Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2015, Article ID 107921, 14 pages
http://dx.doi.org/10.1155/2015/107921

Research Article

A Decision-Aided Situation Awareness Mechanism Based on Multiscale Dynamic Trust

Fangwei Li, Yifang Nie, Jiang Zhu, Haibo Zhang, and Fan Liu

Chongqing Key Lab of Mobile Communications Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China

Correspondence should be addressed to Yifang Nie; nyf1990@yeah.net

Received 25 December 2014; Revised 8 March 2015; Accepted 15 March 2015

Academic Editor: Sergio Toral

Attacks are always seeking ways of exploiting any existing weakness in wireless network. The purpose of security situation awareness is to recognize, analyze, forecast, and handle the misbehaviors, assisting network management. Nevertheless, the appearance of ubiquitous network and the network convergence technology has made a challenge to realize network security and adaptation. Aiming at these research problems, this paper proposes a decision-aided situation awareness mechanism based on multiscale dynamic trust from the perspective of time and space, which can recognize misbehaviors and regard social network as the research object. We build trust and satisfaction based on Ebbinghaus forgetting regular and spatial correlations. This mechanism carries out decision making assessment through trust authenticity test, logicality test, and feedback parameters. In addition, load balance is used to avoid resource congestion. Simulation analysis demonstrates that compared with other trust mechanisms, this mechanism proposed in this paper can recognize and handle entity attacks more effectively, which is relatively eclectic and realistic in aspect of trust mensuration.

1. Introduction

Security situation awareness [1], including security elements detection, perception, prediction, evaluation, visualization, and management, can sense security information, extracting and analyzing security essential elements. It can also find out the existent threats and attacks [2], to cope with the characteristics of dynamics, openness, and uncertainty. Simultaneously, that also contains the steps of decision support, risk assessment, and situation visualization. Security situation awareness is increasingly becoming popular with the appearance of network convergence technology. Those network entities of situation awareness mechanism, which can flexibly and intelligently respond to the environment change, can be regarded as interactive entities existing in multiagent system, Peer to Peer Network named P2P, Wireless Sensor Network named WSN, and Mobile Ad hoc Network named MANET. Nevertheless, attacks and existing weaknesses also have made a challenge for researchers to effect network security and adaptation with the dynamic change of location, permission, role relationship, and information acquisition capability.

An effective method to minimize the threats is to evaluate the trusts of the interactive entities. As the main security element in security situation awareness mechanism [3–5], trust has long played a critical role in trust model or mechanism that can suppress misbehaviors effectively in network security area. Whereas, if mechanisms can process and manage entities dynamically, as the network environment changes, they will evolve to situation awareness mechanisms based on dynamic trust [6–8]. The situation awareness mechanism based on dynamic trust endows the entities with the capabilities of trust acquisition and perception [9] to assist in evaluating performance [10] and process real-time decision [11], against the uncertainty, transitivity, and time decay. This is the principal difference between security situation awareness mechanism and classical network security researches, which just uses access control, authentication, and firewall, not forming a real-time decision system.

At present, there are some aspects needing to be further improved in most of the existent trust mechanisms [12]. (i) In time scale, the weights of historical trusts should conform to the regular that the more recent the trusts are,

the higher their weights are. Furthermore, in space scale, during trust computation, many mechanisms allocate fixed weights to indirect trusts and direct trusts, not changing with the working condition dynamically. Additionally, when taking a third party recommender into consideration, these mechanisms have neglected whether it has been captured or not, thus provoking hidden trouble. (ii) Most of them have considered how to obtain and compute trust without decision support and risk assessment. (iii) They have held uniform trust standard about the whole system, however, independent trust standards are not allowed to exist in individuals, which cannot meet the requirements of independent individuals. (iv) When confronted with some anomalous attacks or behaviors, emergency strategy is invalid. (v) Few mechanisms have adopted the load balance strategy, but preferring to select the most authentic entity as the service object, which can bring about excessive load and resource congestion. (vi) If the malicious and eliminated entities were to attempt to regain system, time interval should have been set. (vii) In case of the fact that entities are honest to a special service, but cheat others, different services should have different trusts in a system; however, most of mechanisms have only one type of trust.

Aiming at these research problems, this paper proposes a decision-aided situation awareness mechanism based on multiscale dynamic trust named DynamicTrust in wireless network, which can visualize abrupt changes about misbehaviors. It takes social network as the research object. DynamicTrust computes integrated trust and exerts fuzzy theory to descript entity relationship and trust, consulting such parameters including satisfaction, indirect trust, direct trust, historical trust, and individual relevance. Building entities trusts based on different service types can deal with the misbehaviors, which are honest to some service types, but cheat others. The usability, capability, vulnerability, trust authenticity test, and trust logicality test have been used to feed back to trust computation in period of decision making and assessment. The utilization of load balance avoids resource congestion. If we newly set environmental parameters, this mechanism can also act on P2P, Ad hoc, and WSN, etc.

For example, if network nodes in WSN or Ad hoc should have adequate sources and store capacities, we could apply the proposed system in WSN or Ad hoc, with definition parameters in system, such as satisfaction and trust. The network nodes in WSN or Ad hoc will become the entities of DynamicTrust and also will collect trust information in process of data interaction with other nodes, computing the parameters required by the proposed system. The parameter collection and computation can be descripted as Section 3.

The remaining part of the paper is organized as follows. Section 2 mainly reviews the related works. In Section 3, we introduce our model about situation awareness mechanism based on multiscale dynamic trust and also present how our situation awareness mechanism has carried out. In Section 4, we research how our model resists the confronted threats and compare our mechanism with others by comparatively studying simulations and performances. Finally, we conclude the paper in Section 5.

2. Related Works

These security trust models or mechanisms based on trust [6–16] contain the steps of information acquisition, trust computation, entity selection, and behavior bonus-penalty. They have considered direct trust and indirect trust, but minority of them has added feedback empiric values in trust computation process, and most of them adopt recommended trust from a third party without checking its usability. In fact, they all cannot shape a real-time security situation awareness system, not visualizing the attacks or evils and also not making management strategy for misbehaviors.

These existing mechanisms based on trust, for such networks including multiagent system, P2P, MANET and WSN, have utilized Bayesian [12], expertise [13], fuzzy theory [14], evidence theory [15], bioinspired [16] and social network graph theory [11] to build and describe dynamic trust relationship. The prestandardization of trust and reputation models in [17] concludes that the research based on fuzzy theory is relatively more than other researches. The existing researches related to network security mechanisms based on trust can be roughly divided into 4 stages from the perspective of processing sequence [18]. Originally, the first stage is information acquisition. In this stage, the mechanism collects information, such as entity parameters, empirical values, and service times. Then, the second stage is trust computation. The transaction weights, direct trust, indirect trust, and empirical values are utilized to compute trust and other parameters. Moreover, the third stage is entity selection. Several rules and methods service to entity selection and decision support. The last stage is bonus-penalty. Misbehaviors are doomed to punished, but honest ones need to be rewarded. The goal of constructing the situation awareness mechanism based on dynamic trust is to provide more reliable service, make full use of system resources, and achieve profit maximization.

PeerTrust in [4] is a peer to peer communication model, whose trust lies on feedback satisfaction, service number, feedback trust, community factor, and transaction factor. PeerTrust can recognize misbehaviors and also can distinguish false information and honest information. Whereas, all the parameters of service providers will be used to compute trust, without considering historical trust and time relation factor, which is the weakness of PeerTrust. Moreover, no load balance strategy has been applied.

A dynamic trust computation model for secured communication in [7], named SecuredTrust, has computed trust depending on trust similarity, trust difference, feedback trust, and historical trust. It thinks over the time near-far effect and ponders behavior bonus-penalty. The load balancing has also been considered. However, only one historical trust value before recent trust has been adopted, but historical trust values on other moments are abandoned. After misbehaviors have been found out, no strategy has been taken.

