ServiceTrust: Trust Management in Service Provision Networks

Zhiyuan Su[*+] *Ling Liu*[+] *Mingchu Li*[*] *Xinxin Fan*[*+] Yang Zhou[+]
*Software school, Dalian university of Technology, Dalian, China
+College of computing, Georgia Institute of Technology, Atlanta, GA, USA
suzhiyuan2006@gmail.com, lingliu@cc.gatech.edu, mingchul@dlut.edu.cn, xinxinyuanfan@mail.dlut.edu.cn, yzhou86@gatech.edu

***Abstract*—Service provision networks are popular platforms for decentralized service management. eBay and Amazon are two representative examples of enabling and hosting service provision networks for their customers. Trust management is a critical component for scaling service provision networks to larger set of participants. This paper presents ServiceTrust, a quality sensitive and attack resilient trust management facility for service provision networks. ServiceTrust has three unique features. First, it encapsulates quality-sensitive feedbacks by multi-scale rating scheme and incorporates the variances of user's behaviors into the local trust algorithm. Second, ServiceTrust measures the similarity of two users' feedback behavior and aggregate the local trust values into the global trust algorithm by exploiting pairwise feedback similarity scores to weight the contributions of local trust values towards the global trust of a participant. Finally, pairwise feedback similarity weighted trust propagation is utilized to further strengthen the robustness of global trust computation against malicious or sparse feedbacks. Experimental evaluation with independent and colluding attack models show that ServiceTrust is highly resilient to various attacks and highly effective compared to EigenTrust, one of the most popular and representative trust models to date.**

Keyword-trust management, multi-scale rating, trust propagation, similarity, attack-resilient

I. INTRODUCTION

Service oriented computing refers to architectures, platforms and technologies of packaging and delivering computing capabilities that are common to multiple domain problems as services. In the last decade, we have witnessed the evolution of service provision platforms from in-house to cloud and from centralized client-server model to service provision network model. A unique feature of a service provision network is to allow every participant to be service provider and service consumer at the same time and to bridge between service consumers and service providers on demand. For example, when purchasing a product from Amazon or eBay, a ranked list of sellers is offered to the users as service providers. This ranking is based primarily on consumers' feedback ratings obtained through their prior transaction experiences with the service providers. Thus, such rankings can serve as valuable references for those users who have no prior experience with or no prior knowledge about the providers. Although the service provision network model offers the opportunities for consumers to be connected to unfamiliar service providers and for providers to reach a larger and growing customer base, the opportunity to interact with unknown providers also opens doors for potential risks of dishonest ratings and malicious manipulations.

Introducing the trust management into a service provision network system has proved to be an effective way to improve the trustworthiness of the system, as evidenced in eBay and Amazon. Trust is regarded by many as one of the most important measures for sharing information and developing new consumer-provider relationships. Trust management has attracted active research in several areas of computer science, such as peer to peer systems, sensor networks, social networks, eCommerce, mobile computing systems and applications, to name a few. Most of the trust models are based on per-transaction feedbacks. Thus the trust of a provider is computed in two steps: First, for each pair of provider and consumer, a local trust is computed by aggregating the set of feedback ratings provided by the consumer who has had transactions with the provider. Second, a global trust of a provider is computed based on the set of local trusts this provider has received from the consumers in the service provision network.

Existing trust management models differ from one another in three aspects: (i) how the users' feedback ratings are aggregated in computing the trust of providers, (ii) how resilient the local and global trust computation is against dishonest feedbacks and malicious manipulations, and (iii) how trust is computed in the presence of sparse feedbacks and cold start. For example, different methods are used to aggregate feedback ratings, ranging from simple algorithmic aggregation methods such as those used in eBay to more complex feedback aggregation methods based on statistical significance, such as naïve Bayesian with majority voting, Bayesian belief networks, eigenvector and so forth. Unfortunately, most of the existing approaches have been developed independently and little efforts have been made to compare and understand the relative strength and inherent vulnerabilities of different approaches. Concretely, how dishonest feedbacks and sparse feedbacks may impact on the effectiveness of the existing trust models? How the cold start (new comers) problem is handled, and how robust and resilient the existing trust models are and whether the proposed trust model will remain to be effective in the presence of some known attacks. We believe that answers to these questions are critical to critical for building an attack resilient trust management for service provision networks.

