INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, 9(5):570-583, October, 2014.

Trust Model in Cloud Computing Environment Based on Fuzzy Theory

L. Gu, J. Zhong, C. Wang, Z. Ni, Y. Zhang

Lichuan Gu, Chengji Wang, Youhua Zhang
School of Computer and Information, Anhui Agricultural University
No.130 ChangJiang Road, Hefei, Anhui, 230036 China
glc@ahau.edu.cn, wang_chengji@qq.com, yhzhang@ahau.edu.cn

Jinqin Zhong
University School of International Business
No.420 Linquan road, Hefei, Anhui, 230031 China.
jinqinzhong@163.com

Zhiwei Ni, Lichuan Gu
School of Management, Hefei University of Technology
No.9 Tunxi Road, Hefei, Anhui, 230009 China
gdnzw@hfut.edu.cn

Abstract: Recent years have witnessed the development of cloud computing. However, there also come some security concerns in cloud computing environment, such as emerging network attacks and intrusions, and instable cloud service provision due to flexible cloud infrastructure and resources. To this end, we research on the trusted computing in cloud computing environment. Specifically, in this paper, we propose a trust model based on virtual machines, with two considerations. First, we introduce timeliness strategy to ensure the response time and also minimize the idle time of servers. Second, we extend the linear trust chain by differentiating the trust of the platform domain and user domain. Besides, we develop a fuzzy theory based method to calculate the trust value of cloud service providers. We also conduct some experiments to evaluate our method.

Keywords: Trust model, fuzzy theory, cloud computing.

1 Introduction

Recent years have witnessed the development of cloud computing, which is an integration of parallel computing, grid computing and distributed computing [1, 2]. With massive computing and storage capability, cloud computing provides various resources to end users through trusted and reliable services. In this way, users can be relieved from trivial management routines and stay focused on the interesting business only. For example, cloud services can help to reduce the complexity of enterprise informatization process, improve the efficiency of companies' operation, and facilitate the utilization of computer resources [3].

However, although cloud computing brings us extremely convenience, there also comes some security concerns [4, 5]. The security flaws are growing ever since the complexity of system softwares increases. Also, the increasingly development of internet, as well as emerging network attacks and intrusions [6], leads to more security events. Besides, the flexibility of cloud infrastructure and resources increases the difficulty of management and brings instability. Moreover,there might be single point failure in demanding for high quality trusted service, and thus the cloud service delivery could be delayed or failed. For example, suitable authentication is required for access to bank accounts, healthrecords, intellectual property and business or politically sensitive information to reduce the security risks of cloud computing infrastructure [7].

To this end, trusted computing [8,9] is proposed by the Trusted Computing Group (TCG) [10]. With trusted computing technologies, computers can be safer and less prone to viruses and mal ware, and therefore hardware and software can consistently behave in expected manner. One way to ensure the functionality of cloud infrastructure through trusted computing is to leverage the idea of chains of trust.

In this paper, we propose a trust model and its evaluation method in cloud computing environment. Specifically, (1) the trust model in this study is built in cloud computing environment instead of traditional scenarios; (2) since time factor is significant for both QoS in requesting cloud services and also maximizing the utilization of cloud resources, we consider timeliness strategy in choosing trusted cloud services; (3) we extend the traditional linear trust chain as a tree-like structure to differentiate the trust of the platform domain and user domain; and (4) the evaluation of trusted computing is based on the fuzzy theory.

The remains of this paper are organized as follows. In Section 2 we provide some related work. Section 3 presents our proposed trust model and the fuzzy theory based evaluation method. Then experiments are conducted in Section 4. Finally, the paper is concluded in Section 5.

2 Related work

In this section, we present some related work. Generally, there are three categories: theoretical research on trusted computing (TC),architecture and implementation of TC, and TC for virtual machines environment.

The first category is theoretical efforts on trusted computing. Blaze et al. [11] first proposed the concept of trust management in 1996. Then, Josang et al. [12–14]proposed a trust model based on subjective logic, and introduced evidence space and opinion space to measure the trust relationship. Beth et al. [15] classified trust into direct trust and indirect trust, and proposed to measure the trust based on the degree of task completion. Fault tolerance capability in trusted computing within the whole life cycle of software development was also discussed [16–18]. Smith et al. [19] developed a outbound authentication model using IBM secure coprocessors. Abadi et al. [20] provided a formal description of the access control process in NGSCB system using secure logical language. Chen et al. [21]described the process of secure bootstrap in trusted computing using predicate logic.

There are also many efforts on the architecture and implementation of TC. IBM 4758 secure coprocessors [22] are one of the most earliest secure hardware. The design of IBM 4758 is to provide an isolated running environment to ensure the computing and storage capability even when something happens to the operating system or the main processors. Stanford University developed an architectural support for copy and tamper resistant software, called XOM [23]. Suh et al. [24] developed AEGIS, an architecture for a single-chip aegis processor which can be used to build computing systems secure against both physical and software attacks. Chen et al. [25] designed another secure processor Cerium. BEAR [26, 27] constructs trusted computing in commercial trusted platforms in Linux, and extends trust chain to the folder layer by checking the integrity of folders when they are first opened. Also, IBM proposed an architecture for trusted computing called IMA [28].