Decision making matters in [8], named DecisionTrust, has referred usability as trust assessment parameter in whole model and also introduced 4 decision makings into trust model. Nevertheless, trust computation relies on direct trust,

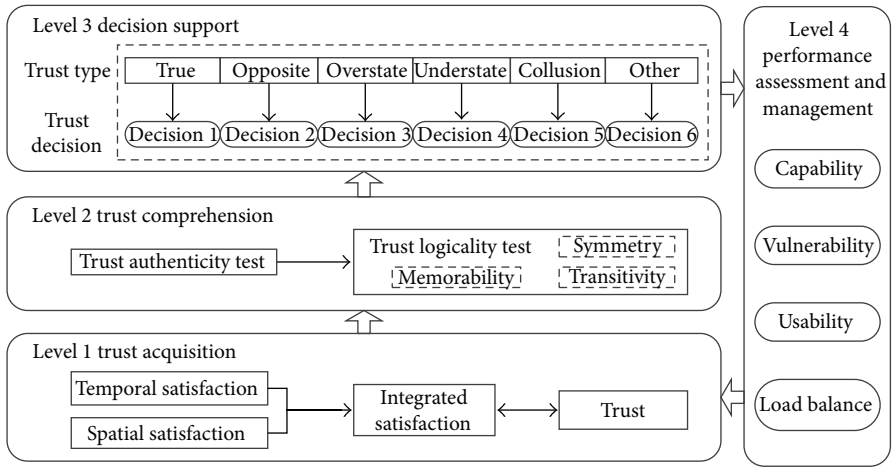


Figure 1: The decision-aided situation awareness model based on multiscale dynamic trust.

recommended trust, and environmental factor, without pondering on the time decay. In addition, the capability of each entity is a given and fixed constant.

The multistrategy trust evolution model in [10] has used fuzzy theory to compute the uncertainty and fuzziness of trust, solving the problem of only obtaining but never sharing information. The game evolvement method has been adopted during trust computation, but no excessive load strategy has been taken into account.

To some extent, the existing models or mechanisms can recognize misbehaviors and improve accuracy of mechanism to actualize security communication. During trust computation, historical trust has been used. However, they all cannot shape a real-time security situation awareness system, without making real-time strategies for attacks.

DynamicTrust proposed in this paper has built a decision-aided situation awareness mechanism based on dynamic trust in wireless network. From time perspective, DynamicTrust computes trust and satisfaction, referring to historical values and Ebbinghaus forgetting factor. From space perspective, after time perspective treatment, DynamicTrust will compute parameters based on social network model and spatial relationships among entities. Moreover, Trust authenticity test and logicality test are used to detect the reliability of entities. Ultimately, the results of decision making and assessment can feed back to trust computation.

3. The Decision-Aided Situation Awareness Mechanism Based on Multiscale Dynamic Trust

3.1. System Model. The intent of our mechanism is to provide an effective dynamic situation awareness mechanism to resist and minimize the threats. The model of DynamicTrust proposed in this paper can be divided into 4 levels, trust acquisition level, trust comprehension level, decision support level, and performance assessment and management level. In first level, DynamicTrust obtains trust based on indirect satisfaction and direct satisfaction from the scales of time and space. In second level, DynamicTrust uses trust authenticity test and trust logicality test to see whether the trusts provided by entities are reliable. The trust logicality contains transitivity, symmetry, and memorability. In third level, after these two trust tests, all trusts of entities will be separated into 6 types, such as true type, opposite type, overstated type, understated type, collusion type, and other type. For each type, this mechanism has made a homologous strategy. In fourth level, DynamicTrust will manage capability, vulnerability, usability, and loads of entities, providing feedback to other 3 levels and achieving the dynamic adaptation. The model of DynamicTrust is given in Figure 1.

3.2. Trust Acquisition. This section describes several definitions at first level of DynamicTrust. The range of satisfaction definition is from 0 to 1. If entity a is entirely satisfied with entity b, the satisfaction of entity a to entity b will be 1. Otherwise, if entity a is not satisfied with entity b, the satisfaction will be 0. If an entity is incompletely satisfied with another one, the satisfaction will be a value in the range of 0 to 1, fuzzily. Similarly, the ranges of vulnerability, capability and trust are similar. The parameters of entities not participating in services will keep their own values.

3.2.1. Satisfaction

Definition 1 (indirect satisfaction named **SAT**). Indirect satisfaction of entity $j \in I_k$ to entity $i \in I_k$ for service $S \in \mathbf{S}$ at time $t \in \boldsymbol{\tau}$ in nth state, $\mathrm{sat}^t_{n,s}(i, j)$, represents the adjacent degree between the service satisfaction of entity i to entity j and the expected satisfaction of entity i. We define indirect satisfaction as $\mathrm{sat}^t_{n,s}(i, j) \in$ **SAT,**

$$\mathrm{sat}^t_{n,s}(i, j) = \begin{cases} 0, & \text{dissatisfaction,} \\ r \in (0, 1), & \text{fuzzy,} \\ 1, & \text{satisfaction,} \end{cases} \tag{1}$$

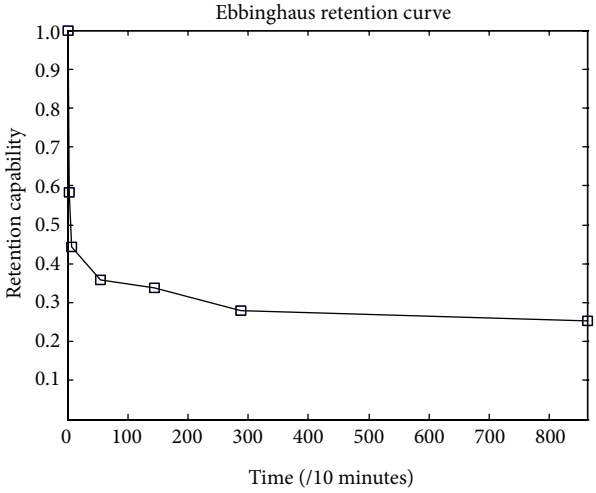


Figure 2: Ebbinghaus retention curve.

where $I_k \in \mathbf{I}$ represents kth community and $\mathbf{I}$ is the community gather of all communities. $\boldsymbol{\tau}$ is the discrete time set of all services, but $\mathbf{U}$ is the state set of all service entities. r obeys the distribution $N(\mu, 1)$ with the cumulative distribution function $f : dr/dx = 1/\sqrt{2\pi} \cdot \exp\{-(x-\mu)^2/2\}$, where μ is the expected value.

Definition 2 (direct satisfaction named $\overline{\mathbf{SAT}}$). Direct satisfaction of entity $i \in I_k$ for service $S \in \mathbf{S}$ at time $t \in \boldsymbol{\tau}$ in nth state can be expressed as $\overline{\text{sat}}^t_{n,S}(i) = \overline{\text{sat}}^t_{n,S}(i,i) = 1, \overline{\text{sat}}^t_{n,S}(i) \in \overline{\mathbf{SAT}}$.

Definition 3 (temporal satisfaction named **SAT**_tim). Temporal satisfaction of entity $j \in I_k$ to entity $i \in I_k$ for service $S \in \mathbf{S}$ at current time $t \in \boldsymbol{\tau}$ in nth state represents the geometric mean of all indirect satisfactions of entity $i \in I_k$ at current and previous time. We define it as $\text{sat\_tim}^t_{n,s}(i,j) \in \mathbf{SAT}\_\text{tim}$,

$$\text{sat\_tim}^t_{n,s}(i,j) = \frac{1}{N}\sum_{q=0}^{N} \omega_q \text{sat}^{t-q}_{n-q,s}(i,j), \tag{2}$$

where N represents the number of time windows, and the weight $\omega_q = f_q / \sum_{i=1}^{N} f_i$ is proportional to f_q, the Ebbinghaus retention function value at time $t-q$. We set $\omega_0 = 1 - \sum_{q=1}^{N} \omega_q$, $\text{sat}^0_{0,S}(i,j) = 0.5$, and suppose the Ebbinghaus retention function value at time t is 1. The Ebbinghaus retention curve [17] is like Figure 2.

Example 4. If N is 4, the weights $\omega_i = f_i / \sum_{q=1}^{N} f_q$, $i = 0, 1, 2, 3$ of different average service response intervals will have different values. The detailed weights are given in Table 1.

Definition 5 (correlated consistency named $\mathbf{R}$). Correlated consistency between entity $j \in I_k$ and entity $i \in I_k$ for service $S \in \mathbf{S}$ at time $t \in \boldsymbol{\tau}$ in nth state represents the satisfaction

Table 1: Time weights based on Ebbinghaus retention rate.