Based on these observations, we present ServiceTrust, an attack resilient trust management scheme for service provision networks. ServiceTrust offers two distinct capabilities for establishing and managing trust in a service provision network. First, with multi-scale service level agreements (SLAs) becoming pervasively employed in evaluating quality of services today, ServiceTrust provides a quality-sensitive aggregation method to encapsulate multi-scale feedback ratings as a generalization of binary feedback rating scheme. Thus, the local trust computed based on multi-scale ratings is more sensitive to quality differentiation and more resilient to colluding attacks. Second, ServiceTrust enhances the attack resilience of global trust computation through two steps: (i) ServiceTrust introduces the pairwise feedback similarity as a measure of feedback quality, aiming at preventing the detrimental effects of dishonest feedbacks and malicious manipulation of feedback ratings on the validity of global trust value of a participant. (ii) ServiceTrust computes the global trust of a participant by utilizing the trust propagation kernel powered with the feedback similarity based propagation control. This capability enables us to effectively discredit those participants who are strategically malicious and to control the amount of trust propagating from a good participant to a malicious participant. We have conducted an extensive experimental evaluation to show the effectiveness and efficiency of ServiceTrust. To the best of our knowledge, ServiceTrust is the first to exploit multi-scale rating similarity based trust propagation to provide attack resilience in trust management for service provision networks.

II. Related Work

Reputation management often refers to trust management in a large social network of participants where trust of a member participant is computed based on the feedback ratings he or she receives from the rest of the participants. An overview of some key issues in reputation management is given in [8]. Trust metrics on graphs were given [1]. The notion of transitive trust, though initially presented in [1], was made popular by EigenTrust [6] through the use of Eigen vector [7] based propagation kernel. The feedback credibility concept and its role for defending against dishonest feedbacks is first introduced in PeerTrust[11]. In addition, [11] also describes the roles of transaction context may play in making trust model more resilient to attacks. Trust relationships in Web-based social networks are studied in [2,4,5,9]. [10] proposed a fuzzy logic based aggregation method to compute trust. [12] utilizes contextual similarity to evaluate trust in eCommerce environment. The contribution of this work is two folds. First, we analyze the vulnerability of existing trust models in the presence of dishonest feedbacks, sparse feedbacks, colluding behaviors. Second, we show through the development of ServiceTrust how to utilize pairwise feedback similarity during the aggregation of local trust assessments of all participants in an efficient and robust manner.

III. Background and Overview

A. Reference Trust Model

In a service provision network of *N* participants, each participant can be a service provider and also a service consumer. Thus the relationship between any two pair of participants simulates the peer to peer relationship.

When a peer P_i acting as a provider to respond to a service request from another peer P_j, the receiving peer P_j is allowed to enter a feedback rating on P_i in terms of the quality of the service provided by P_i. We refer to this as a per-transaction based feedback.

1) Computing local trust by aggregating feedbacks

Let $tr(i, j)$ denote the feedback rating from peer *i* to peer *j* where $i,j \in [1,\ldots,N]$ and *tr(i,j)* is initialized to zero. Using a binary rating scheme, when a peer *i* competes a transaction with another peer *j* at time *t*, then peer *i* may rate the transaction it has with peer j as positive by $tr(i, j, t) = 1$ or negative by $tr(i, j, t) = -1$. Let $sat(i, j)$ denote the total number of satisfactory transactions between peer *i* to peer *j*, and $unsat(i, j)$ denote the number of unsatisfactory transactions between peer i to peer *j*. We define $s(i, j)$ as the aggregation of all feedback ratings from peer *i* to peer *j*, namely, $s(i, j) = sat(i, j) - unsat(i, j)$. Clearly, $s(i, j)$ is a positive integer if there are more positive feedback ratings and negative otherwise.

Now we compute the local trust based on *s(i,j)*, denoted by c_{ij}. We need to normalize the local trust values into the range of [0,1] in order to make the comparison meaningful between peers with high volume of transactions and peers with low volume of transactions. For example, we can define c_{ij} by normalizing $s(i, j)$ using the maximum satisfactory score from all participants who have had the direct transactional experiences with peer *i* as follows:

$$c_{ij} = \begin{cases} \dfrac{\max(s(i,j),0)}{\sum_j \max(s(i,j),0)} & if \sum_j \max(s(i,j),0) \neq 0 \\ p_j & otherwise \end{cases} \quad (1)$$

$$p_j = \begin{cases} 1/|P| & if \ j \in P \\ 0 & otherwise \end{cases}$$

In Formula (1), *P* denotes a set of pre-trusted seed participants (pre-trusted peers). We use the pre-trusted seed peers to bootstrap the system initially and to allow new comers to be linked to some existing participants in the network.