The last category of related work is trusted computing for virtual machines. Garfinkel et al. [29] developed a virtual machine-based platform for trusted computing, called Terra. It allows applications with a wide range of security requirements to run simultaneously on commodity hardware. Also, IBM implemented vTPM [30] to support trusted computing for multiple virtual machines, by introducing a virtual layer and extending the trust chain for virtual machines based on typical trusted platform module (TPM).

3 Proposed trust model

In typical cloud computing environment, the service model is multi-layered. That is, the cloud service provider (CSP) not only provides services to end users, but also to the upper layer CSPs, which forms a trust chain of service providers and consumers. As shown in Figure 1, the arrow denotes from providers to consumers. For example, the service of end users might be provided by a SaaS CSP directly, or first PaaS CSP then SaaS CSP, or from IaaS CSP downstream to end users.

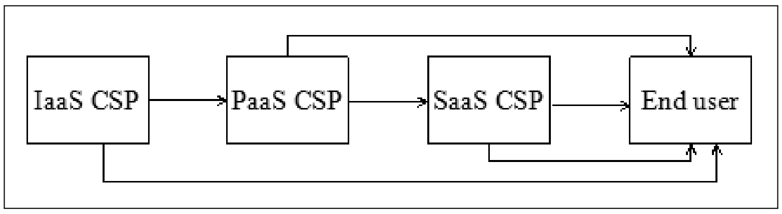

Figure 1: Trust transitivity service providers and consumers

There are two kinds of trust in cloud computing environment: direct and indirect trust. Direct trust means the impression of consumer users on the service quality of CSP, while indirect trust denotes the aggregated impression of all other previous consumers who have used the service or other CSPs who have connections with current CSP.

Suppose user u_a wants to use the cloud service from CSP X .The objective is to calculate the trust value of X and determine if X is trusted. Denote $\text{Imp}(a, b)$ as the impression of b on a. Therefore, the trust value of X for u_a can be calculated as:

$$\text{Imp}(X, u_a) = C_1 \sum_{u_i \in U} \alpha_i \text{Imp}(X, u_i) + C_2 \sum_{Y} \beta_i \text{Imp}(X, Y) \tag{1}$$

where $\text{Imp}(X, u_i)$ is the impression value of u_i on CSP X , $\text{Imp}(X, Y)$ is the impression of other CSPs on X, and C_1, C_2, α_i, β_i are coefficients.

3.1 Timeliness strategy

In a cloud computing environment, host data nodes provide trusted services with high credible and stable resources, where timeliness is a significant indicator. Generally, the process of trusted cloud services is as follows. First, if there exist idle host data nodes, then evaluate the timeliness of the node. Second, calculate the trust value of data node based on the timeliness and trust model. If the trust value is satisfactory, then the node is assigned for cloud services; otherwise, repeat the process for the next service request.

We assume that the trust value is related to time, and the more recent the evaluation is, the more contribution it has to the current trust calculation. Let $\text{Imp}(X, u_i)_L$ be the trust value at time L Therefore, the trust value of X for u_a can be represented by rewriting Equation (1) through introducing timeliness strategy:

$$\text{Trust}(X, u_a) = \text{Imp}(X, u_a)_L - \Delta\text{Imp} \cdot F(t - L) \tag{2}$$

where $\text{Imp}(X, u_a)_L$ is the current impression value at time L , and $\Delta\text{Imp} \cdot F(t - L)$ indicates the affect of previous trust value.

Suppose there are k historical impression values before L, and $\text{Imp}(X, u_a)_L$ is defined as:

$$\text{Imp}(X, u_a)_L = \sum_{i=1}^{k} \text{Imp}(X, u_a)_i w_i \tag{3}$$

where $\sum_{i=1}^{k} w_i = 1$.

Since the history impression evaluation happens randomly, we equally split the history time period into m segments in order to differentiate the importance of each historical evaluation. Historical impressions within the same time window W are assigned an identical weight ω_W . Therefore, inspired by [31], weight w_i can be calculated as:

$$w_i = \sum_{j=1}^{n} \frac{\sqrt[m-1]{\omega_{W_1}(m-j)\omega_{W_m}(j-1)}}{n} \tag{4}$$

where n is the number of values within i-th historical impression. Substitute Equation (4) into (3), we get

$$\mathrm{Imp}(X, u_a)_L = \sum_{i=1}^{k} \mathrm{Imp}(X, u_a)_i \sum_{j=1}^{n} \frac{\sqrt[m-1]{\omega_{W_1}(m-j)\omega_{W_m}(j-1)}}{n} \tag{5}$$

Suppose the experience distribution of $\Delta\mathrm{Imp}$ follows Gumbel distribution [32]. Therefore the estimate of $\Delta\mathrm{Imp}$ is:

$$\hat{G}(\mathrm{Imp}_i) = \exp\left\{ -\exp\left\{ -\frac{\mathrm{Imp}_i - \hat{u}}{\hat{\delta}} \right\} \right\} \tag{6}$$

where $\hat{u}$, $\hat{\delta}$ is the maximum likelihood estimates for Gumbel distribution.

Time based function $F(t-L)$ follows exponential distribution:

$$F(t-L) = \int_{L}^{+\infty} (t-L)f(t)dt = \frac{\exp(-\lambda L)}{\lambda} \tag{7}$$

3.2 Tree structured trust chain

In this section, we consider the trust between a specific CSP and users. The objective is to measure the trust value from the CSP hardware to user defined software.

