evaluation between the trustor and the recommender.

$$S_{ir_k}^{(t)} = 1 - \frac{\sum_{l \in Set(i,r_k)} |DT_{il} - DT_{r_kl}|}{|Set(i,r_k)|} \tag{4}$$

In Eq. (4), $S_{ir_k}^{(t)}$ denotes the similarity of direct trust evaluation between trustor i and recommender r_k. It falls in the interval of $[0,1]$ where 1 means that the trustor and the recommender give exactly the same evaluation for each trustee. $Set(i,r_k)$ represents trustees common to i and r_k, and $|Set(i,r_k)|$ is the number of common trustees. Assuming that a recommender only gives the correct recommendations about part of trustees, the direct trust evaluation of the trustor and the recommender about the same trustee may not be similar. In such a case, the similarity of the direct trust evaluation will be very small. Therefore, the trust model can resist selective misbehavior attacks by using the factor of similarity in calculating.

The last factor is the confidence level of the recommender about the trustee. The confidence level reflects the number of interactions between the recommender and the trustee. The higher the confidence level is, the more the interactions between them are. Consequently, the recommender with a high confidence level is more popular because it can evaluate the direct trust of the trustee accurately through sufficient interactions. Eq. (5) shows the calculation of the confidence level. It evolves from the beta distribution standard deviation.

$$C_{r_kj}^{(t)} = 1 - \sqrt{\frac{12(\alpha_{r_kj}^{(t)}+1)(\beta_{r_kj}^{(t)}+1)}{(\alpha_{r_kj}^{(t)}+\beta_{r_kj}^{(t)}+2)^2(\alpha_{r_kj}^{(t)}+\beta_{r_kj}^{(t)}+3)}} \tag{5}$$

In Eq. (5), $C_{r_kj}^{(t)}$ denotes the confidence level of recommender r_k about trustee j at time t. $\alpha_{r_kj}^{(t)}$ and $\beta_{r_kj}^{(t)}$ is the accumulated positive and negative feedback given by the k'th recommender r_k about target trustee j. Eq. (6) shows the way to calculate them and it is similar to Eq. (2).

$$\begin{aligned} \alpha_{r_kj}^{(t)} &= \sum_{i=1}^{m} e^{-\gamma(t-t_i)} * \alpha_{r_kj}^{(t_i)} \\ \beta_{r_kj}^{(t)} &= \sum_{i=1}^{m} e^{-\sigma(t-t_i)} * \beta_{r_kj}^{(t_i)} \end{aligned} \tag{6}$$

In Eq. (6), m is the size of the sliding window between r_k and j. γ and σ are decay factors of the time decay function. $\alpha_{r_kj}^{(t_i)}$ and $\beta_{r_kj}^{(t_i)}$ is the positive and negative feedback at time t_i, respectively. We combine the three important factors explained above and give the calculation of the recommendation trust in Eq. (7).

$$RT_{ij}^{(t)} = \sum_{k=1}^{n} \frac{DT_{ir_k}^{(t)} S_{ir_k}^{(t)} C_{r_kj}^{(t)}}{\sum_{k=1}^{n} DT_{ir_k}^{(t)} S_{ir_k}^{(t)} C_{r_kj}^{(t)}} * DT_{r_kj}^{(t)} \tag{7}$$

In Eq. (7), $RT_{ij}^{(t)}$ is the recommendation trust of trustee j calculated by trustor i. n is the number of recommenders after filtering. $DT_{r_kj}^{(t)}$ is the recommendations provided by recommender r_k about trustee j. Its correctness depends on the behavior of the recommender. The utilization of $DT_{ir_k}^{(t)}$, $S_{ir_k}^{(t)}$ and $C_{r_kj}^{(t)}$ can minimize the impact of bad and imprecise recommendations and thereby improve the precision of the recommendation trust evaluation.

5.3. *Synthesis trust*

Neither direct trust nor recommendation trust can comprehensively reflect the trustee's trustworthiness. Hence, our trust model uses synthesis trust that is calculated by combining direct trust and recommendation trust. Eq. (8) shows the calculation of the synthesis trust.

$$T_{ij}^{(t)} = \omega DT_{ij}^{(t)} + (1-\omega)RT_{ij}^{(t)} \tag{8}$$

In Eq. (8), $T_{ij}^{(t)}$ denotes the synthesis trust of trustor i about trustee j. Its range is between 0 and 1, where 1 means complete trust while 0 means complete distrust. $DT_{ij}^{(t)}$ and $RT_{ij}^{(t)}$ is direct trust and recommendation trust calculated by Eqs. (3) and (7). ω is a weight that weighs the importance of direct trust and recommendation trust. It falls in the interval $[0,1]$ and the bigger it is, the more important direct trust is. The selection of ω is pivotal to the trust model. We adopt an adaptive weight that can adjust automatically according to the dynamically hostile environment. The utilization of adaptive weight can resist trust related attacks such as bad mouthing attacks and ballot stuffing attacks so that improving the accuracy of trust evaluation. Eq. (9) illustrates the calculation of weight ω.

$$\omega = \begin{cases} 1-\theta^{e^{-\Delta t}IN} & \overline{DT_{ir}} \geq DT_{threshold}, \\ 1 & \overline{DT_{ir}} < DT_{threshold}. \end{cases} \tag{9}$$