2) Computing global trust by trust propagation kernel

Let *n* denote the total number of participants in our service network system. We define $C = [c_{ij}]$ as a matrix of *n* rows by *n* columns. Let t^0_i denote the initial global trust value of peer *i*. We have t^0_i =1/n.

Let t^k_i denote the *k*-hop global trust value of peer *i*. We have $t^1_i = \sum_{j=1}^n c_{ji} / n = \sum_{j=1}^n c_{ji} t^0_i$. Let $\vec{t}^{\,k} = (t^k_1, \ldots, t^k_i, \ldots, t^k_n)$ denote the global trust vector of size *n* at kth iteration ($1<k<n$). The general formula for computing the global trust vector at the (*k+1*) iteration is:

$$\vec{t}^{\,k+1} = C^T \vec{t}^{\,k} \quad (2)$$

Thus we have $\vec{t} = (C^T)^m \vec{t}^{\,0}$. The trust vector $\vec{t}$ will converge to the left principal eigenvector of *C* if *m* is large enough [6,7]. For each element of this global trust vector, t_i, it quantifies how much trust the system as a whole places on the participant *i*.

Recall the cold start problem in terms of new comers or participants with no feedback ratings from others, we address this problem by utilizing a set *P* of pre-trust seed peers as the bootstrap peers. Any participant that does not know whom to trust can always trust one of the pre-trust peers with a probability of 1/|P|. Thus, we can also initialize the golbal trust verctor $\vec{t}^{\,0}$ by some distribution $\vec{p}$ over pre-trust peers, e.g., 1/|P|. Furthermore, we need to handle malicious collectives who aim at manipulating the system by giving each other high local trust values and giving all others low local trust values in an attempt to gain high global trust values. A common way to break the collectives is by having each participant placing some trust on pre-trust seed peers that are definitely not a part of the collectives. This will avoid getting stuck within a malicious collective.

With these observations in mind, we revise formula (2) by formula (3). Let a be the probability of a participant choosing to trust only some pre-trusted peers. The revised formula for computing the global trust vector of size *n* at *(k+1)th* round of iterations is:

$$\vec{t}^{\,k+1} = (1-a)C^T\vec{t}^{\,k} + a\vec{p} \qquad (3)$$

This formula implies that a peer *i's* global trust at *(k+1)th* iteration can be defined by the sum of the local trust values that other peers have given to *i*, weighted by their global trust values obtained at the *k*th iteration, namely

$$t_i^{k+1} = (1-\alpha)(c_{1i}t_1^k + c_{2i}t_2^k + ... + c_{ni}t_n^k) + \alpha p_i \qquad (4)$$

This reference trust model is known as EigenTrust [6]. Before analyzing the inherent vulnerabilities of this trust model, we first present the common attack models used in this paper.

B. Threat models

In the service provision domain, the following threat models are frequently used to characterize different forms of malicious behaviors.

Threat model A (Independently Malicious)

Malicious users always provide inauthentic services when selected as service providers. Malicious peers always value inauthentic services instead of authentic services.

As malicious participants never provide authentic (good) services, they do not expect getting a good rating from non-malicious participants.

Threat model B (Malicious Collectives)

Malicious participants always provide inauthentic (bad) services when selected as a service provider. In addition, malicious peers form a malicious collectives by giving a high local trust value, say 1, to another malicious peer in the network, leading to a malicious chain of mutually high local values. This malicious collective chain traps non-malicious participants to enter the collective and once a good peer has entered, it will be hard to exit and the global trust value of the good peer will be exploited to boost the global trust values of all peers in the malicious collectives.

Threat model C (Malicious Collectives with Camouflage)

In this type of attack, malicious entities provide an inauthentic service in f% when selected as a service provider. At the same time malicious peers form a malicious collective as describe in the threat model B.

Under this threat model, malicious peers attempt to get positive ratings from some good peers in the network by providing good services sometimes, i.e., when $f > 0$. Consequently, malicious peers in the collective could get higher global trust values.

Threat model D (Malicious spies)

Malicious participants are strategically organized into two groups. One group of malicious peers (type D) try to act as normal users in the system in an attempt to increase their global trust values by only providing good services but provide dishonest feedback ratings. The type D peers act like spies and use their high trust values to boost the trust values of another group of malicious peers (type B) who only provide bad (inauthentic) services when selected as service providers.