Trust relationship in cloud environment is more complicated than the traditional scenarios. For example, the services are typically running in virtual machines with larger management and user domain, which is hardly measured by the traditional linear structure. Moreover, for some business requirements, services are assembled across multiple user domains, which also increase the difficulty of measuring the trust. Note that by management domain, we mean the set of objects, typically refers to the component in traditional trust chain; by user domain, we mean that different users that request for the service.

Typically, cloud users don't have controls over the hardware devices, and thus the safety is only ensured by service-level agreement (SLA). However, users wish to somehow control the virtual computing resources, i.e., virtual machines (VM), so that user-defined security strategy can be passed over VM and therefore ensure the safety of cloud resources.

One popular solution is to use trusted platform module (TPM), which is a specialized chip that can securely store information, such as passwords and encryption keys, with independent execution CPU unit. The typical architecture of TPM is shown in Figure 2. TPM provides trusted computing by ensuring security to operating systems and TCG Software Stack (TSS). TSS is a support software of TPM, and its architecture is shown in Figure 3.

In cloud computing environment, trusted computing is typically implemented by virtualizing the Trusted Platform Module (vTPM). However, vTPM is only a virtual instance of physical TPM in user domain, which calls the physical TPM resource to provide TPM service, as shown

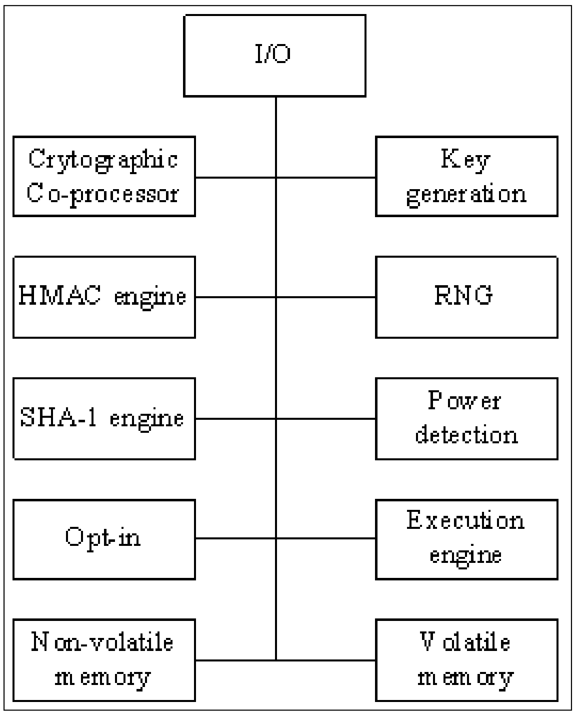

Figure 2: TPM component architecture

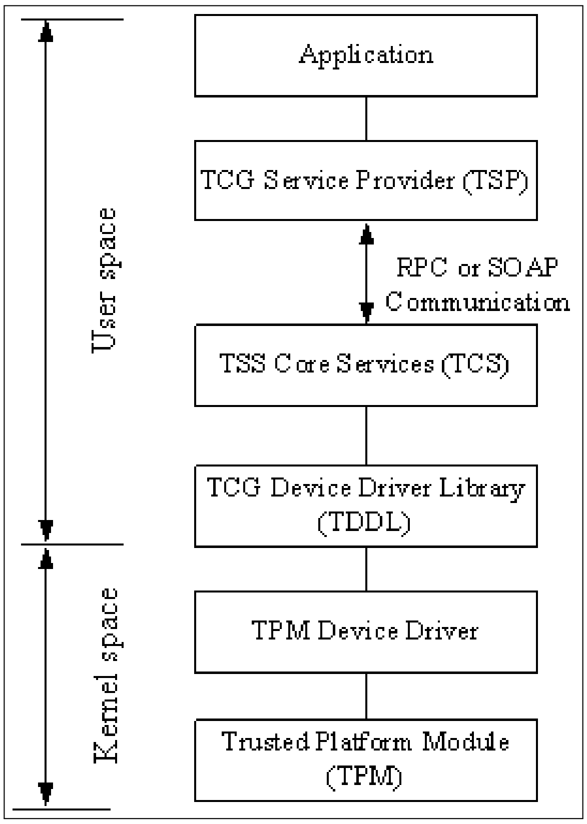
User space
Kernel space
Application
TCG Service Provider (TSP)
RPC or SOAP Communication
TSS Core Services (TCS)
TCG Device Driver Library (TDDL)
TPM Device Driver
Trusted Platform Module (TPM)

Figure 3: Architecture of TSS

in Figure 4. There are some limitations of this simple structure. First, the whole structure is dependent on limited physical TPM, which is not scalable for large scale virtual machines in cloud platform. Second, evaluation of each individual vTPM is sequential due to the limited physical resource, and therefore it is not dynamic and elastic. Last, vTPM relies on physical TPM, that is, the trusted capability is ensured by the cloud infrastructure only. Accordingly, users cannot specify their personalized security strategies on demand. Moreover, we observe that existing trust models, where the evaluation is all performed by one single node, are inappropriate especially in cloud computing environment [33].