In Eq. (9), the calculation of ω is divided into two parts. The principle of separate calculation is to compare $\overline{DT_{ir}}$ and $DT_{threshold}$. $\overline{DT_{ir}}$ denotes the average value of direct trust of trustor i about all recommenders. $DT_{threshold}$ is the threshold of direct trust and is set to 0.5 by default. If $\overline{DT_{ir}}$ is less than $DT_{threshold}$, ω will be equal to 1. It means that trustor i will only use the direct trust depending on the direct interactions with the target trustee if most of the recommenders are malicious. This way of calculating weight can prevent the trustor mistaking a good trustee as a malicious one when the proportion of malicious recommenders is high. When $\overline{DT_{ir}}$ is equal to or greater than $DT_{threshold}$, the calculation of ω is related to the number of interactions IN between trustor i and trustee j, and Δt that the difference between the current time and the time of last interaction. The high number of interactions means that the trustor has already adequately known about the trustee, so the direct trust of the trustor about the trustee is more accurate. But if the trustor and the trustee have not interacted recently, even if they interacted with each other many times long time ago, the direct trust still cannot reflect the current trustworthiness of the trustee. In such a case, we regulate the importance of the direct trust via Δt. The adaptive weight can be dynamically adjusted according to the current interaction situation to adapt to the dynamically hostile environment.

6. Experimental results and analysis

The detailed performance evaluation of our work is done in two main parts. In the first part, we compare our proposed system architecture based on TTPs with the centralized architecture and the distributed architecture in terms of energy consumption. In the second part, we first validate the effectiveness of the recommendation trust evaluation and the adaptive weight. Then, we compare our trust model with three related models: TBSM [9], NRB [23] and NTM [4]. These three related models all adopted some methods to avoid the negative impact caused by malicious nodes on trust evaluation. In TBSM [9], they established social relationships between nodes and used these relationships to help the trustor not to use bad recommendations provided by malicious recommenders. In NRB [23], they divided recommendation trust into direct recommendation trust and indirect recommendation trust according to whether the trustor interacts with the trustee directly. In direct recommendation trust, the trustor selects common nodes between it and the trustee as recommenders. The trustor can easily know the trustworthiness of recommenders by interacting with them directly and thus do not use incorrect recommendations when calculates the recommendation trust of the trustee. In NTM [4], they did not use too high or too low recommendations provided by recommenders to avoid bad mouthing attacks and ballot stuffing attacks. To ensure the fairness and the effectiveness of the comparative experiments, we first use the best parameter values mentioned in their paper for the unique parameters of each trust model. Then, for the network simulation parameters such as the number of nodes, the moving range of nodes and the mobility model of nodes, we use the same parameter settings explained below.

We perform comprehensive experiments based on the ns-3 simulator. The ns-3 simulator is a discrete-event network simulator and the components of our trust model can be added to it. Table 1 lists the basic parameter values used to configure the network for experiments and the default parameter values of our trust model. We consider an IoT environment with 200 nodes and 20 TTPs. The nodes randomly move in an area of 1000×800 square meter with a speed of 20 m/s while TTPs are uniformly distributed in this area. Each node selects the closest TTP on the distance to help it store the feedback about neighbor nodes and evaluate the trust value of trustees every round of trust evaluation. The mobility model we use in ns-3 is the random waypoint which is very similar to the real world movement [29]. We choose the AODV routing protocol which has been realized in ns-3 and can support our trust model well. λ, γ and σ are decay factors of the decay function in the calculation of the number of feedback. The size of the decay factor can control the speed of decay. m is the size of the sliding window. θ is the parameter used to calculate the adaptive weight in Eq. (9). For the sake of determining the exact value of the constant terms in the above equation, we need to do multiple sets of experiments in the given network environment to obtain better results. For the three relevant trust models in the comparative work, we adopt the same method to obtain the values of some constant terms in their equation. This is to ensure the fairness of the comparative experiments.

To verify the attack resistance of our trust model, a proportion of malicious nodes that will perform trust related attacks is randomly selected from all nodes. Trust related attacks in our experiments include on–off attacks, self promoting attacks, bad mouthing attacks, ballot stuffing attacks and selective misbehavior attacks which are mentioned in Section 2. Under on–off attacks, malicious nodes will perform these attacks in a random period of time. Under self promoting attacks, malicious recommenders will give recommendations about themselves to the trustor. Malicious recommenders will give false recommendations that are opposite to the ground truth about trustee when performing bad mouthing attacks and ballot stuffing attacks. However, malicious recommenders will not perform trust related attacks to all trustors. They will randomly perform these attacks to part of the trustors. These are selective misbehavior attacks. The range of the proportion is from 10% to 70% and the default value is 30%. The trust evaluation interval is 100 s and the total simulation time is 10000 s. There are 14,833 data packets related to trust messages sent and received by nodes in the entire network and the packet loss rate is 2.7%. These trust messages include feedback messages about neighbors, trust value request messages, feedback messages from other TTPs and such similar messages sent by the nodes and TTPs.