Average interval for each service (/minute)	ω_0	ω_1	ω_2	ω_3
Interval = 10	0.342	0.271	0.200	0.187
Interval = 20	0.394	0.230	0.202	0.174
Interval = 30	0.422	0.231	0.186	0.161

correlation between arbitrary two entities $i, j \in I_k$. It can be shown as $r_{ij}(S) \in \mathbf{R}$,

$$r_{ij}(S) = 1 - \frac{2}{(m_k - 1)\, m_k} \cdot \sum_{\substack{k1=1,k2=1 \\ k1 \le k2}}^{m_k} \Big| \text{sat\_tim}^t_{n,s}(i,k1) - \text{sat\_tim}^t_{n,s}(i,k2) + \text{sat\_tim}^t_{n,s}(j,k1) - \text{sat\_tim}^t_{n,s}(j,k2) \Big|, \tag{3}$$

where m_k is the number of kth community members. $r_{ij}(S)$, $i \le j$ ranges from 0 to 1 and holds the symmetric reflex, $r_{ij}(S) = r_{ji}(S)$.

Definition 6 (spatial satisfaction named **SAT**_spac). Spatial satisfaction of entity $j \in I_k$ to entity $i \in I_k$ for service $S \in \mathbf{S}$ at time $t \in \boldsymbol{\tau}$ in nth state represents the weighted mean of all temporal satisfactions of entity $i \in I_k$ at time $t \in \boldsymbol{\tau}$. We define it as $\text{sat\_spac}^t_{n,S}(i) \in \mathbf{SAT}\_\text{spac}$,

$$\text{sat\_spac}^t_{n,S}(i) = \frac{1}{\sum_{j=1, i\neq j}^{M_k} r_{ij}(S)} \sum_{\substack{j=1 \\ i\neq j}}^{M_k} r_{ij}(S)\, \text{sat\_tim}^t_{n,S}(i,j), \quad r_{ij}(S) > \hat{r}, \tag{4}$$

where M_k is the number of kth community members meeting the condition $r_{ij}(S) > \hat{r}$, $\text{usability}_j > 0.5$, usability_j is the usability of entity j, and $\hat{r}$ is the threshold of correlated consistency.

Example 7. With the known condition that there is only one community $\mathbf{E} = \{e1, e2, e3\}$ with $N = 4$ and interval = 10 for only one service type $A \in \mathbf{S}$, the time weights based on Ebbinghaus retention rate will be $\omega_0 = 0.342$, $\omega_1 = 0.271$, $\omega_2 = 0.200$, and $\omega_3 = 0.187$. Without considering

the existence of attacks, we suppose the indirect satisfactions at time $t, t-1, t-2, t-3$ are, respectively, as follows:

$$\begin{aligned}
\mathbf{SAT}_{n,A}^{t} &= \begin{pmatrix} 0.800 & 0.800 & 0.800 \\ 0.500 & 0.500 & 0.200 \\ 0.200 & 0.200 & 0.200 \end{pmatrix}, \\
\mathbf{SAT}_{n-1,A}^{t-1} &= \begin{pmatrix} 0.600 & 0.600 & 0.600 \\ 0.500 & 0.500 & 0.800 \\ 0.400 & 0.400 & 0.400 \end{pmatrix}, \\
\mathbf{SAT}_{n-2,A}^{t-2} &= \begin{pmatrix} 0.400 & 0.400 & 0.400 \\ 0.500 & 0.500 & 1.000 \\ 0.600 & 0.600 & 0.600 \end{pmatrix}, \\
\mathbf{SAT}_{n-3,A}^{t-3} &= \begin{pmatrix} 0.200 & 0.200 & 0.200 \\ 0.500 & 0.500 & 0.800 \\ 0.800 & 0.800 & 0.800 \end{pmatrix}.
\end{aligned} \tag{5}$$

According to the indirect satisfactions, we can get temporal satisfactions

$$\mathbf{SAT}\_\mathrm{tim}_{n,A}^{t} = \begin{pmatrix} 0.554 & 0.554 & 0.554 \\ 0.500 & 0.500 & 0.635 \\ 0.446 & 0.446 & 0.446 \end{pmatrix}. \tag{6}$$

Furthermore, correlated consistency and spatial satisfaction are as follows:

$$\begin{aligned}
\mathbf{R}(A) &= \begin{pmatrix} 1.000 & 0.910 & 1.000 \\ 0.910 & 1.000 & 0.910 \\ 1.000 & 0.910 & 1.000 \end{pmatrix}, \\
\mathbf{SAT}_\mathrm{spac}_{n,A}^{t} &= \begin{pmatrix} 0.955 \\ 0.910 \\ 0.955 \end{pmatrix}.
\end{aligned} \tag{7}$$

Definition 8 (integrated satisfaction named $\mathbf{SAT}'$). Integrated satisfaction of entity $i \in I_k$ at time $t \in \boldsymbol{\tau}$ in nth state represents the weighted mean of spatial satisfaction and direct satisfaction, which can be expressed as $\mathrm{sat}_n^{t'}(i) \in \mathbf{SAT}'$,

$$\mathrm{sat}_n^{t'}(i) = \beta \cdot \mathrm{sat}\_\mathrm{spac}_{n,S}^{t}(i) + (1-\beta)\,\overline{\mathrm{sat}}_{n,S}^{t}(i), \tag{8}$$

where $\beta \in [0,1]$ is the indirect satisfaction factor.

3.2.2. Trust Definition

Definition 9 (entity trust named $\mathbf{T}$). Entity trust of entity $i \in I_k$ for service $S \in \mathbf{S}$ at time $t \in \boldsymbol{\tau}$ in nth state represents the weighted mean of spatial satisfactions of all entities in I_k. We describe it as $T_n^i \in \mathbf{T}$, $T_n^i \in [0,1]$,

$$T_n^i = \frac{1}{\sum_{j=1}^{M_k} r_{ij}} \sum_{j=1}^{M_k} r_{ij}\, \mathrm{sat}\_\mathrm{spac}_{n,S}^{t}(i), \quad i = 1, \ldots, m_k, \tag{9}$$

where m_k is the number of kth community I_k members.

Definition 10 (local trust named $\mathbf{TI}$). Local trust of community I_k in nth state represents the trust expectation of all entities in community I_k. It also ranges from 0 to 1, described as $\mathrm{TI}_n^i \in \mathbf{TI}$,

$$\mathrm{TI}_n^i = \frac{1}{m_k} \sum_{d=1}^{m_k} T_n^d, \quad i = 1, \ldots, K, \tag{10}$$

where K is the total number of communities, and TI_n^i is the local trust of ith community.

Definition 11 (globe trust named TA). Globe trust in nth state represents the whole trust of this system, which can be described as

$$\mathrm{TA} = \frac{1}{K} \sum_{i=1}^{K} \mathrm{TI}_n^i. \tag{11}$$

In Example 7, the trust of community $\mathbf{E}$ is $\mathbf{T} = (0.970 \;\; 0.938 \;\; 0.970)^T$, and the local trust is $\mathbf{TI} = (\mathrm{TA}) = (0.959)$.

3.3. Trust Comprehension. Trust comprehension is at second level of DynamicTrust. The main purpose of this section is to validate the authenticity and logicality of entity trust. There are two trust tests, trust authenticity test and trust logicality test.

3.3.1. Trust Authenticity Test. The trust authenticity test is to detect the authenticity of entity. We define the object collection as $\mathbf{G} \subseteq I_k$ in authenticity test and quantize all entity trusts into 10 grades by spacing 0.1. For arbitrary entity p, according to the difference between the real trust and the mean trust of entity $p \in \mathbf{G}$ at nth state, we can get the trust measure function

$$f(\mathbf{G}) = \frac{1}{\mathrm{size}(\mathbf{G})} \left(\frac{1}{\mathrm{size}(\mathbf{G})} \sum_{p \in \mathbf{G}} \sum_{i \in \mathbf{G}} T_n^{pi} - \sum_{p \in \mathbf{G}} T_n^{p} \right), \tag{12}$$

where $\mathrm{size}(\mathbf{G})$ is the number of the object set $\mathbf{G}$ elements in trust authenticity test. T_n^{pi} is the trust of entity p said by entity i, and T_n^p is the real trust of entity p.