There are other types of threat models, such as multiple independent or colluding malicious colletives, which can be represented as a generalized form of the four attack models. Other type of attacks, such as Sybil attacks, can be regulated through some cost based enforcement, such as enforcing the network ID of a participant correspond to a hash value of the peer's unique ID in real life, e.g., drive license number. Often, Sybil attack is used in conjunction with one of the above four threat models. Thus we argue that a highly effective trust model for the service provision networks should be resilient to all of the above four threat models.

C. Trust based service selection models

There are four known service selection schemes:

Random selection. When trust is not supported, a service requestor often randomly selects one provider in the list of matching providers as her preferred provider.

Threshold-based random selection. A requestor randomly select a provider from the subset of providers in the matching list, whose trust values are higher than a given threshold value.

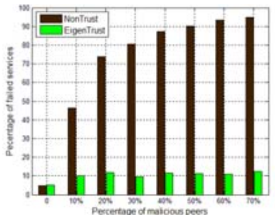
Deterministic selection. A requestor only selects the provider with the highest global trust value among the list of matching providers as her preferred provider. The problem with this approach is the potential of overloading the providers with high trust values.

Probabilistic-based selection. A requestor chooses each matching provider *i* as its preferred provider with probability $t_i / \sum_{j=1}^{M} t_j$ assuming that there are M matching providers that can provide the requested service. In order to give the newcomers a chance to be selected, we can complement the trust enabled probabilistic selection by allowing a peer *j* with zero trust value to be selected at a system defined maximum probability, say 10% [6].

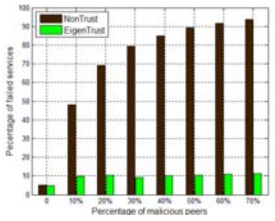
D. Vulnerabilities in the Reference Model

Although EigenTrust by design has incorporated several decisions to increase its attack resilience, which is also the main factors for its high popularity and citation in the literature, EigenTrust has reported [6] that there are some inherent vulnerabilities.

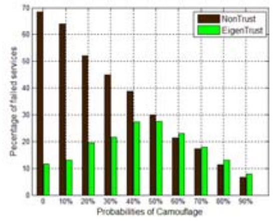
Figure 1 shows that EigenTrust is effective in the presence of varying percentage of malicious peers up to 70% in Threat models A and B where malicious peers either are independently malicious or form a malicious chain of high feedback ratings. However, EigenTrust performs poorly when malicious peers are up to 50% or more in Threat model C. For threat model D, EigenTrust performs worse than non-trust case when the type D peers reach 50% or more of the malicious collective. These experiments used the identical setup as in [6]. In Threat models A and B, the total number of participants is 63, with 3 pre-trust peers. In Threat model C, there are 73 participants in the network, including 20 malicious ones, 3 pre-trust ones and 50 good peers. In Threat model D, there are 103 total participants with 40 malicious participants, which are divided into two groups (type B and type D), 3 pre-trust peers and 60 good peers.



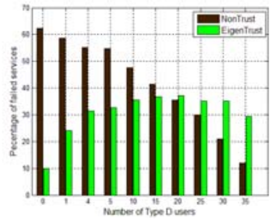
(a)Threat Model A



(b)Threat Model B



(c)Threat Model C



(d)Threat Model D

Figure 1. Robustness and Vulnerabilities of EigenTrust model

Now we analyze the vulnerabilities inherent in the EigenTrust model and illustrate why these vulnerabilities cause EigenTrust to perform poorly compared to non-trust case when the malicious participants strategically collude with one another (Threat models C and D). First, the feedback aggregation formula (1) is more vulnerable when type D malicious peers exist since it fails to distinguish good participants from type D spy peers. Thus, the system fails to recognize the dishonest feedbacks given by type D peers, which harms the good peers and increases type B malicious peers' local trust values. This is one of the most important reasons that EigenTrust fails when 50% or more malicious collectives with camouflage in Figure 3(c) (Threat model C).
Second, the formula (3) for computing global trust vector uses a weighted trust propagation model where the local trusts received by peer i from other peers, say j, are weighted by the global trust value of peer j. This leads to unexpected vulnerability when encountering some sophisticated attacks. Consider Threat model D, type D disguised peers accumulate high global trust values by acting as spy and utilize their high global trust values to boost the global trusts of all type B malicious peers in the collective. This can easily subvert the system as the number of spy peers increases, as shown in Figure 3(d). Figure 4 shows the global trust values of all participants. We observe that except the 3 pre-trust peers, the global trust values of malicious peers are higher than good ones, even when the good peers are 60% and malicious peers are 40% with 30% type D and 10% type B malicious peers.