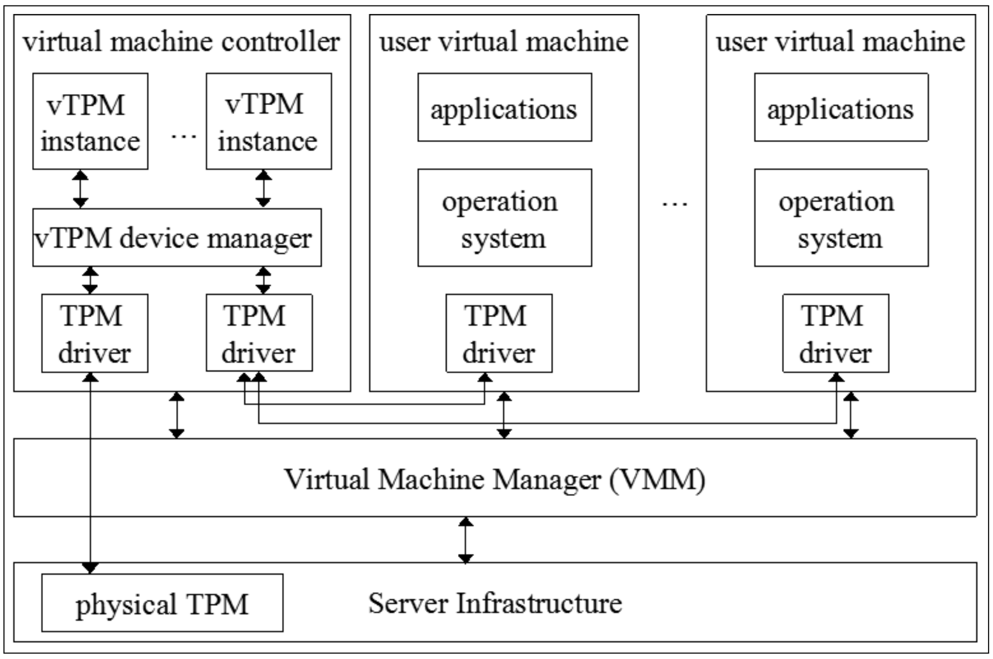

Figure 4: Illustration of user virtual machine

To this end, in this section, we propose a tree structured trust model by distributing the evaluation work to multiple nodes. Specifically, we combine the trust evaluation from physical platform and user domain. As shown in Figure 5, there are two stages in the evaluating of trust value. First, there is a chain of trust in physical TPM, i.e., CRTM'BIOS'GRUB'VMM. The integrity and security of the system is ensured by the isolation mechanism of physical TPM and cloud infrastructure. Second, TPM controller creates TPM for each user, i.e., uTPM, which is responsible for the evaluation and security of software components in the user domain. Each uTPM is created for a user virtual machine, and holds the results of integrity evaluation and reports to users.

There are two kinds of trust transitivity in this model. (1) Pass the trust from physical TPM to TPM controller, from hardware to user virtual machine. In this way, we can combine the trust of platform and the trust of user virtual machine together, to provide a complete trust chain. (2) Pass the trust from user to virtual machine. By virtualizing TPM, an independent vTPM is created for each user virtual machine, i.e., uTPM. In Figure 5, each node in the upper physical trust chain is responsible for evaluating trust, and the evaluation is performed for each user in parallel in the lower user domain trust chain.

Theorem 1. *The tree structured trust model in Figure is trusted.*

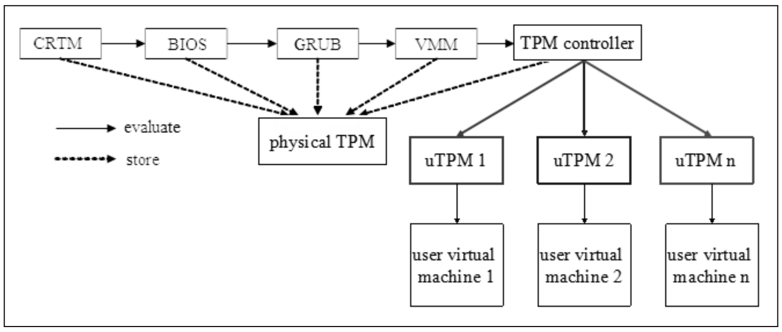

Figure 5: Illustration of tree structured trust model

Proof: For ease of description, we simplify the tree structured trust model in Figure 6. Denote the trust value of node A for B as $T(A,B)$. Inferred by the trust transitivity principle, we have $T(A,B) = \min\{T(A,B), T(B,C), \ldots, T(N-1,N)\}$. As proved in [24], the left part chain $A \to B \to C \to \cdots \to N$ is trusted.

Now we consider the right part chain. Each node U_i represents a trust chain of each user virtual machine, and therefore U_i itself is trusted. There are multiple direct paths, i.e., $T_1(N,U_1), T_2(N,U_2), \ldots, T_N(N,U_N)$. The final trust value $T(N,U)$ is no less than

$$\max\big\{T_1(N,U_1), T_2(N,U_2), \ldots, T_N(N,U_N)\big\}.$$

Therefore the right part chain is trusted. □

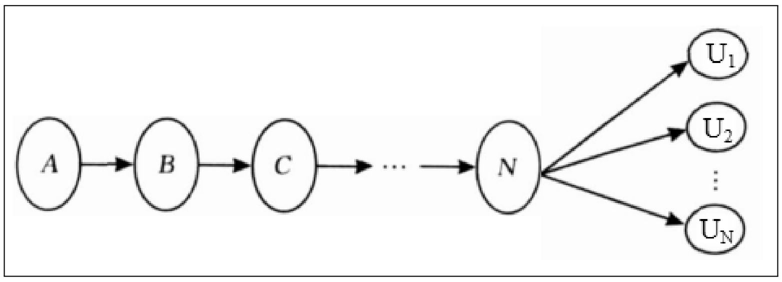

Figure 6: Illustration of tree structured trust model

3.3 Fuzzy theory based trust model

As used in many Suppose the confidence level for trust is $U = \{U_1, U_2, U_3, U_4\}$ = {'not trusted', 'somehow trusted', 'normal trusted', 'complete trusted'}, and the confidence vector $V = \{v_1, v_2, v_3, v_4\}$, where $v_i (i = 1,2,3,4)$ denotes the degree of membership of each level U_i.