If the test result is $f(\mathbf{G}) = 0$, DynamicTrust will consider all entities authentic in set $\mathbf{G}$. However, if the consequence is $f(\mathbf{G}) \neq 0$, we consider not all entities authentic in $\mathbf{G}$.

During the treat processing, the under test set $\mathbf{D}$ collects the entities being put out of $\mathbf{G}$, $\mathbf{G} \cup \mathbf{D} = I_k$. If we regard I_k as the test object, the processing can be described as follows.

Step 1. Initially, $\mathbf{G} = I_k$, if the function value is $f(\mathbf{G}) = 0$, we can see that all entities in $\mathbf{G} = I_k$ are authentic and then drop out of the trust authenticity test. Otherwise, $f(\mathbf{G}) \neq 0$, this mechanism will get into next step.

Step 2. If the trusts of entities are equal to 0.1, we will put them out of set $\mathbf{G}$ and then put set $\mathbf{D}$ under test. After that, we will also compute trust measure function value again. If the value is $f(\mathbf{G}) = 0$, we will drop out of the trust authenticity

Input: T_n^p All trusts of entities and $T_n^{pi}, \forall i, p \in I_k$ all mutual trusts of entities in group I_k
Output: the collection **D** in which the entities may be untruthful
(1) let $\mathbf{G} = I_k$
(2) **if** $f(\mathbf{G}) = 0$
(3) all the entities in group I_k is true
(4) **else**
(5) **while** $f(\mathbf{G}) \neq 0$
(6) let $num = 0$
(7) **for** each $p \in I_k$ **do**
(8) **if** the trust of entity p is equal to $0.1num$
(9) let p get out of collection **G** and get into new collection **D**
(10) **end if**
(11) **if** $f(\mathbf{G})$ is equal to 0
(12) break
(13) **end if**
(14) **end for**
(15) **end while**
(16) **end if**

Algorithm 1: Truth detection of entities in group $I_k(T_n^p, T_n^{pi}, \forall i, p \in I_k)$.

test. In contrast, if the trusts of entities are equal to 0.2, we will put them out of set **G** and then also put **D** under test set, with repetitive operation and computation. According to the sequence of trusts from low to high, we will repeat the operation sequentially, until we get $f(\mathbf{G}) = 0$ or the set **G** is empty.

The algorithm in trust authenticity test is given in Algorithm 1.

3.3.2. Trust Logicality Test

Definition 12 (trust difference named $\boldsymbol{\sigma}$). The trust difference between the real trust of entity h and the trust of entity h said by entity q at nth state, which can be described as $\sigma_{hq} = |T_n^{hq} - T_n^h|$, $\sigma_{hq} \in \boldsymbol{\sigma}$. If entity q trusts entity h, we will set $\sigma_{hq} = 0$.

The under test set **D** is the object collection in trust logicality test. After trust authentic test, if under set **D** is not empty, we will carry out trust logical test for entities in **D**. There are three aspects in trust logicality test, such as transitivity test, symmetric consistency test, and memorability test.

(a) Symmetric Consistency Test. The trust of entity h said by entity q should be equal to that of entity h said by entity u, $T_n^{hq} = T_n^{hu}$, or else we will take $q, h, u \in \mathbf{D}$ as unbelievable entities which will be doomed to enter next stage decision support.

(b) Transitivity Test. In view of the fact that multilevel transmission may bring up the expending of attacks, we set transitivity with only one level. For example, if entity q trusts entity h and entity h trusts entity p and entity p trusts entity y, we can know entity q trusts entity p but do not know whether entity q trusts entity y.

Theorem 13. *With the known condition $\sigma_{hq} = \sigma_{ph} = 0$, if p, q, h are credible, we can get $\sigma_{pq} = 0$.*

Proof. From $\sigma_{hq} = |T_n^{hq} - T_n^h| = 0$ and $\sigma_{ph} = |T_n^{ph} - T_n^p| = 0$, we can see $\sigma_{pq} = |T_n^{pq} - T_n^p| = |T_n^{ph} - T_n^p| = 0$. In other words, if q trusts h and h trusts p, p will trust q. However, if there are $\sigma_{hq} = \sigma_{ph} = 0$ and $\sigma_{pq} \neq 0$, p, q, h will be considered as incredible, which is opposite to the known condition. Therefore, $\sigma_{pq} = 0$. □

(c) Memorability Test. Compared with historical entity trust, if current entity trust is higher or identical, we will think the entity is credible. However, if current entity trust is lower, we will consider it as an unbelievable entity.

Those entities not passing the trust authenticity test or trust logicality test, which will be regarded as unbelievable entities, will come under decision set $\boldsymbol{\xi}$ and enter next stage, decision support.

3.4. Decision Support. Decision support exists at third level of DynamicTrust. After trust acquisition and trust comprehension, in this section, DynamicTrust will determine the trust type and then make decision for different trust types.

3.4.1. Trust Type Decision

Definition 14 (trust deviation named $\boldsymbol{\Delta}$). The trust deviation between the real trust and the mean trust of entity i in $\boldsymbol{\xi}$ at nth state represents the mean difference between the real trust of entity $i \in \boldsymbol{\xi}$ and the trusts of entity i said by other entities in $\boldsymbol{\xi}$, which can be described as $\Delta(i) \in \boldsymbol{\Delta}$,

$$\Delta(i) = \frac{1}{M_k - 1} \sum_{p=1, p \neq i}^{M_k} \left(\widehat{T}_n^{ip} - T_n^i \right), \tag{13}$$

where $\widehat{T}_n^{ip}$ is the actual trust of entity i said by entity $p \in \boldsymbol{\xi}$ and T_n^i is the real trust of entity i.

Taking the possible contingencies into consideration, we classify trusts into 6 types, such as true type, opposite type, overstated type, understated type, collusion type, and other type.

True Type. If there is $\Delta(i) = 0$, entity i will be considered as honest. We take such entity into true type.

Opposite Type. If there are $\Delta(i) \neq 0$ and $T_n^i + \widehat{T}_n^i = 1$, trust of entity i will be considered as opposite, where $\widehat{T}_n^i$ is the mean trust of entity i agreed by all entities in $\boldsymbol{\xi}$. We put entity i into opposite type.

Overstated Type. If there is $\Delta(i) > 0$, trust of entity i will be overstated. We take entity i into overstated type.

Understated Type. If there is $\Delta(i) < 0$, trust of entity i will be understated. We put entity i as understated type.

Collusion Type. If there are more than 3 entities whose trusts are identically opposite or overstated or understated in a same community, we will take such entities into collusion entities.

Other Type. If entities do not belong to the former 5 types, we will collect them into other types.

3.4.2. Trust Decision Support

Definition 15 (deviation factor named $\boldsymbol{\rho}$). The deviation factor of the trust deviation to the real trust of entity $i \in \boldsymbol{\xi}$ in nth state represents the absolute value of the ratio of $\Delta(i)$ to T_n^i, which can be described as $\rho_i \in \boldsymbol{\rho}$,

$$\rho_i = \left| \frac{\Delta(i)}{T_n^i} \right|, \quad 0 \leq \rho_i \leq 1. \tag{14}$$

Decision 1. For entity $i \in \boldsymbol{\xi}$ of true type, we will adapt its trust as $\widetilde{T}_n^i = T_n^i$.

Decision 2. For entity $i \in \boldsymbol{\xi}$ of opposite type, we will modify its trust as $\widetilde{T}_n^i = T_n^i - \rho_i$.

Decision 3. For entity $i \in \boldsymbol{\xi}$ of overstated type, we will adjust its trust as $\widetilde{T}_n^i = (1 - \rho_i) T_n^i$, $0 \leq \rho_i \leq 1$.

Decision 4. For entity $i \in \boldsymbol{\xi}$ of understated type, we will modify its trust as $\widetilde{T}_n^i = (1 - \rho_i) T_n^i$, $0 \leq \rho_i \leq 1$.

Decision 5. For entities $i \in \boldsymbol{\xi}$ of collusion type, we will not only adapt trust based on its corresponding trust type, but will also reduce trust to $\widetilde{T}_n^i = \widetilde{T}_n^i - \rho_i$.

Decision 6. To entity $i \in \boldsymbol{\xi}$ of other types, we allocate a value in the range of $[0, 1]$ to the trust, randomly.