The dynamically hostile environment in our experiments includes two aspects. First, nodes randomly move with random directions according to the random waypoint mobility model. As a result, the neighbors of a node are constantly changing. Second, the proportion of malicious nodes changes from 10% to 70%. Our experiments focus on the effectiveness of trust evaluation mechanisms that we adopt and the

Table 1
Simulation parameters.

Parameter	Value
Nodes	200
Area	1000 m × 800 m
TTPs	20
Speed	20 m/s
PF	1.5
Radio range	100 m
Malicious ratio	30%
$DT_{threshold}$	0.5
Routing protocol	AODV
Mobility	Random waypoint model
λ	0.05
γ	0.7
m	5
σ	0.7
θ	0.1

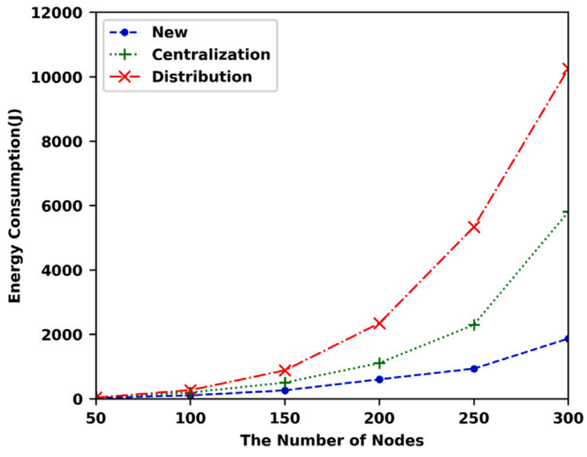


Fig. 3. Energy consumption varying the number of nodes.

comparison with other relevant models in terms of convergence rate, stability and attack resistance in the dynamically hostile environment. For simplicity, we assume that an honest node has a 95% probability of generating positive feedback, and a malicious node has a 95% probability of generating negative feedback. This premise simplifies the way of sending and receiving service requests between the trustor and the trustee. Our trust model depends on the feedback provided by the trustor, so this premise does not affect the function of the trust model and we can pay more attention to the performance of the trust model in the dynamically hostile environment.

The trust value in our experiments is between 0 and 1 where 1 means completely trustworthy and 0 means completely untrustworthy. We can understand 0 and 1 as the likelihood of being trustworthy. Therefore, even if the trust evaluation results are close, the meaning of the trust value is different. We use the mean absolute error (MAE) to measure the accuracy of trust evaluation. The smaller the MAE is, the higher the accuracy of the trust evaluation is. The comparative results show that our model converges fast and remains stable in the trust evaluation. Besides, when the proportion of malicious nodes reaches 70%, the mean absolute error (MAE) of our trust evaluation is still less than 0.05 while the MAE in others becomes larger. It means that our model is more resistant to trust related attacks than other models.

6.1. Energy consumption of the system architecture

In this subsection, we compare our proposed system architecture based on TTPs with the centralized architecture and the distributed architecture in terms of energy consumption. We apply our proposed trust model to the three architectures, respectively. The trust model in

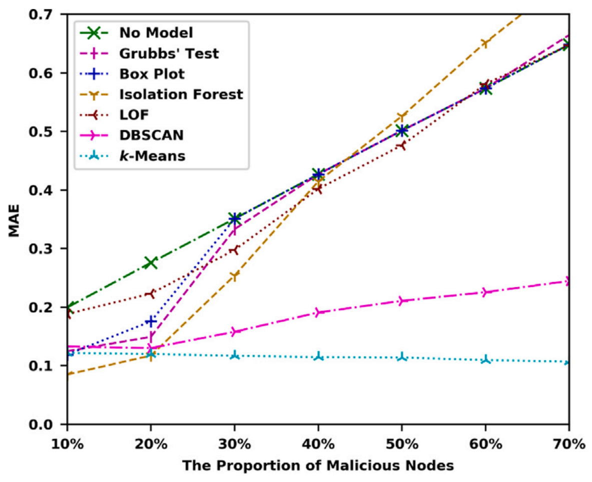


Fig. 4. Effectiveness of k-means.

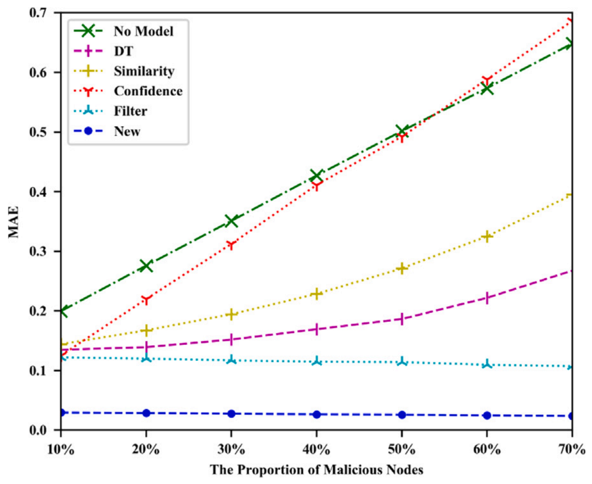


Fig. 5. Effectiveness of recommendation trust evaluation.