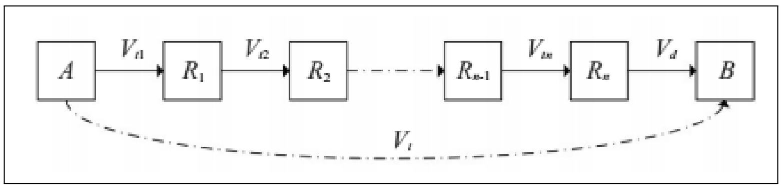

Figure 7: Illustration of trust transitivity

The trust between initial entity and target entity is evaluated through the stepwise evaluation and the transitivity between them. As shown in Figure 7, where A is observed entity, B is evaluation entity, and $R_1, R_2, \ldots, R_n$ are the intermediate entities. The corresponding trusts between nodes are $V_{t1}, V_{t2}, \ldots, V_{tn}, V_d$, and the trust vector from A to B is V_t . The integrity of direct measurement is mainly affected by the measuring capability of the current evaluation node.

Now we consider the timeliness factor. Let $DT(R_i, Rj, t)$ be the direct trust of R_j in R_i at time t , and it can be calculated as:

$$DT(R_i, R_j, t) = \frac{S_{R_jR_i}}{S_{R_jR_i} + F_{R_jR_i}}\delta(t, t_0) \tag{8}$$

where $S_{R_jR_i}$ denotes the number of successful historical integrity evaluation of R_j on R_i, $F_{R_jR_i}$ is the number of failure evaluation, and t, t_0 denote the current and first evaluation time respectively. $\delta(t, t_0)$ is the time decay function, defined as:

$$\delta(t, t_0) = 1 - \frac{t - t_0}{t}\xi \tag{9}$$

where $\xi \in [0, 1]$ is the adjustment factor of the decay.

The value of direct trust $DT(R_i, R_j, t)$ can be transformed into a fuzzy vector $V(i, j) = (v_1, v_2, v_3, v_4)$, where

$$v_1 = \begin{cases} 1 - 2\left(\frac{DT}{0.5}\right)^2, & 0 \le DT \le 0.25; \\ 2\left(\frac{0.5-DT}{0.5}\right)^2, & 0.25 < DT < 0.5; \\ 0, & 0.5 < DT < 1. \end{cases} \tag{10}$$

$$v_2 = \begin{cases} 0, & 0 \le DT \le 0.25; \\ 1 - 2\left(\frac{DT-0.25}{0.5}\right)^2, & 0.25 < DT \le 0.25; \\ 2\left(\frac{0.75-DT}{0.5}\right)^2, & 0.5 < DT \le 0.75; \\ 0, & 0.75 < DT \le 1. \end{cases} \tag{11}$$

$$v_3 = \begin{cases} 0, & 0 \le DT \le 0.25; \\ 2\left(\frac{DT-0.25}{0.5}\right)^2 & 0.25 < DT \le 0.5; \\ 1 - 2\left(\frac{0.75-DT}{0.5}\right)^2 & 0.5 < DT \le 0.75; \\ 0 & 0.75 < DT \le 1. \end{cases} \tag{12}$$

$$v_4 = \begin{cases} 0, & 0 \le DT \le 0.5; \\ 2\left(\frac{DT-0.5}{0.5}\right)^2 & 0.5 < DT \le 0.75; \\ 1 - 2\left(\frac{DT-1}{0.5}\right)^2 & 0.75 < DT \le 1. \end{cases} \tag{13}$$

Suppose there are n evaluation metrics $Z = \{z_1, z_2, \ldots, z_n\}$, we get the evaluation matrix for entity R_i :

$$V_i = \begin{bmatrix} v_{1,1} & v_{1,2} & \cdots & v_{1,n} \\ v_{2,1} & v_{2,2} & \cdots & v_{2,n} \\ v_{3,1} & v_{3,2} & \cdots & v_{3,n} \\ v_{4,1} & v_{4,2} & \cdots & v_{4,n} \end{bmatrix} \tag{14}$$

Therefore, the indirect trust of R_i is calculated as:

$$IT_i = W_i \cdot V_i \tag{15}$$

where $W_i = (w_1, w_2, w_3, w_4)$ is the weights of each metric, and $w_i \in [0,1]$, $\sum\limits_{i=1}^{4} w_i = 1$. We employ a fuzzy reasoning method based on similarity. Suppose there exist a rule:

$$R : \text{If } A \text{ Then } B, \lambda, W \tag{16}$$

where A is the antecedent component, B is the consequent component of the rule, and λ is a threshold which decides the rule to be executed or not.