IV. SERVICETRUST

This section presents the design of ServiceTrust. The design of ServiceTrust should embrace a number of design objectives with respect to performance, ease of maintenance, privacy and attack resilience. First, the system should by design scale to large number of participants in terms of trust computation, storage requirement and communication complexity of the system. This scalability requirement demands high performance system level facilities that can minimize computational complexity and communication overhead for trust management. Second, the system should maintain and respect the anonymity of participants' trust values. Third, the system should not rely on central authority for maintenance and should be self-configuring, self-policing and self-healing. For example, the enforcement of per-transaction based feedback rating and real-world ID based identifier generation should be exercised without central authority. Finally, the system should be robust and attack resilient with respect to failure and malicious manipulations of colluding collectives.

A. Multi-scale Rating Scheme

Multi-scale rating scheme has been widely adopted in many real world eCommerce recommendation systems, such as Amazon, eBay, to provide high quality and fine grain recommendations. We believe that by replacing binary rating with multi-scale rating in ServiceTrust, we can potentially increase the attack resilience by employing quality sensitive metrics to differentiate dishonest ratings by malicious participants from feedback ratings by non-malicious participants. For example, by enabling a peer *i* to rate another peer *j* using a scale of 5 or 10, the feedback ratings can be more accurately reflect the actual quality difference regarding the service received by peer *i* from peer *j*. In the rest of the paper, the following scale will be used by peer i to rate peer *j* regarding the quality of the transaction service provided by peer *j* to peer *i*.

$$tr(i,j) = \begin{cases} -1 & bad \\ 0 & no\ rating \\ 1 & neutral \\ 2 & fair \\ 3 & good \\ 4 & very\ good \\ 5 & excellent \end{cases} \qquad (5)$$

B. Local trust computation by multi-scale ratings

Recall the vulnerability analysis in Section 3, the feedback rating aggregation scheme used in EigenTrust [6] has a number of inherent vulnerabilities. Thus the design of ServiceTrust follows a number of objectives: (i) malicious peers should not accumulate positive ratings from good users by simply performing well in some transactions while providing dishonest feedbacks, such as type D peers in Threat model D or camouflage peers in Threat model C. This implies that ServiceTrust needs to provide capability of identifying such malicious behaviors. (ii) A peer who wants to get a good local trust value from other peers must provide consistently good enough services. Also peers who always get high scale ratings, such as 5 or 4 score in the multi-scale rating scheme

(4) should have higher local trust values than those with always low feedback ratings of 1 or 2. (iii) Malicious behavior, be it a bad service or dishonest feedback rating, should be punished, in the sense that the trust of a peer is hard to build but can drop sharply once detected being malicious.

With these design goals in mind, we propose to compute the local trust value that peer *i* has for peer *j* by using their rating variance. Concretely,. let *J* denote the set of participants that *i* gives rating to. We define *s*(*i, j*) as follows:

$$s(i,j) = \begin{cases} \dfrac{v(i,j)\mu(i,j)(sat(i,j) - unsat(i,j))}{\max(tr(i,J))\sum_k v(k,j)} & if\ \ v(i,j) != 0 \\ \dfrac{\mu(i,j)(sat(i,j) - unsat(i,j))}{\max(tr(i,J))} & otherwise \end{cases} \tag{6}$$

In Formula (6), $v(i,j)$ denotes the variance of all the ratings that *i* gives *j*. Usually, the smaller $v(i,j)$ is, the more stable *j's* service is, which means that *j* should have relatively higher local trust. $v(i,j)$ is computed as follows:

$$v(i,j) = \frac{1}{m} \times \sum_l (tr_l(i,j) - \mu(i,j))^2 \tag{7}$$

In Formula (7), we assume that there are *m* transactions in one spectacular interval. $tr_l(i,j)$ is the *l*th transaction's rating value that *i* gives to *j*. $\mu(i,j)$ is the mean value of all the ratings that *i* gives to *j*. $\mu(i,j)$ is defined as follows:

$$\mu(i,j) = \frac{1}{m} \times \sum_{l=1}^{m} tr_l(i,j) \tag{8}$$

To ensure the local trust value that peer *i* has for peer *j*, denoted by c_{ij} is in the range of [0,1], we normalize *s(i,j)* as follows:

$$c_{ij} = \begin{cases} \dfrac{\max(s(i,j),0)}{\sum_j \max(s(i,j),0)} & if \sum_j \max(s(i,j),0) \neq 0 \\ p_j & otherwise \end{cases}$$

$$P_j = \begin{cases} 1/|P| & if\ \ \text{j} \in P \\ 0 & otherwise \end{cases}$$