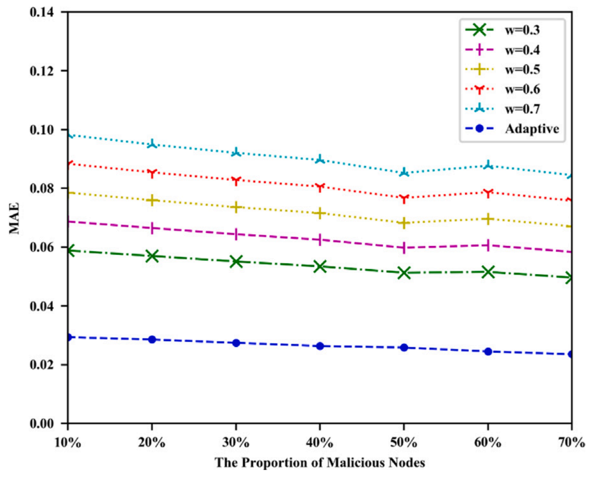


Fig. 6. Effectiveness of adaptive weight.

our architecture has been introduced in Section 4. The implementation of the trust model in the centralized architecture and distributed architecture is slightly different from our proposed architecture based on TTPs. In the centralized architecture, there is only one TTP at the center of the network, so each node periodically sends feedback to the same TTP which is responsible for handling all trust evaluation requests. There is no TTP in the distributed architecture and each node evaluates the trust value of other nodes by itself. Therefore, each node sends and receives feedback with its neighbor nodes for the trust evaluation.

In the trust management system, the energy consumption of a node is mainly divided into two parts: computing energy consumption and communication energy consumption. Computing energy consumption mainly refers to the energy consumption when nodes generate feedback. This part of the energy consumption is the same in the three architectures. Additionally, in the distributed architecture, each node needs to evaluate the trust of others, resulting in their computing energy consumption being greater than the centralized architecture and our proposed architecture. As for communication energy consumption, it is closely related to the number of trust messages sent and received by nodes in the network. We focus on the communication energy consumption of nodes in the three architectures, which can be obtained through the ns-3 energy module.

Fig. 3 shows the results of the overall energy consumption of nodes by varying the number of nodes in the three architectures. In view of that, we conclude that energy consumption increases as the number of nodes increases. But the difference is that the energy consumption increases slowly in our proposed architecture while it increases rapidly in both the centralized architecture and the distributed architecture. Meanwhile, our proposed architecture achieves better energy saving than the other two architectures. The reason behind that is the nodes in our proposed architecture only need to transmit trust messages between their own TTP and themselves. Beyond that, they do not have any additional energy consumption related to trust messages. Therefore, the overall energy consumption does not increase rapidly even if the number of nodes increases. In the centralized architecture, some nodes cannot directly transmit trust messages with the only TTP which is not in their transmission range, and need other nodes to help them relay trust messages. The relay of trust messages increases the energy consumption of some nodes, resulting in higher overall energy consumption than our proposed architecture. Besides, the increase in the number of nodes which all send trust messages to the same TTP will be more likely to cause packet loss, so the retransmission of trust messages further increases the overall energy consumption. In the distributed architecture, each node needs to transmit trust messages with all neighboring nodes, so the average number of trust messages transmitted by each node is more. Compared with the first two architectures, this will undoubtedly cause each node to consume more energy, leading to higher overall energy consumption. Increasing the number of nodes leads to a high density of network which increases the average number of neighbor nodes of each node, which in turn significantly increases the energy consumption of each node and results in the maximum overall energy consumption of the distributed architecture. Obviously, our proposed architecture shows better performance in terms of energy consumption than the other two architectures.

6.2. Effectiveness, convergence and attack resistance of the trust model

6.2.1. Effectiveness of k-means

In this subsection, we compare the recommendation filtering algorithm based on k-means with other outlier detection methods introduced in Section 2.3 to further justify the effectiveness of our recommendation filtering algorithm. Fig. 4 shows the MAE of the recommendation trust evaluation when using recommendation filtering algorithms based on different outlier detection methods. In order to ensure the fairness of the comparative experiments, we conduct multiple sets of experiments to determine the values of the parameters required for each outlier detection method. The proportion of malicious nodes increases from 10% to 70%. When each malicious node plays the role of a recommender, it will be a malicious recommender which performs trust related attacks and provide bad recommendations to the trustor. We can see from Fig. 4 that when the proportion of malicious

Table 2
Trust evaluation convergence of the trust model.

Node type	Trust model			
	Our model	TBSM	NRB	NTM
Honest	0.97	0.94	0.93	0.78
Malicious	0.02	0.05	0.06	0.20
Honest to malicious	0.03	0.09	0.15	0.24

Table 3
Trust evaluation accuracy rate of the trust model.