This system will filtrate out entities with trust not in the range of $[0.2, 1]$. After a period of time Δt, they can be allowed to reenter the system. This mechanism will also redistribute initial parameters for all entities, after executing 50 services.

3.5. Aided Performance Assessment. This section is at fourth level of DynamicTrust. To cope with the condition that an entity with high trust is compromised at current service, but we also look on it as reliable entity, we introduce usability into this mechanism, to reduce the probability of attack success. We take integrated trust as the capability.

Definition 16 (entity capability named **CAP**). Entity capability of entity $i \in I_k$ at time $t \in \boldsymbol{\tau}$ in nth state represents the integrated satisfaction of entity $i \in I_k$, which can be expressed as $\text{cap}_n^t(i) \in \mathbf{CAP}$,

$$\text{cap}_n^t(i) = \text{sat}_n^{t'}(i). \tag{15}$$

If a new entity a enters the system, system will initialize its capability as $\text{cap}_0^0(a) = 0.5$.

Definition 17 (community capability named **CAPI**). Community capability of $I_k \subset \mathbf{I}$ at time $t \in \boldsymbol{\tau}$ in nth state represents the expectation of all capabilities of entities in I_k, which can be expressed as $\text{cap}_n^t I_k \in \mathbf{CAPI}$,

$$\text{cap}_n^t I_k = \frac{1}{m_k} \sum_{i=1}^{m_k} \text{cap}_n^t(i). \tag{16}$$

Definition 18 (entity vulnerability named **V**). Entity vulnerability of entity $i \in I_k$ at time $t \in \boldsymbol{\tau}$ in nth state represents the recuperative capability, which can be expressed as $\text{vul}_n^t(i) \in \mathbf{V}$,

$$\text{vul}_n^t(i) = 1 - \text{cap}_n^t(i). \tag{17}$$

Definition 19 (entity relative usability named **usability**). Entity relative usability of entity $i \in I_k$ at time $t \in \boldsymbol{\tau}$ in nth state represents the ratio of entity capability to the maximum capability in I_k at current state, which can be expressed as $\text{usability}(i, t) \in \mathbf{usability}$, $\mathbf{U} \times \mathbf{CAP} \times \boldsymbol{\tau} \rightarrow \mathbf{usability}$,

$$\text{usability}(i, t) = \frac{\text{cap}_n^t(i)}{\max \text{cap}_n^t(i)} \times U_n^t(i), \quad i \in I_k, \tag{18}$$

where $U_n^t(i) \in \mathbf{U}$, $U_n^t(i) \in \{0, 1\}$ is the state of entity i at time $t \in \boldsymbol{\tau}$ in nth state. $U_n^t(i) = 0$ represents the entity i which is unavailable, but $U_n^t(i) = 1$ represents entity i which is available. If a new entity a enters the system, system will initialize its usability as $\text{usability}(a, 0) = 1$.

Supposing that the capability of entity i is $\text{cap}_n^t(i) = 0.8$ and the maximum capability is $\max_{i \in I_k} \text{cap}_n^t(i) = 1$, if the entity i is in the system at time t, $U_n^t(i) = 1$, its usability can be expressed as $\text{usability}(i, t) = 0.8$. In contrast, if it is not in the system, its usability will be $\text{usability}(i, t) = 0$.

Example 20. In Example 7, with the known parameters, $\mathbf{SAT}\_\text{spac}_{n,A}^t = (0.955, 0.910, 0.955)^T$, $\overline{\text{sat}}_{n,S}^t(1) = 0.554$,

Table 2: Service parameter table.

Service type	Significant degree	The parameters of entities providing service			
		Logicality	Trust	Usability	Remark[a]
Type A	Most significant	Satisfaction	$\widetilde{T}_n^i > \theta_1$	Usability > 0.8	Selecting the entity providing service for type B
Type B	Special significant	Dissatisfaction	$\widetilde{T}_n^i > \theta_2$	Usability > 0.6	Selecting the entity providing service for type C
Type C	Significant	Dissatisfaction	$\widetilde{T}_n^i > \theta_3$	Usability > 0.5	Selecting the entity providing service for type D
Type D	General	Dissatisfaction	$\widetilde{T}_n^i > \theta_4$	Usability > 0.4	Selecting service entity, randomly

[a]What should we select to provide service, if there is no entity meeting the parameter requirement?

$\overline{\text{sat}}_{n,S}^t(2) = 0.500$, $\overline{\text{sat}}_{n,S}^t(3) = 0.446$, and $\beta = 0.7$, we can obtain the entity capability $\mathbf{CAP}_n^t = (0.835, 0.787, 0.802)^T$, the entity vulnerability $\mathbf{V}_n^t = (0.165, 0.213, 0.198)^T$, and the entity relative usability $\mathbf{usability}_n^t = (1.000, 0.943, 0.960)^T$.

3.6. Load Balance Related to Select Service Object. This section introduces a load balance method at fourth level of DynamicTrust. According to the significance of service, services have been separated into 4 types, such as type A, type B, type C, and type D, thus constituting service set $\mathbf{S} = \{A, B, C, D\}$. Different service objects are used for different service type. The trust threshold set can be $\mathbf{\Theta} = \{\theta_1, \theta_2, \theta_3, \theta_4\}$. The service parameters are as Table 2.

In load balance, the related steps are listed as follows.

Step 1. After entity p makes a request for service S, all entities responding to the request compose the response set H. We define the response entity finally providing service as entity q. $\forall i \in H$, if entity i passes the trust logicality test and meets the condition usability$(i, t) > \vartheta_u$, $\widetilde{T}_n^i > \vartheta_T$, we will bring it into the credible set P. However, if entity i only meets the condition usability$(i, t) > \vartheta_u$, $\widetilde{T}_n^i > \vartheta_T$, but does not satisfy the trust logicality, it will be put into the candidate set Q. All the entities in H but not in Q or P will be put into the second choice set R.

Step 2. If set P is nonempty, $\forall i \in P$, we will compute the load of entity i, $L(i)$. Then, we will find out g entities with smallest loads. Finally, we randomly select an entity as entity q from g entities.

Step 3. If credible set P is empty but the candidate set Q is nonempty, $\forall i \in Q$, we will compute the load of entity i, $L(i)$, selecting the entity with smallest load as entity q.

Step 4. If sets P and Q are empty, we will select response entity from $R = H$. $\forall i \in R$, if entity i meets the condition $\widetilde{T}_n^i > 0.5$, we will compute the load of entity i, $L(i)$, and select the entity with smallest load as entity q. Otherwise, we will randomly select response entity q from H.

Assuming that g is 1, the load balance is given as Algorithm 2.

4. Simulation Analysis and Performance Comparison

In this section, we simulate the decision-aided situation awareness mechanism based on multiscale dynamic trust relaying on the above theoretical frame to evaluate the mechanism performance and prove the applicability and effectivity. We have fulfilled our simulation at MATLAB 7.1 simulation platform in Windows operating system with Intel Core (TM) Duo 2.66 GHz CPU, 2 GB Memory. There are M entities and the total number of service times is C in our simulation. The length of time window is N. We set M as 100, classifying all services into 4 communities I_1, I_2, I_3, I_4, with 27, 25, 22, 26 entities, respectively. We can get $\mathbf{I} = I_1 \cup I_2 \cup I_3 \cup I_4$. The simulation regards community I_3 as the research object. The parameter setting is given in Table 3.

In this part, we compare our DynamicTrust with SecuredTrust [7], PeerTrust [4] and DecisionTrust [8], from 4 aspects, including sensitivity and consistency evaluation, stability evaluation, usability evaluation and load balance evaluation.

4.1. Sensitivity and Consistency Evaluation

Definition 21 (sensitivity named **sensitivity**). The sensitivity of entity i in nth state represents the average deviation degree between the real trust and the actual trust of entity i, which can be described as $\text{sensitivity}_n^i \in \mathbf{sensitivity}$,

$$\text{sensitivity}_n^i = \frac{\left|\widehat{T}_n^i - T_n^i\right|}{\max\left(\widehat{T}_n^i, T_n^i\right)}. \tag{19}$$

Based on above theory, trust is a significant parameter in trust model. In community I_3, we artificially set the frequencies of attacks at 2nd entity, 4th entity, 6th entity, 10th entity as ma_res = 0%, ma_res = 12.5%, ma_res = 25%, ma_res = 100%, respectively. If the real trusts of these 4 entities are 0, the attacks situation distribution will be like Figure 3.