C. Global trust computation by trust propagation

In addition to measure pairwise multi-scale rating similarity by variance and mean based normalization, we also measure the rating similarity between two peers based on how they rate another participant. The former allows us to assign higher local trust values to peers that have more similar rating behavior. The latter enables us to distinguish those strategically malicious participants from good peers, because such malicious peers provide satisfactory transactions to get high feedback ratings but provide dishonest feedbacks in an attempt to subvert the system. Therefore, in ServiceTrust, we advocate to incorporate the pairwise feedback similarity into the trust aggregation algorithm. The basic motivation for using feedback similarity weighted trust propagation metric is to differentiate the positive ratings generated by good peers from malicious peers acting as spies. We capture the pairwise feedback similarity in terms of both positive feedback similarity and negative feedback similarity, denoted by *pos_sim(i,j)* and *neg_sim(i,j)* respectively.

Positive similarity. Let *pos_comm(i,j)* denote the subset of common participants that both i and j have given positive ratings. We propose a similarity function based on the *Normalized Euclidean Distance(NED)* as follows:

$$pos\_sim(i,j) = \begin{cases} 1 - \sqrt{\dfrac{\sum_{k \in pos\_comm(i,j)} (\mu(i,k) - \mu(j,k))^2}{|pos\_comm(i,j)|}} & |pos\_comm| > 0 \\ 0 & otherwise \end{cases}$$

$$\mu(i,k) = \sum_{m=1}^{|R(i,k)|} tr_m(i,k) / (|R(i,k)| \times \max(S)) \tag{9}$$

$$\mu(j,k) = \sum_{m=1}^{|R(j,k)|} tr_m(j,k) / (|R(j,k)| \times \max(S))$$

$R(i,k)$ is the total number of transactions happened between peers *i* and *k*. $\mu(i,k)$ is the normalized mean rating value that *i* gives to *k* after $R(i,k)$ transactions. *max_mean(S)* denotes the maximum mean rating value of the entire service network.

Negative similarity. Let *neg_comm(i,j)* denote the subset of common users that both i and j have rated. This measures the negative similarity in terms of the number of peers that peer i and peer j have opposite mean rating values.

$$neg\_sim(i,j) = \begin{cases} 1 - \dfrac{\sum_{k \in neg\_comm(i,j)} count(\mu(i,k),\mu(j,k))}{|neg\_comm(i,j)|} & |neg\_comm(i,j)| > 0 \\ 0 & otherwise \end{cases} \tag{10}$$

$$count(\mu(i,k),\mu(j,k)) = \begin{cases} 1 \ \text{ if } \mu(i,k) \times \mu(j,k) \le 0 \\ 0 \ \text{ if } \mu(i,k) \times \mu(j,k) > 0 \end{cases}$$

The feedback similarity between two users depends on the historical transaction information. Thus $\mu(i,k)$ is the average rating value between *i* and *k* after certain number of transactions. We define *sim(i,j)* as the final similarity value between *i* and user *j* after we get the *neg_sim(i,j)* and *pos_sim(i,j)*.

Let w_n be the weight of negative similarity and w_p be the weight of positive similarity.

$$sim(i,j) = w_n \times neg\_sim(i,j) + w_p \times pos\_sim(i,j)$$

(11)Let $R(i)$ denote the set of peers that have transactions with peers *i*. We can compute the similarity weighted local trust that *i* places on *j*, denoted by sc_{ij} as follows:

$$sc_{ij} = c_{ij} \times (sim(i,j) / \textstyle\sum_{k \in R(i)} sim(i,k)) \cdot$$

To facilitate the comparison of different local trust values, we normalize sc_{ij} by l_{ij} as follows:

$$l_{ij} = \begin{cases} \dfrac{\max(sc_{ij},0)}{\sum_{k \in R(i)} \max(sc_{ik},0)} \\ 0 \end{cases} \tag{12}$$

Let $L = [l_{ij}]$ denote the local trust matrix of size *n* by *n* and $\vec{t}^{\,k+1}$ denote the global trust value vector of size *n* to be computed at *(k+1)th* round of iterations. We define $\vec{t}^{\,k+1} = (1-a)L^T\vec{t}^{\,k} + a\vec{p}$, where a is the probability of a user knows none and thus relies on the pre-trusted peers. We can expand $\vec{t}^{\,k+1}$ as follows:

$$t_i^{k+1} = (1-\alpha)(l_{1i}t_1^k + l_{2i}t_2^k + \ldots + l_{ni}t_n^k) + \alpha p_i \tag{13}$$

V. Experimental Evaluation

This section evaluates the performance of ServiceTrust by comparing it with EigenTrust and Non_Trust service provision network system under the four attack models.