Trust model	Accuracy rate
Our model	97.35%
TBSM	90.73%
NRB	91.45%
NTM	71.23%

nodes is small, all outlier detection methods can work and reduce the MAE of recommendation trust evaluation. Due to the different characteristics of each method, the filtering effect is also different, resulting in different MAE. When the proportion of malicious nodes increases, grubbs' test, box plot and LOF have almost no effect on the MAE of the recommendation trust evaluation, which means that they cannot effectively detect the bad recommendations. We have already analyzed the reasons why these three methods do not work in sub Section 5.2.1. We can observe that the MAE of the recommendation filtering algorithm based on isolation forest is even larger than the MAE without any model when the proportion of malicious nodes reaches 50%. The reason is that isolation forest detects outliers based on the global distribution of data instances. When the proportion of malicious nodes increases, the area where the bad recommendations are located becomes denser and the area where the good recommendations are located is relatively sparser. This causes isolation forest to mistake good recommendations for outliers and increases the MAE of recommendation trust evaluation. We can judge that the filtering effect of DBSCAN is not as good as k-means according to the MAE of the recommendation trust evaluation. DBSCAN marks the data instances as core points, border points and noise points in the process of clustering. When the proportion of malicious nodes increases, some good recommendations that are not in the dense area will be marked as noise points. This part of the recommendations will not be in the cluster selected by the trustor and will be filtered out. In this case, the MAE of the recommendation trust evaluation will increase slightly. But it will not happen in the recommendation filtering algorithm based on k-means which only divides the good recommendations and the bad recommendations into two different clusters. Through comparative experiments based on different outlier detection methods, we justify the effectiveness of our recommendation filtering algorithm based on k-means once again.

6.2.2. Effectiveness of recommendation trust evaluation

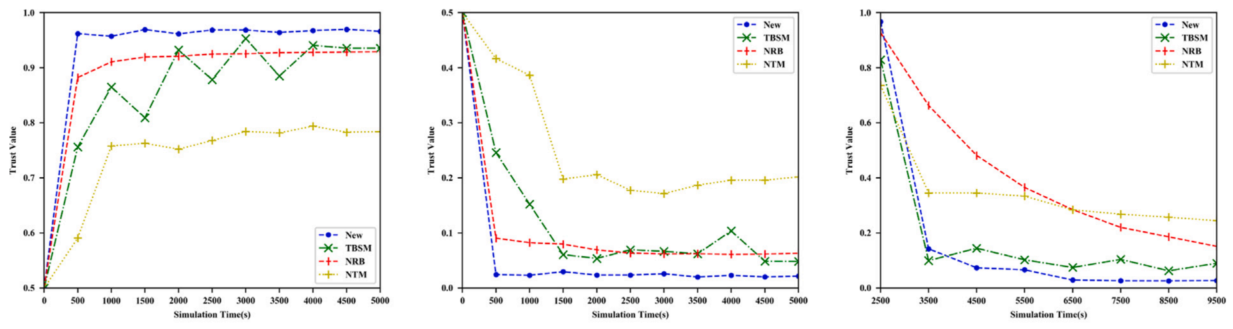
In this subsection, we validate the effectiveness of our recommendation trust evaluation by separately observing the impact of factors we discuss in Section 5.2. Fig. 5 shows the MAE of recommendation trust evaluation when considering different factors and the proportion of malicious nodes increases from 10% to 70%. We can observe that the MAE increases rapidly without any defensive measure because the trustor will use bad recommendations provided by malicious recommenders in the environment to evaluate the recommendation trust of the trustee. These bad recommendations will seriously affect the accuracy of recommendation trust and thus lead to a bigger MAE. When we use the proposed recommendation filtering algorithm, the MAE is around 0.1 and remains stable regardless of the increase of the proportion of malicious nodes. Because the recommendation filtering algorithm we proposed can effectively filter out bad recommendations. Even if the proportion of malicious nodes in the environment increases, the filtering algorithm can still select trustworthy recommenders. The utilization of the direct trust of the trustor about the recommender and the similarity of the direct trust evaluation between the trustor and the recommender can also resist trust related attacks when the proportion is not too high. The reason is that the direct trust and the similarity of honest recommenders are higher and thus the weight of their recommendations is bigger. The confidence level has no effect on evaluation because it only reflects the quantity of interactions between the recommender and the trustee. The recommendation filtering algorithm may not be effective when the proportion of malicious recommenders exceeds half. In this case, we must use the direct trust, the similarity and the confidence level of the recommender. These three important factors can reduce the weight of the bad recommendations as much as possible. When we combine the filtering algorithm and the three important factors together, the MAE approaches 0 and keeps stable. Through the experimental results, we demonstrate that the recommendation trust evaluation used in our trust model can effectively exclude bad and imprecise recommendations.

6.2.3. Effectiveness of adaptive weight

Fig. 6 shows the MAE of the trust evaluation using different weights when the proportion of malicious nodes increases from 10% to 70%. We can see that the MAE is the least when using the adaptive weight in the trust evaluation. The reason is that the adaptive weight can adjust automatically according to the current interaction situation between the trustor and the trustee so as to adapt to the dynamically hostile environment. If the trustor has frequently interacted with the trustee recently, it can judge whether the trustee is trustworthy from its direct trust toward the trustee and the direct trust is more credible than the recommendation trust calculated from recommendations provided by other recommenders in such case. However, a fixed weight cannot freely regulate the importance of the direct trust and the recommendation trust, which results in a larger MAE. For example, if the weight of the direct trust is 0.3 while the weight of the recommendation trust is 0.7, the convergence rate of the trust evaluation is fast even if the trustor does not interact with the trustee directly. The reason behind that is the trustor can rely on the recommendation trust which is evaluated from recommendations provided by recommenders. But if the proportion of malicious nodes is large, most of the recommendations are wrong and thus affect the accuracy of the recommendation trust. However, the weight of the recommendation trust will not be high when we use the adaptive trust. The trustor determines the weight of the direct trust and the recommendation trust according to the number of interactions between the trustor and the trustee, the time of interaction and the average trust value of recommenders. In conclusion, the adaptive weight in our trust model effectively combines the direct trust with the recommendation trust and reduces the MAE of trust evaluation.