As is shown in Figure 3, 4th entity will tell other entities that its own real trust is 0.6 every 4 services, but 6th entity will tell other entities that its own real trust is 0.6 every 8 services. The 10th entity will always report its trust value as 0.6, maliciously. The 2nd entity will always report the real trust value. After we compare the trusts and sensitivities of

Input: Entity p, all trusts $\widetilde{T}_n^i, \forall i \in H$ and all usability$(i,t), \forall i \in H$ of entities responding to p for service S, and under decision set $\boldsymbol{\xi}$ in which elements are illogic.
Output: Entity q

```
(1) for each i ∈ H do
(2)    if usability(i,t) > ϑ_u, T̃_n^i > ϑ_T and i ∉ ξ then
(3)       put i ∈ H into collection P
(4)    else if usability(i,t) > ϑ_u, T̃_n^i > ϑ_T and i ∈ ξ then
(5)          put i ∈ H into collection Q
(6)       else
(7)          put i ∈ H into collection R
(8)       end if
(9)    end if
(10) end for
(11) if P ≠ ∅ then
(12)    for each i ∈ P do
(13)       compute the load L(i)
(14)       return the entity q with the smallest load
(15)    end for
(16) else if Q ≠ ∅ then
(17)       for each i ∈ Q do
(18)          compute the load L(i)
(19)          return the entity q with the smallest load
(20)       end for
(21)    else
(22)       for each i ∈ R do
(23)          if T̃_n^i > 0.5 then
(24)             compute the load L(i)
(25)             return the entity q with the smallest load
(26)          else
(27)             return the entity q, randomly
(28)          end if
(29)       end for
(30)    end if
(31) end if
```

Algorithm 2: Load balance for entity $p(\widetilde{T}_n^i, \text{usability}(i,t), \xi, \forall i \in H)$.

these 4 entities according to the known condition in Figure 3, we will gain the situation comparison as Figure 4.

In Figure 4, we can see the sensitivity of SecuredTrust is the lowest. The sensitivities of other 3 entities are relatively bigger. Most of trusts in these four models are from 0.5 to 1. In these 4 sub-figures, only in DecisionTrust model can the 2nd entity trust reach 1, but cannot reach 1 in other 3 models, because other models have used community parameters to compute trust. If there is a malicious entity, all entity trusts cannot reach 1, but can be extremely near to 1.

In Figure 4(a), if there is no attack, the entity trust will rise, but when there exists an attack, it will decline very soon. The 10th entity always launching attacks has been eliminated and never returned the system, after providing the 8th service, which may mean DecisionTrust has overestimated the effects of attacks.

In Figure 4(b), if there is no attack, the entity trust will keep, but when there exists an attack, it will decline with relative smaller amplitude. The 6th entity trust should be lower than the 4th entity trust and the 10th entity trust, factually. However, some trust values of 4th entity are higher than 6th entity's, not agreeing with the fact.

In Figure 4(c), if there is no attack, the entity trust will keep and slowly go up, but when there is an attack, it will decline with smaller amplitude. Whereas, no matter the entity is malicious, the entity trusts are always from 0.55 to 0.85, to a large extent, which reveals the extreme underestimation of the whole system.

In Figure 4(d), if there is no attack, the entity trust will retain and slowly rise, but when there is an attack, it will decline with relative smaller amplitude. There is no overestimation and underestimation in DynamicTrust model, which is relatively eclectic and realistic.

4.2. Stability Evaluation. The mechanism regards the sensitivity variance named **SV** as norm to evaluate stability, referring to [19].

Definition 22 (sensitivity variance named **SV**). The sensitivity variance of entity i represents the fluctuation of the average

Table 3: Parameter settings.

Settings	Parameters	Description	Default
Simulation environment settings	M	#The total number of entities in system	100
	m_3	#The total number of community I_3 in system	22
	k	#The number of malicious entities	0%
	ma_res	#The frequency of a malicious entity launching attack	0%
	ma_rep	#The percentage of collusion attack in malicious attacks	0%
	interval	#The average interval for each service (/minute)	10
	C	#The total number of service times	60
Parameter computation settings	Θ	#The trust threshold about entities	{0.8, 0.6, 0.5, 0.2}
	β	#The indirect satisfaction factor	0.3
	N	#The length of time window	4
	μ	#The expectation value of each service in computation indirect satisfaction	1
	g	#The output number of function $\min_g(L)$ in load balance	1
	ϑ_u	#The threshold of entity usability in load balance	0.5
	ϑ_T	#The threshold of entity trust in load balance	0.4

Table 4: Stability comparison.

SV	DynamicTrust	SecuredTrust	PeerTrust	DecisionTrust
SV of the 2nd entity	0.0000	0.0000	0.0000	0.0000
SV of the 4th entity	0.0072	0.0072	0.0016	0.0165
SV of the 6th entity	0.0141	0.0141	0.0015	0.0100
SV of the 10th entity	0.0513	0.0513	0.0003	0.0005

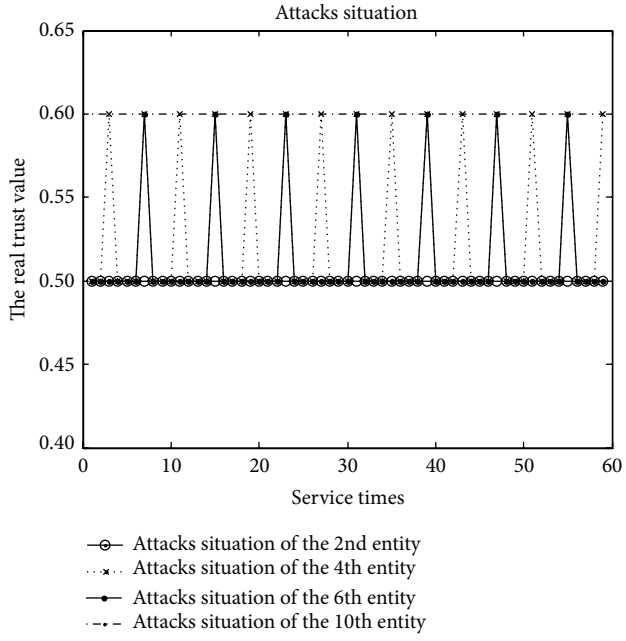


Figure 3: Attaks distribution situation.

deviation degree between the real trust and the factual trust of entity i, which can be described as