A. *Experimental Setup*

We simulate service provision networks of varying sizes by building a decentralized file sharing service network where peers can request files from other peers in the network (as service consumer) or respond to file download requests from others (as service provider). To make sure that a peer *i* can enter a feedback to another peer *j* only when peer *i* has received a service provided by peer *j,* we create a Gnutella-like peer to peer file sharing network by interconnecting peers using a power-law network (resemble to a real world p2p network). A query issued by any peer will be propagated by iteratively broadcast hop by hop over the chosen hop-count horizon of the network, say 7 hop-horizon.

Nodes in each service network consists of three types: pre-trust nodes whose initial global trust is greater than zero, normal nodes who participate the network for download and upload services, and malicious nodes who are the adversaries attempting to degrade or destroy the quality and performance of the service network system.

We use a probabilistic content distribution model where each peer initially select a number of content categories and share files only in these categories. Within one content category files have different popularizes following Zipf distribution. Files are assigned to peers probabilistically at initialization time based on file popularity and the content categories the peer plans to share.

The simulation of the service network dynamics is done through simulation cycles. Each cycle consists of multiple query cycles. In each query cycle peers can probabilistically choose to ask queries or forward queries or respond to queries. The number of queries issued for different file download services are also based on Zipf distribution. After issuing a query peer waits for response. Upon obtaining a list of providers that responded to the query, the peer select one provider from the list and starts downloading the file. The selection process will be repeated until a user has received an satisfied service. We choose the probabilistic-based selection as the selection method. At the end of each simulation cycle, the local and global trust values are computed. We run each experiment several times and the results of all runs are averaged. The performance metrics used in this paper include the number of inauthentic file downloads (unsatisfactory services) v.s. the number of authentic file downloads (satisfactory services). If the global trust values accurately reflect each peer's actual behavior, then high global trust values minimize the number of inauthentic downloads. We are also interested in time complexity and the iteration rounds used for trust propagation.

To make our comparison with EigenTrust fair, in the first set of experiments, we choose the same set of configuration parameters as those in EigenTrust experiments [6] as shown in Table I. In the second set of experiments, we extend to larger size networks. We show that the trust scheme that does well in small scale networks also performs proportionally well in large networks. We set pre-trusted peers to have 10 initial neighbors. To simulate the hostile environment as is done in [6], we also set malicious peers to have at least 10 initial neighbors and normal peers with at least 2 initial neighbors. This scheme allows malicious peers to be connected to the highly connected peers and to other malicious peers.

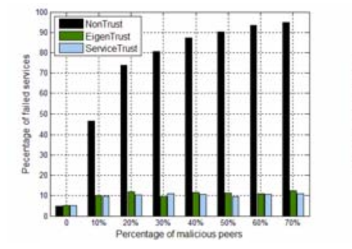
B. *Effectiveness of ServiceTrust*

In this section, we compare the effectiveness of Non_Trust, EigenTrust and ServiceTrust in terms of attack resilience. All simulations are executed under Threat models A, B, C and D. The experiment results are shown in Fig.2.

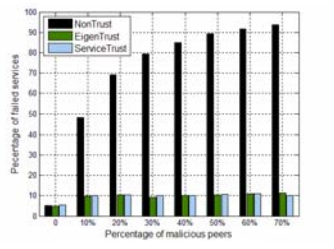
For Threat models A and B, both EigenTrust and ServiceTrust outperforms non_trust systems with consistently about 5%-10% of failed services (malicious downloads) no matter how the ratio of malicious nodes varies in the network from 0% to 90%. Note that the 5-10% of failed services are primarily from normal peers performing unintentionally bad service or due to the use of the maximum 10% probability as newcomer policy, which allows a peer with global trust value 0 to be selected as a provider. This gives the malicious peers with zero trust values 10% probability of being selected as a provider.