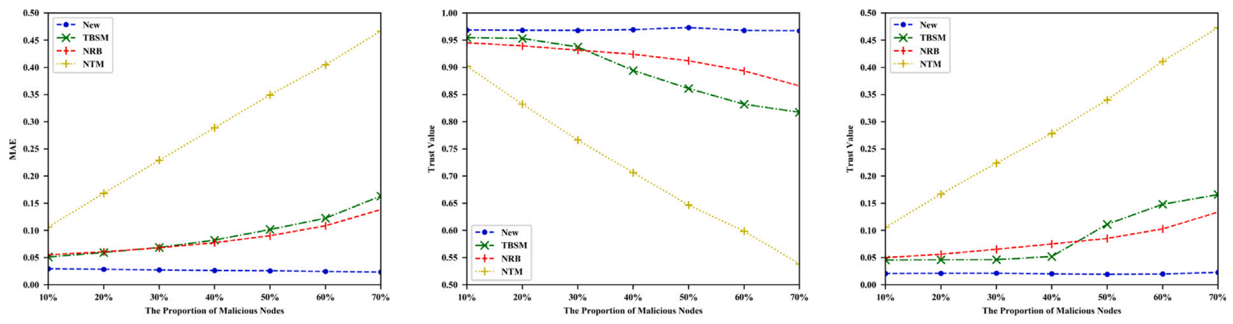
6.2.4. Convergence rate and stability of the trust model

In this subsection, we investigate the convergence rate and stability of our trust model and three relevant trust models (TBSM [9], NRB [23] and NTM [4]) are used for comparison. Fig. 7(a) shows the trust evaluation of the trustor about an honest trustee who is randomly selected and its ground truth is constant at 1 over time. We observe that our trust model converges faster than other trust models and remains stable with the minimum trust deviation. The convergence rate of NRB [23] and NTM [4] is slow and the stability of them is poor because they cannot effectively filter out bad recommendations and reduce the impact of malicious nodes. In our trust model and TBSM [9], the convergence rate is fast and the stability is good because we can effectively filter out bad recommendations. Our trust model is better because the adaptive weight we proposed can better combine the direct trust and the recommendation trust to reduce the MAE of the trust evaluation. Fig. 7(b) shows the trust evaluation about a malicious trustee whose ground truth is always 0. Our trust model also approaches ground truth faster. The reason is the same as the



(a) Honest node (b) Malicious node (c) From honest to malicious

Fig. 7. Convergence rate and stability of the trust model.



(a) MAE (b) Honest node (c) Malicious node

Fig. 8. Attack resistance of the trust model.

results in Fig. 7(a). Fig. 7(c) shows a randomly selected trustee whose behavior changes from honest to malicious after $t = 2500$ s. We can see that after the status changes, the trust evaluation in our trust model converges toward the new ground truth quickly while NRB and NTM converge slowly. The reason is that our trust model uses the sliding window and the time decay function to store recent feedback and reduce the impact of previous feedback. Besides, our trust model uses the filtering algorithm and the adaptive weight to make better use of recommendations provided by other recommenders.

Table 2 shows the trust evaluation of the trustor about a trustee when using different trust models. The second row of Table 2 shows the trust value of an honest trustee when t = 5000 s. We can observe that the trust evaluation result of our model is closest to the ground truth which is 1. Similarly, the third row is about the trust value of a malicious trustee whose ground truth is always 0. Our trust model is also closest to the ground truth. The results in the fourth row are similar to the third row. The trustee changes from honest to malicious after $t = 2500$ s and the trust evaluation results are at t = 9500 s. Our trust model still performs better than the other three models. Therefore, our trust model is better than the other three models in the convergence rate and stability of the trust evaluation.

6.2.5. *Attack resistance of the trust model*

In this subsection, we investigate the resistance of trust related attacks in our trust model and we still choose the above three relevant models for comparison. The malicious nodes randomly selected can perform trust related attacks we summarized in Section 2.2 and the proportion of malicious nodes is from 10% to 70%. From Fig. 8(a), we can see that the MAE of trust evaluation approaches 0 and is hardly affected by the proportion of malicious nodes in our trust model. The reason is that our model can exclude most of the bad recommendations by the filtering algorithm and minimize the impact of malicious recommenders through three important factors. On the contrary, NTM has no special measure to resist trust related attacks, so the MAE rises rapidly when the proportion of malicious nodes increases. TBSM and NRB both decrease the weight of bad recommendations. But they cannot be efficient when there are a large number of malicious nodes in the environment. Fig. 8(b) shows the trust evaluation of an honest trustee randomly selected and Fig. 8(c) shows the evaluation of a malicious trustee. We observe that the trust bias between the estimated value and ground truth is minimal in our model and remains stable at various proportion of malicious nodes. It means that our trust model performs better in the dynamically hostile environment.