The similarity between two entities is calculated as:

$$S(A\prime_i, A_i) = \begin{cases} \frac{M(A\prime_i \cap A_i)}{M(A\prime_i) \vee M(A_i)}, & \text{if } A\prime_i \subseteq A_i \text{ or } A\prime_i \supseteq A_i \\ \frac{A\prime_i \cap A_i}{M(A\prime_i)}, & \text{else.} \end{cases} \tag{17}$$

where $A\prime_i$, A_i are fuzzy set, $M(A_i) = \sum\limits_{x \in X_i} \mu_{A_i}(x)$, X_i is the mathematical domain of discourse for $A\prime_i$, A_i.

As shown in Figure, in our case, the antecedent components are $V_{t1}, V_{t2}, \ldots, V_{tn}$, and the consequent component is V_t. That is, the rule is:

$$\begin{aligned} R : & \text{ If the measurement of } A \text{ for } R_1 \text{ is } V_{t1} \\ & \text{AND the measurement of } R_1 \text{ for } R_2 \text{ is } V_{t2} \\ & \text{AND } \cdots \\ & \text{AND the measurement of } R_n \text{ for } B \text{ is } V_d \\ & \text{Then the measurement of } A \text{ for } B \text{ is } V_t \end{aligned} \tag{18}$$

Given the observed values are $V\prime_{t1}, V\prime_{t2}, \ldots, V\prime_{tn}$, the goal is to calculate the observed $V\prime_t$. The process of inference is as follows.

Step 1: Calculate the similarity between observed $S(V\prime_{ti}, V_{ti}), i = 1, 2, \cdots, n$;

Step 2: If $S(V\prime_{ti}, V_{ti}) > \lambda$, calculate the overall similarity:

$$S_W(V\prime_{ti}, V_{ti}) = \sum_{i=1}^{n} S(V\prime_{ti}, V_{ti})^* \frac{w_i}{\sum\limits_{j=1}^{n} w_j} \tag{19}$$

Step 3: Compute the inference results:

$$\theta_1 = \sum_{i=1}^{k_1} \frac{S(V\prime_{ti}, V_{ti})^* w_i}{\sum\limits_{i=1}^{k_1} w_j} \tag{20}$$

where $j \in \{i : M(V\prime_{ti}) \geq W(V_{ti})\}$, and

$$\theta_2 = \sum_{i=1}^{k_2} \frac{S(V\prime_{ti}, V_{ti})^* w_i}{\sum\limits_{i=1}^{k_2} w_j} \tag{21}$$

where $j \in \{i : M(V\prime_{ti}) < W(V_{ti})\}$.
If $\theta \neq 0$ and $\theta_2 \neq 0$, then

$$V\prime_t = \begin{cases} \frac{V_t * \theta_1}{\theta_2}, & \theta_1 \leq \theta_2; \\ \min\left\{1, \frac{V_t * \theta_1}{\theta_2}\right\}, & \text{otherwise.} \end{cases} \tag{22}$$

If $\theta_1 = 0$ or $\theta_2 = 0$, then

$$V_t = \begin{cases} V_{t^*} S_W, & \theta_1 \leq \theta_2; \\ \min\left\{1, \frac{V_t}{S_W}\right\}, & \text{otherwise.} \end{cases} \tag{23}$$

To sum up, in this section we presented the proposed trust model, which works as follows. When user a wants to use some service, for each qualified candidate CSP X, he/she first inquires the information from other users and CSPs who have interactions with X. Then, for each X, a tree structured trust chain is constructed, where the upper part is the trust chain for physical TPM of X, and the lower part is built for each possible user. After that, an evaluation method based on fuzzy theory is performed to calculate the trust value of a for X. Once all the trust value is learned, user a can determine if a CSP X is trusted or not.

4 Experiment

In this section, we evaluate the efficiency of our proposed tree structured trust model. The configuration of PC is as follows. Intel Core i5 2.8 GHz CPU with four cores, 4 GB memory, 500 G hard disk. We use Xen virtualization platform for virtualization implementation, CloudSim [34] for cloud computing platform simulation, and Matlab for fuzzy system implementation.

4.1 Evaluating trusted cloud service selection

Figure 8 shows the trust value of four types of CSPs with different number of transactions. We have the following observations. First, for complete trusted and normal trusted CSPs, the trust value is near linearly growing since they are offering real trusted services. Second, for somehow trusted CSPs, the trust value is unstable. Third, the value of not trusted CSPs decreases quickly, and they would not be selected as the cloud service provider.

4.2 Evaluating timeliness

We compare our method with traditional Dynamic Level Scheduling (DLS) algorithm [35] for scheduling cloud services with the consideration of timeliness factor. The results are reflected as the average of 100 executions.

Figure 9 shows the ratio of successful execution with different numbers of tasks. We can observe that the successful execution tends to increase when the number of tasks is growing. However, for DLS algorithm, the maximum success ratio is around 0.65. The reason is that it does not consider the node failure at specific time. By contrast, our method performs better because we consider the timeliness factor in all historical evaluations.

Figure 10 shows the average schedule length with different numbers of tasks. The average schedule length grows when there are more tasks. Also, the average schedule length of DLS is smaller than that of our method. Therefore, we can see that our method achieves larger success ratio by sacrificing the scheduling time. Although the time cost increases, our method can help to select more successful trusted cloud services.