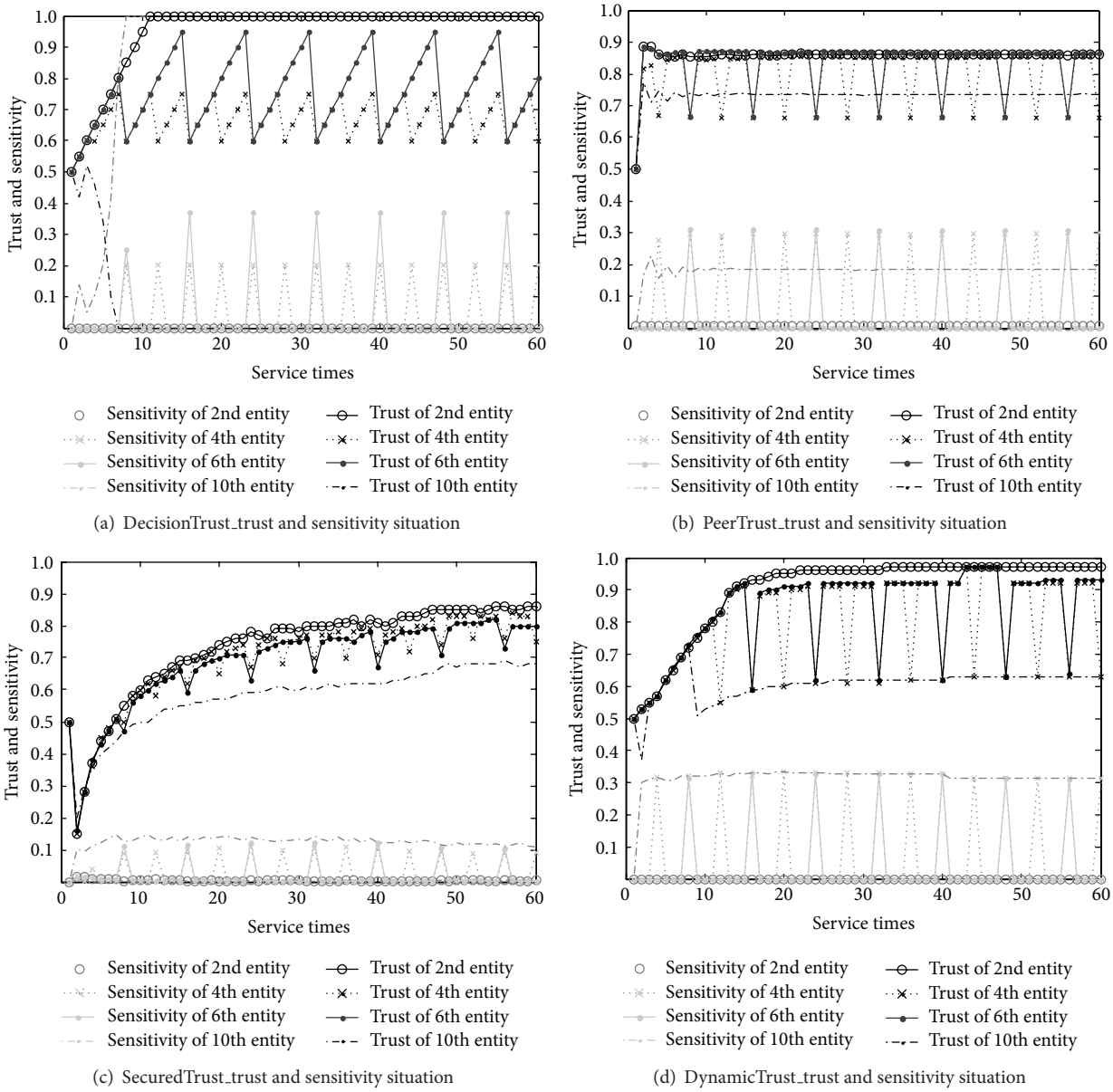
$$\mathrm{SV}_i = \frac{1}{C}\left(\underset{1\le n\le C}{\mathrm{mean}}\left[\mathrm{sensitivity}_n^i\right] - \mathrm{sensitivity}_n^i\right)^2, \quad (20)$$

where C is the total number of service times.

Malicious entities want to alter its own trust to mislead other entities, which will arouse the fluctuations of entity trust. According to sensitivity variances of 2nd, 4th, 6th, and 10th entities in DynamicTrust, SecuredTrust [7], PeerTrust [4], and DecisionTrust [8] models, we can obtain variances listed as Table 4 based on sensitivity evaluation in Figure 4.

In Table 4, we can see that the mean SV of PeerTrust is the smallest, but that of DynamicTrust and that of SecuredTrust are higher. After 60 services, the SV of 10th entity in DynamicTrust reaches 0.05, which is the same in DecisionTrust. Luckily, all sensitivity variances of entities in those 4 models are small.

4.3. Entity Relative Usability. Based on the known condition in Figure 3, we can also gain the usability of 2nd, 4th, 6th and



(a) DecisionTrust_trust and sensitivity situation

(b) PeerTrust_trust and sensitivity situation

(c) SecuredTrust_trust and sensitivity situation

(d) DynamicTrust_trust and sensitivity situation

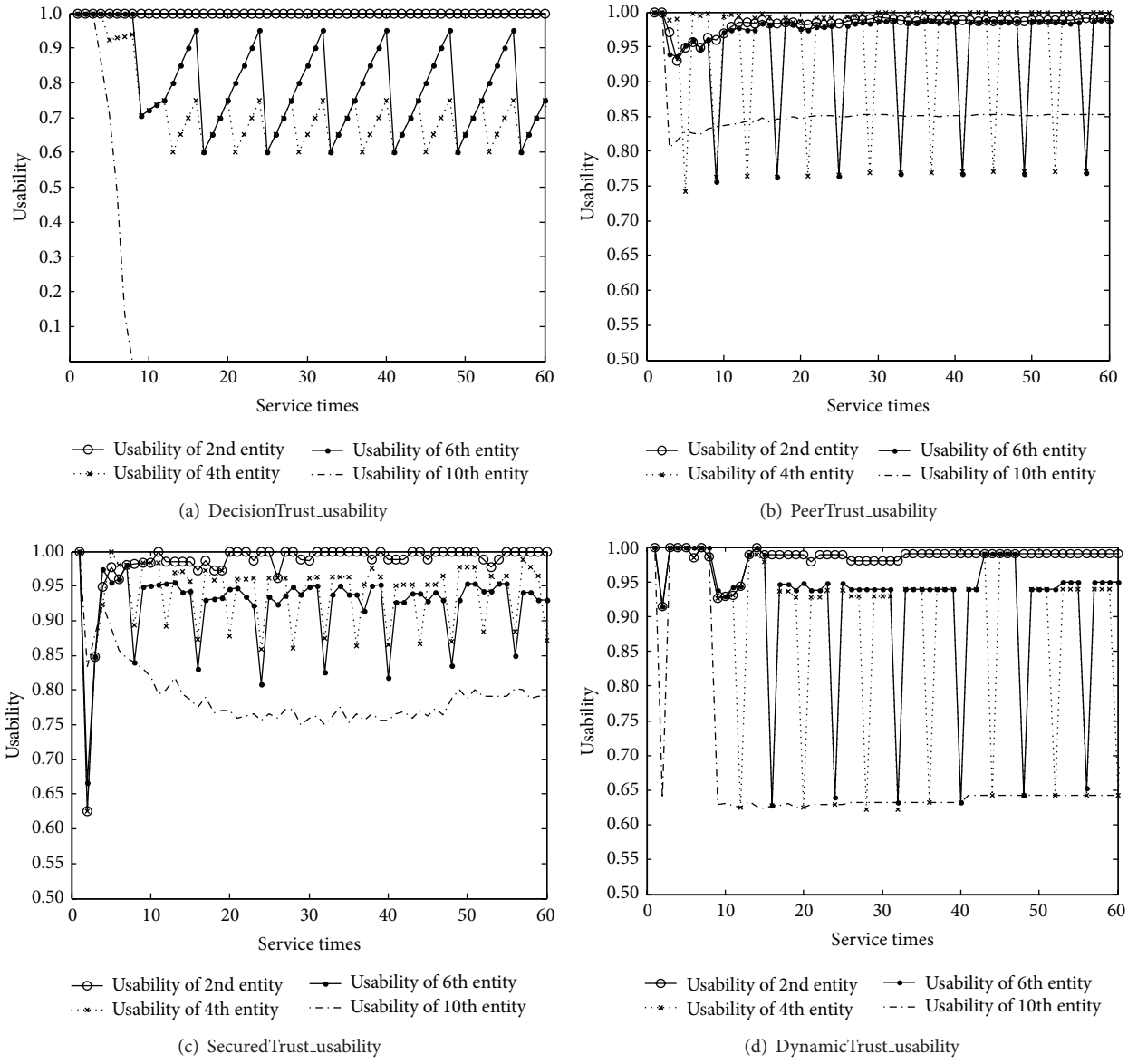
Figure 4: Trust and sensitivity situation.

10th entities as Figure 5. There is a regular that the usability of 2nd entity should be higher than that of 4th entity and that of 6th entity. The usability of 10th entity should be the lowest.

From Figure 5(a), we know most of the relative usability of entities is from 0.6 to 1 and they meet the usability regular. However, the 10th entity has also been eliminated and never returned the system, after providing the 8th service. In Figure 5(b), the relative usability is from 0.75 to 1, but does not meet the usability regular. Sometimes, the usability of 10th entity is higher than that of 4th entity, not in line with the fact. In Figures 5(c) and 5(d), the usability of entities both meets the usability regular and is in line with reality.

4.4. Load Evaluation. In this section, we suppose there are 1000 services and 12 entities in a system. The 1000 services contains 250 type A services, 250 type B services, 250 type C services and 250 type D services. The trust threshold set is $\Theta = \{0.8, 0.7, 0.5, 0.4\}$. The entity parameters are listed as Table 5.