Table 3 shows the average trust evaluation accuracy rate of these four trust models. We can see that our trust model is the most accurate with an average accuracy rate of 97.35%. We have analyzed the reason why our trust model can evaluate trustees accurately and is resistant to trust related attacks. The experimental results further validate that our trust model can have high resistance toward attacks even in the extremely hostile environment.

7. Conclusion

In this paper, we design and implement an adaptive trust model based on the recommendation filtering algorithm for the IoT systems. We first propose a system architecture based on TTPs that can save energy of objects. In the direct trust evaluation, we use a sliding window and a time decay function to reduce the impact of previous feedback so as to converge faster. In the recommendation trust evaluation, we design a recommendation filtering algorithm to filter out bad recommendations and introduce three important factors to minimize the impact of trust related attacks. Finally, we use an adaptive weight to combine direct trust and recommendation trust into the synthesis trust in order to improve the convergence rate and accuracy of the trust evaluation. Simulation experiments validate that the utilization of the recommendation trust evaluation and the adaptive weight can minimize the MAE and adapt to the dynamically hostile environment. Compared with TBSM [9], NRB [23] and NTM [4], our trust model converges faster and remains more stable. The MAE is always less than 0.05 when the proportion of malicious nodes is from 10% to 70%. Experimental results further verify that our adaptive trust model enables fast and accurate trust evaluation and resists trust related attacks in the dynamically hostile environment.

However, there are still some limitations in our work. At present, TTPs are fixed in our system architecture. So the trust model based on our architecture needs to select TTPs in advance. In the future work, we will study how to design a TTP designation algorithm to automatically determine TTPs based on factors such as the trust value of IoT objects, the remaining energy and the computing power. Furthermore, we plan to further improve the resistance to trust related attacks of our trust model so that it will still accurately evaluate the IoT objects when the proportion of malicious nodes exceeds 70%. Then, we will design and implement a trust management system based on our proposed model. Therefore, the feedback information used in trust evaluation will be generated based on the actual service provided by nodes. We will also validate our trust model in a real IoT environment.

CRediT authorship contribution statement

Guozhu Chen: Conceptualization, Methodology, Software, Writing - original draft. **Fanping Zeng:** Writing - review & editing, Resources, Supervision. **Jian Zhang:** Writing - review & editing. **Tingting Lu:** Writing - review & editing. **Jingfei Shen:** Writing - review & editing. **Wenjuan Shu:** Writing - review & editing.

Declaration of competing interest

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

Acknowledgments

This work is supported partly by the National Key R&D Program of China 2018YFB2100300 and 2018YFB0803400, the National Key Basic Research (973) Program of China (2014CB340701) and National Natural Science Foundation of China (NSFC) under grant 61772487.