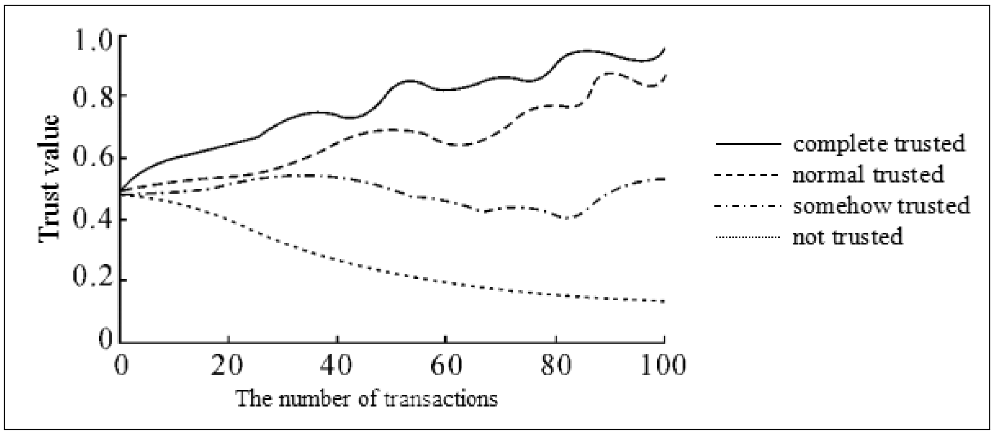

Figure 8: Trust value of four types with different number of transactions

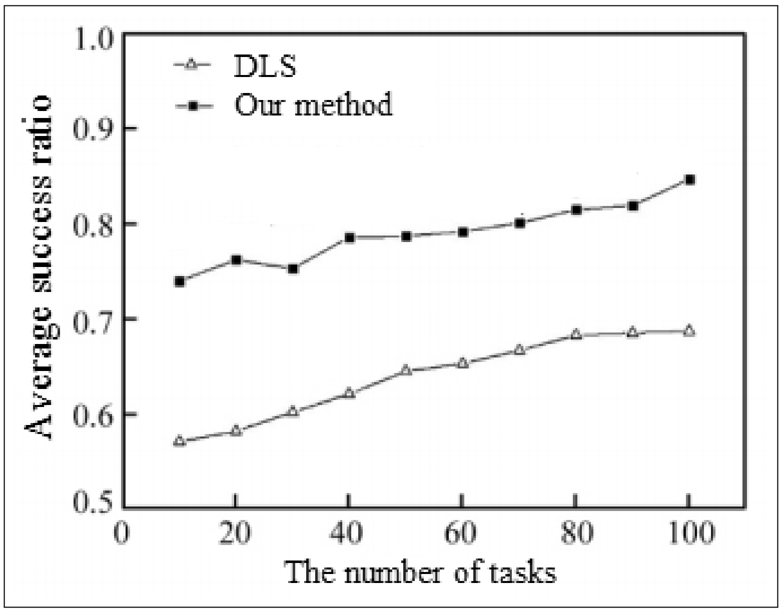

Figure 9: The ratio of successful execution with different numbers of tasks

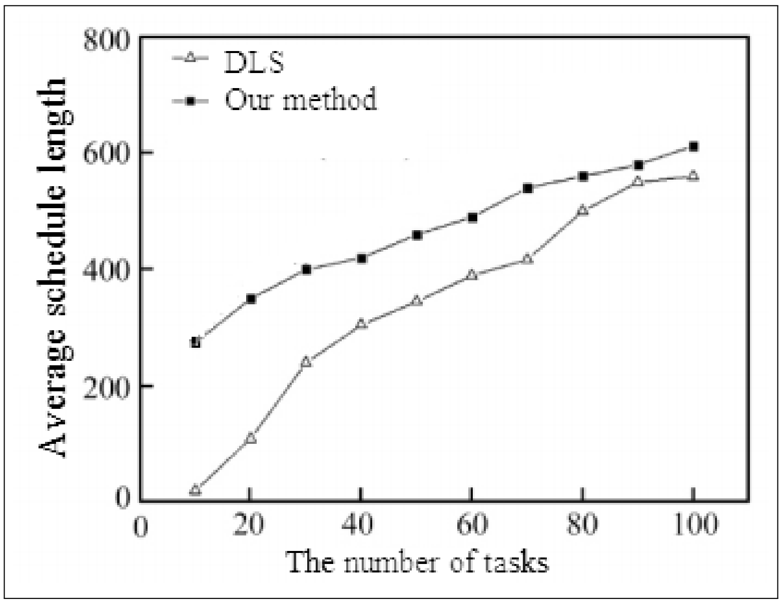

Figure 10: Average schedule length with different numbers of tasks

5 Conclusion

In this paper, we proposed a trust model in cloud computing environment. Specifically, the trust model is designed for virtual machines with the consideration of timeliness factor. Moreover, we employ a fuzzy theory based method to calculate the trust value of specific CSP.

In our experiments, we exhibit the trust value of four pre-defined confidence levels, and also evaluate the efficiency of the timeliness consideration. We find that our method can improve the successful response of selecting cloud services at the expense of average schedule length.

However, in future works, we might want to explore a more optimal balance between efficiency and effectiveness.

Acknowledgements

This work was supported by the National Natural Science Foundation of china (Grant No.313715
33, Grant No. 71271071), National Science and Technology Support Program (Grant No. 2013BAJ10B12), the Key Technologies R & D Program of AnHui Province (Grant No.1301032169) and by the Natural Science Foundation of AnHui Province (Grant No.1308085MF89).

Bibliography

[1] Armbrust, Michael, et al. (2010); A view of cloud computing, *Communications of the ACM*, 53(4): 50–58.

[2] Mell P., Grance T. (2011); The NIST definition of cloud computing, `http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf`, 1-7.