We suppose that entity trust threshold and usability threshold are both 0.7. After an entity emits service request, PeerTrust will randomly select service object from the most credible entities with trusts bigger than 0.7. DecisionTrust will always select service object from the entities whose trusts and usability are both bigger than 0.7. SecuredTrust has its own load balance strategy, which preferentially selects service object from the entities with trusts bigger than 0.7 or randomly selecting entity as service object. DynamicTrust will select service object, according to the service type, load, logicality, usability, and entity trust. For type A services,



(a) DecisionTrust_usability

(b) PeerTrust_usability

(c) SecuredTrust_usability

(d) DynamicTrust_usability

Figure 5: Entity relative usability.

Table 5: Entity parameter settings.

Entity	1	2	3	4	5	6	7	8	9	10	11	12
Entity trust	0.95	0.96	0.88	0.85	0.85	0.75	0.71	0.6	0.55	0.5	0.5	0.45
Logical	Yes	Yes	Yes	No	No	No	No	No	No	No	No	No
Usability	0.85	0.88	0.8	0.79	0.78	0.72	0.68	0.65	0.6	0.54	0.66	0.5

the entities with higher usability and trusts can be service objects. For type D services, the entities with lower usability and trusts can become service objects, thus making full use of all entities existing systems. We have obtained Figure 6, comparing the loads of DynamicTrust, PeerTrust, DecisionTrust, and SecuredTrust.

As is indicated in Figure 6, PeerTrust will always select 1st entity, or 2nd entity, or 3rd entity as service object. The average load of PeerTrust is high. DecisionTrust will select service object from 1st entity to 6th entity, so its average load is lower. SecuredTrust will select service object from 1st entity to 7th entity. DynamicTrust will select service object with load balance strategy for different services. For type D service with lower requirement, DynamicTrust selects service object from 8th entity to 12th entity. The average load of DynamicTrust is the lowest.

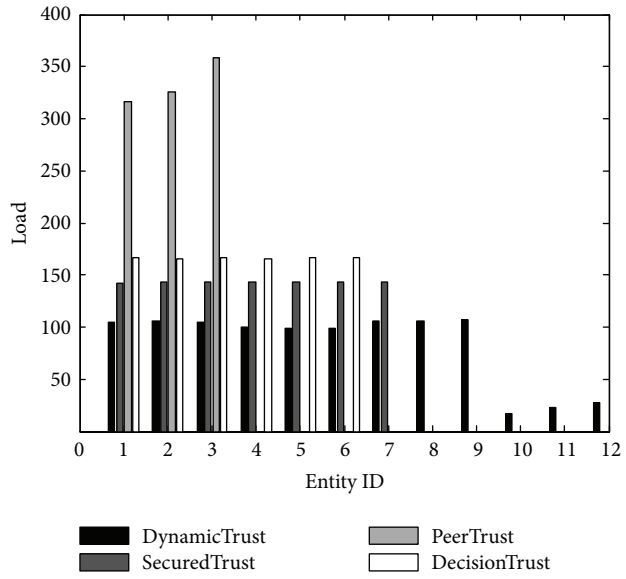


Figure 6: Load comparison.

5. Conclusion

Aiming at existing research problems, this paper proposes a decision-aided situation awareness mechanism based on multiscale dynamic trust in wireless network. DynamicTrust computes and defines satisfaction, trust, and other parameters based on Ebbinghaus forgetting regular from time perspective and spatial relationships from space perspective. We have also used usability, capability, and trust tests to form feedback and aid decision making and assessment. Compared with 3 other models, DynamicTrust is relatively eclectic and realistic for trust mensuration, which can also make full use of entities in system, avoiding resource congestion. However, the trust situation may arise interrupted. To emergencies, the mechanism should make more perfect strategies based on historical and current information, which needs a large-scale database. That is the challenge of situation awareness technology, remaining to be improved.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by the National Nature Science Foundation of China (nos. 61271260; 61102062; 61301122), NSF of Chongqing (no. cstc2014jcyjA40052), the Research Program of Chongqing Municipal Education Commission (no. KJ1400405), Program for Changjiang Scholars and Innovative Research Team in University (IRT1299), the special fund of Chongqing key laboratory (CSTC), NSF of CQUPT (no. A2013-30), and the Doctor Science Research Starting Foundation of CQUPT (no. A2013-23).

References

[1] J. Webb, A. Ahmad, S. B. Maynard, and G. Shanks, "A situation awareness model for information security risk management," *Computers & Security*, vol. 44, pp. 1–15, 2014.

[2] K. R. Sarkar, "Assessing insider threats to information security using technical, behavioural and organisational measures," *Information Security Technical Report*, vol. 15, no. 3, pp. 112–133, 2010.

[3] S. Ganeriwal, L. K. Balzano, and M. B. Srivastava, "Reputation-based framework for high integrity sensor networks," *ACM Transactions on Sensor Networks*, vol. 4, no. 3, article 15, 37 pages, 2008.

[4] L. Xiong and L. Liu, "PeerTrust: supporting reputation-based trust for peer-to-peer electronic communities," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, pp. 843–857, 2004.

[5] F. L. Li, Y. F. Nie, F. Liu, J. Zhu, and H. B. Zhang, "Event-centric situation trust data aggregation mechanism in distributed wireless network," *International Journal of Distributed Sensor Networks*, vol. 2014, Article ID 585302, 11 pages, 2014.

[6] Y. Hou, Y. Xiong, X. Wang, and X. Liang, "The effects of a trust mechanism on a dynamic supply chain network," *Expert Systems with Applications*, vol. 41, no. 6, pp. 3060–3068, 2014.

[7] A. Das and M. M. Islam, "SecuredTrust: a dynamic trust computation model for secured communication in multiagent systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 2, pp. 261–274, 2012.

[8] D. Jelenc, R. Hermoso, J. Sabater-Mir, and D. Trček, "Decision making matters: a better way to evaluate trust models," *Knowledge-Based Systems*, vol. 52, pp. 147–164, 2013.

[9] Y. A. Kim and H. S. Song, "Strategies for predicting local trust based on trust propagation in social networks," *Knowledge-Based Systems*, vol. 24, no. 8, pp. 1360–1371, 2011.

[10] G. Yin, Y. Wang, Y. Dong, and H. Dong, "Wright-Fisher multi-strategy trust evolution model with white noise for Internetware," *Expert Systems with Applications*, vol. 40, no. 18, pp. 7367–7380, 2013.

[11] J. Wu and F. Chiclana, "A social network analysis trust-consensus based approach to group decision-making problems with interval-valued fuzzy reciprocal preference relations," *Knowledge-Based Systems*, vol. 59, pp. 97–107, 2014.

[12] X. Yang, Y. Guo, and Y. Liu, "Bayesian-inference-based recommendation in online social networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 4, pp. 642–651, 2013.

[13] Z.-P. Fan, W.-L. Suo, B. Feng, and Y. Liu, "Trust estimation in a virtual team: a decision support method," *Expert Systems with Applications*, vol. 38, no. 8, pp. 10240–10251, 2011.

[14] M. Brunelli and M. Fedrizzi, "Fuzzy m-ary adjacency relations in social network analysis: optimization and consensus evaluation," *Information Fusion*, vol. 17, no. 1, pp. 36–45, 2014.

[15] M. W. Hassan, R. McClatchey, and I. Willers, "A scalable evidence based self-managing framework for trust management," *Electronic Notes in Theoretical Computer Science*, vol. 179, pp. 59–73, 2007.

[16] H. Marzi and M. Li, "An enhanced bio-inspired trust and reputation model for wireless sensor network," *Procedia Computer Science*, vol. 19, pp. 1159–1166, 2013, Proceedings of the 4th International Conference on Ambient Systems, Networks and Technologies.

[17] F. G. Mármol and G. M. Pérez, "Towards pre-standardization of trust and reputation models for distributed and heterogeneous

systems," *Computer Standards & Interfaces*, vol. 32, no. 4, pp. 185–196, 2010.

[18] K. Govindan and P. Mohapatra, "Trust computations and trust dynamics in mobile adhoc networks: a survey," *IEEE Communications Surveys and Tutorials*, vol. 14, no. 2, pp. 279–298, 2012.

[19] S. V. Halunga and N. Vizireanu, "Performance evaluation for conventional and MMSE multiuser detection algorithms in imperfect reception conditions," *Digital Signal Processing*, vol. 20, no. 1, pp. 166–178, 2010.