References

[1] L. Atzori, A. Iera, G. Morabito, The internet of things: A survey, Comput. Netw. 54 (15) (2010) 2787–2805.
[2] X. Li, R. Lu, X. Liang, X. Shen, J. Chen, X. Lin, Smart community: an internet of things application, IEEE Commun. Mag. 49 (11) (2011) 68–75.
[3] A. Altaf, H. Abbas, F. Iqbal, A. Derhab, Trust models of internet of smart things: A survey, open issues, and future directions, J. Netw. Comput. Appl. 137 (2019) 93–111.
[4] T. Wang, G. Zhang, M.Z.A. Bhuiyan, A. Liu, W. Jia, M. Xie, A novel trust mechanism based on fog computing in sensor–cloud system, Future Gener. Comput. Syst..
[5] A.M. Shabut, K.P. Dahal, S.K. Bista, I.U. Awan, Recommendation based trust model with an effective defence scheme for MANETs, IEEE Trans. Mob. Comput. 14 (10) (2014) 2101–2115.
[6] R. Chen, J. Guo, F. Bao, Trust management for SOA-based IoT and its application to service composition, IEEE Trans. Serv. Comput. 9 (3) (2014) 482–495.
[7] M.D. Alshehri, F.K. Hussain, O.K. Hussain, Clustering-driven intelligent trust management methodology for the internet of things (CITM-IoT), Mob. Netw. Appl. 23 (3) (2018) 419–431.
[8] Z. Gao, W. Zhao, C. Xia, K. Xiao, Z. Mo, Q. Wang, Y. Yang, A credible and lightweight multidimensional trust evaluation mechanism for service-oriented IoT edge computing environment, in: 2019 IEEE International Congress on Internet of Things (ICIOT), IEEE, 2019, pp. 156–164.
[9] R. Chen, F. Bao, J. Guo, Trust-based service management for social internet of things systems, IEEE Trans. Dependable Secure Comput. 13 (6) (2015) 684–696.
[10] H. Hellaoui, A. Bouabdallah, M. Koudil, TAS-IoT: Trust-based Adaptive Security in the IoT, in: 2016 IEEE 41st Conference on Local Computer Networks (LCN), IEEE, 2016, pp. 599–602.
[11] F.E. Grubbs, Procedures for detecting outlying observations in samples, Technometrics 11 (1) (1969) 1–21.
[12] J. Laurikkala, M. Juhola, E. Kentala, N. Lavrac, S. Miksch, B. Kavsek, Informal identification of outliers in medical data, in: Fifth International Workshop on Intelligent Data Analysis in Medicine and Pharmacology, Vol. 1, Citeseer, 2000, pp. 20–24.
[13] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation forest, in: 2008 Eighth IEEE International Conference on Data Mining, IEEE, 2008, pp. 413–422.
[14] M.M. Breunig, H.-P. Kriegel, R.T. Ng, J. Sander, LOF: identifying density-based local outliers, in: Proceedings of the 2000 ACM SIGMOD international conference on Management of data, 2000, pp. 93–104.
[15] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in: Kdd, Vol. 96, 1996, pp. 226–231.
[16] M.A. Wong, J. Hartigan, Algorithm as 136: A k-means clustering algorithm, J. R. Stat. Soc. Ser. C Appl. Stat. 28 (1) (1979) 100–108.
[17] J. Guo, R. Chen, J.J. Tsai, A survey of trust computation models for service management in internet of things systems, Comput. Commun. 97 (2017) 1–14.
[18] D. Chen, G. Chang, D. Sun, J. Li, J. Jia, X. Wang, TRM-IoT: A trust management model based on fuzzy reputation for internet of things, Comput. Sci. Inf. Syst. 8 (4) (2011) 1207–1228.
[19] M. Nitti, R. Girau, L. Atzori, Trustworthiness management in the social internet of things, IEEE Trans. Knowl. Data Eng. 26 (5) (2013) 1253–1266.
[20] Y.B. Saied, A. Olivereau, D. Zeghlache, M. Laurent, Trust management system design for the Internet of Things: A context-aware and multi-service approach, Comput. Secur. 39 (2013) 351–365.
[21] H. Xia, F. Xiao, S.-s. Zhang, C.-q. Hu, X.-z. Cheng, Trustworthiness inference framework in the social Internet of Things: A context-aware approach, in: IEEE Conference on Computer Communications (INFOCOM), IEEE, 2019, pp. 838–846.
[22] Z. Lin, L. Dong, Clarifying trust in social internet of things, IEEE Trans. Knowl. Data Eng. 30 (2) (2017) 234–248.
[23] H. Xia, B. Li, S. Zhang, S. Wang, X. Cheng, A novel recommendation-based trust inference model for MANETs, in: International Conference on Wireless Algorithms, Systems, and Applications, Springer, 2018, pp. 893–906.
[24] U. Jayasinghe, G.M. Lee, T.-W. Um, Q. Shi, Machine learning based trust computational model for IoT services, IEEE Trans. Sustain. Comput. 4 (1) (2018) 39–52.
[25] J. Caminha, A. Perkusich, M. Perkusich, A smart trust management method to detect on-off attacks in the internet of things, Secur. Commun. Netw. (2018).
[26] M. Bahutair, A. Bouguettaya, A.G. Neiat, Adaptive trust: Usage-based trust in crowdsourced IoT services, in: 2019 IEEE International Conference on Web Services (ICWS), IEEE, 2019, pp. 172–179.
[27] C. Boudagdigue, A. Benslimane, A. Kobbane, M. Elmachkour, A distributed advanced analytical trust model for IoT, in: 2018 IEEE International Conference on Communications (ICC), IEEE, 2018, pp. 1–6.
[28] A. Josang, R. Ismail, The beta reputation system, in: Proceedings of the 15th Bled Electronic Commerce Conference, Vol. 5, 2002, pp. 2502–2511.
[29] H. Simaremare, A. Syarif, A. Abouaissa, R.F. Sari, P. Lorenz, Performance comparison of modified AODV in reference point group mobility and random waypoint mobility models, in: 2013 IEEE International Conference on Communications (ICC), IEEE, 2013, pp. 3542–3546.

Guozhu Chen: received his B.S. degree in Software Engineering from Anhui University. Currently a Master student in the School of Computer Science and Technology at University of Science and Technology of China. His research interests mainly include Internet of Things and Trust Model.

Fanping Zeng: is an Associate Professor of the School of Computer Science and Technology, University of Science and Technology of China. He graduated from Harbin Institute of Technology with a bachelor's degree and received Ph.D. degree from the University of Science and Technology of China in 2009. His main research interests include software testing, network and system security. Recently, he mainly focuses on the Internet of things, studies the collaborative optimization of edge and end resources, security analysis, security testing and evaluation.

Jian Zhang: received the B.Sc. degree from the University of Science and Technology, China, in 1988, and the Ph.D. degree from the Institute of Software, Chinese Academy of Sciences, China, in 1994. He is a Research Professor with the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, and also a Professor with the University of Chinese Academy of Sciences. His current research interests include software testing, program analysis, automated reasoning and constraint solving.

Tingting Lu: received his Bachelor of Management degree in Information Management and Information System from Anhui Medical University. Currently a Master student in the School of Cyber Science and Engineering at University of Science and Technology of China. His research interests mainly include Edge Computing and Task Scheduling.

Jingfei Shen: received his B.S. degree in Software Engineering from Zhengzhou University. Currently a Master student in the School of Computer Science and Technology at University of Science and Technology of China. His research interests mainly focus on Intrusion Detection System.

Wenjuan Shu: received the B.S. degree in computer science from Anhui Normal University, China, in 2018. She is currently working toward the M.S. degree in the School of Computer Science and Technology at University of Science and Technology of China. Her research interests include cloud computing, deep learning, resource prediction and resource optimization.