[3] Lin C., Pervan G. (2001); A review of IS/IT investment evaluation and benefits management issues, problems and processes, in *Information technology evaluation methods and management*, ISBN:1-878289-90-X, 2-24.

[4] Brodkin J. (2008); Gartner: Seven cloud-computing security risks. *Infoworld* (2008): 1–3.

[5] Zissis D., Lekkas D. (2012); Addressing cloud computing security issues. *Future Generation Computer Systems*, 28(3): 583–592.

[6] Lonea A.M., Popescu D.E., Tianfield H.(2012); Detecting DDoS Attacks in Cloud Computing Environment, *International Journal of Computers Communications & Control*, 8(1): 70–78.

[7] Popescu D.E, , Lonea A.M. (2013); An Hybrid Text-Image Based Authentication for Cloud Services, *International Journal of Computers Communications & Control*, 8(2): 263–274.

[8] Pearson S., Balacheff B., eds. (2003); *Trusted computing platforms: TCPA technology in context*, Prentice Hall Professional.

[9] Mitchell C. ed.(2005), *Trusted computing*, Institution of Electrical Engineers.

[10] Sumrall N., Novoa M. (2003); Trusted Computing Group (TCG) and the TPM 1.2 Specification. *Intel Developer Forum.* Vol. 32.

[11] Blaze M., Feigenbaum J., Lacy J. (1996); Decentralized trust management. *Security and Privacy, 1996 IEEE Symposium on*, 164-173.

[12] Josang A. (2001); A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(3): 279–311.

[13] Knapskog, S. J.(1998); A metric for trusted systems, *Proc. of the 21st National Security Conference*, Available at `http://folk.uio.no/josang/papers/JK1998-NSC.pdf`, 1-14.

[14] Josang A. (1999); Trust-based decision making for electronic transactions, *Proc. of the Fourth Nordic Workshop on Secure Computer Systems*, 1-21.

[15] Beth T., Borcherding M., Klein B. (1994); *Valuation of trust in open networks*, Springer Berlin Heidelberg.

[16] Meyer J. F. (1980); On evaluating the performability of degradable computing systems. *Computers, IEEE Transactions on*, 100(8): 720–731.

[17] Isermann R. (1984); Process fault detection based on modeling and estimation methods, a survey. *Automatica*, 20(4): 387–404.

[18] Arlat J. et al.(1993); Fault injection and dependability evaluation of fault-tolerant systems, *Computers, IEEE Transactions on*, 42(8): 913–923.

[19] Smith S. W. (2002); Outbound authentication for programmable secure coprocessors. *Computer Security, ESORICS, 2002. Springer Berlin Heidelberg*, 72–89.

[20] Abadi M., Wobber T. (2004); A logical account of NGSCB. Formal Techniques for Networked and Distributed Systems, CFORTE 2004. Springer Berlin Heidelberg, 2004. 1–12.

[21] Chen S., Wen Y., Zhao H. (2007); Formal analysis of secure bootstrap in trusted computing, *Autonomic and Trusted Computing*, Springer Berlin Heidelberg, 352–360.

[22] Dyer J. G., et al. (2001); Building the IBM 4758 secure coprocessor. *Computer*, 34(10): 57–66.

[23] Lie, David, et al.(2000); Architectural support for copy and tamper resistant software, *ACM SIGPLAN Notices* , 35(11): 168–177.

[24] Suh G. E. et al. (2003); AEGIS: architecture for tamper-evident and tamper-resistant processing. *Proc. of the 17th annual international conference on Supercomputing. ACM*, 1-18.

[25] Chen B., Morris R. (2003); Certifying Program Execution with Secure Processors, *HotOS*, Available at `http://pdos.csail.mit.edu/papers/cerium:hotos03.pdf`, 1-6.

[26] MacDonald R. et al. (2003); Bear: An open-source virtual secure coprocessor based on TCPA. *Computer Science Technical Report TR2003-471*, Dartmouth College.

[27] Marchesini J. et al.(2003); Experimenting with TCPA/TCG hardware, or: How I learned to stop worrying and love the bear. *Computer Science Technical Report TR2003-476*, Dartmouth College .

[28] Sailer R. et al. (2004); Design and Implementation of a TCG-based Integrity Measurement Architecture. *USENIX Security Symposium*, 13:223-238.

[29] Garfinkel T. et al.(2003); Terra: A virtual machine-based platform for trusted computing, *ACM SIGOPS Operating Systems Review*. 37(5):193-206.

[30] Berger S. et al. (2006); vTPM: virtualizing the trusted platform module, *Proc. 15th Conf. on USENIX Security Symposium*, 305-320.

[31] Fullér R., Majlender P. (2001); An analytic approach for obtaining maximal entropy OWA operator weights, *Fuzzy Sets and Systems*, 124(1): 53–57.

[32] Saure D. et al. (2010); Time-of-use pricing policies for offering cloud computing as a service, *Service Operations and Logistics and Informatics (SOLI), 2010 IEEE International Conference on*, 300-305.

[33] Bo Z. et al.(2010); The system architecture and security structure of trusted PDA, *Chinese Journal of Computers*, 33(1): 82–92.

[34] Calheiros R. N. et al.(2011); CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1): 23–50.

[35] Wang W. et al. (2012); Dynamic trust evaluation and scheduling framework for cloud computing. *Security and Communication Networks*, 5(3): 311–318.