TABLE I. SIMULATION CONFIGURATION

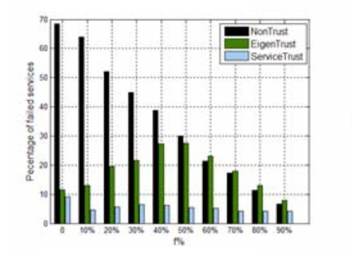
	parameter	**value**
Network	# of good users, pre-trust users and malicious users	good-(60-100) pre-trust(3) malicious(0-42)
	# of neighbors for pre-trust and malicious users	10
	# of neighbors for regular good users	2
	# of hops for forwarding queries	7
Service distribution	Number of distinct services of the whole system	20
	fraction of distinct services at good user *i*	20%
	Set of services categories supported by good peer i	Zipf distribution
	Top % of queries for most popular services malicious users respond to	20%
	Queries of services issued by good user i	Zipf distribution
System Behavior	% of service requests in which good user i provides unsatisfied services	5%
	% of service requests in which malicious user i provides unsatisfied services	Varied in different threat model
	provider source selection algorithm	Probabilistic based and similarity based
	Probability that user with global trust value 0 is selected as service provider	10%



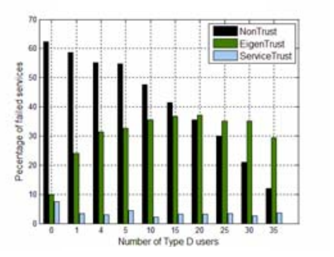
Threat Model A



Threat Model B



Threat Model C



Threat Model D

Figure 2. Fraction of failed services in four threat models

For Threat models C and D, the ServiceTrust model consistently performs well with 5%-10% of failed services (malicious downloads) with the percentage of camouflage malicious nodes is increased to 90% in the network. However, EigenTrust starts failing compared to Non_Trust system when the camouflage probability *f%* reaches 50% in Threat model C. For Threat model D, we simulate a network of 103 peers with 60 normal peers, 3 pre-trust users and 40 malicious users, divided into two groups, regular malicious peers (Type B) and strategically malicious spy peers (type D). We vary the ratio of type D and type B peers in each experiment from 40 type B users, 0 spy users to 5 type B peers, 35 type D peers. ServiceTrust performs consistently better than both Non-Trust system and EigenTrust in all cases. When the number type D peers (spy) reaches 50% of the malicious peers (20 out of 40), EigenTrust starts getting higher number of failed services than Non_trust system as the number of type D peers increases, while ServiceTrust constently performs well with only 5% failed services due to the 10% probabalistic new comer selection policy. The strength of ServiceTrust against strategic malicious attempts attributes to its feedback variance weighted local trust computation and its feedback similarity weighted global trust propagation, which makes use of the sharp difference between the rating behavior of malicious users in the malicious collective and the feedback rating behavior of normal peers to control the trust propagation from good peers to malicious peers.

C. *Time Complexity and Impact of Network Scale*

In the previous sets of experiments, we use the same size of networks as those in [6] in order to provide a fair comparison with EigenTrust. Also the network density in [6] is fairly dense with degree of 10 for malicious peers. In this section we will evaluate ServiceTrust using varying sizes of service provision networks, ranging from 100, 500, 1000, 5000 and 10,000, all with low node degree (on average of 5) to evaluate the performance of ServiceTrust in a sparse network of varying size from 100 to 10,000 participants.

Figure 3 shows that when the scale of network is 100, only 4 iteration rounds are needed to compute global trust for every peer. However, we can only get the global trust values for 14% and 7% of all peers in the network within 4 iteration rounds when the scale of network are 5,000 and 10,000 respectively. As the size of the service network increases, the number of iteration rounds we need to compute global trust through trust propagation also increases. But ServiceTrust can finish the global trust computation through trust propagation in 7 rounds of iteration for all the network scales in this experiment. This shows that even with a network of size 10,000 and average degree of 5, most nodes can reach out to a large population in 7 hops.

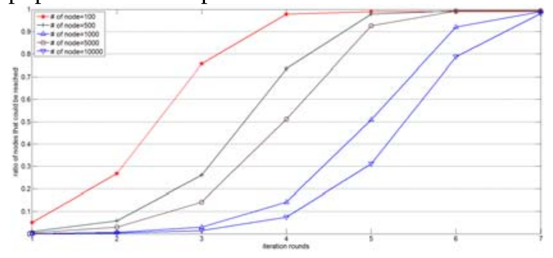


Figure 3. The ratio of reached peers in different iterations

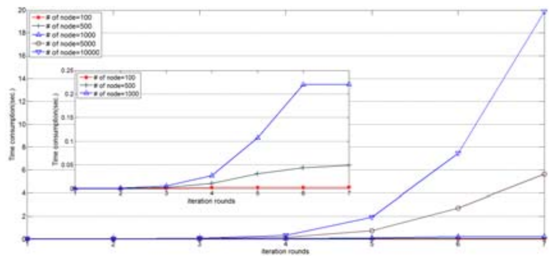
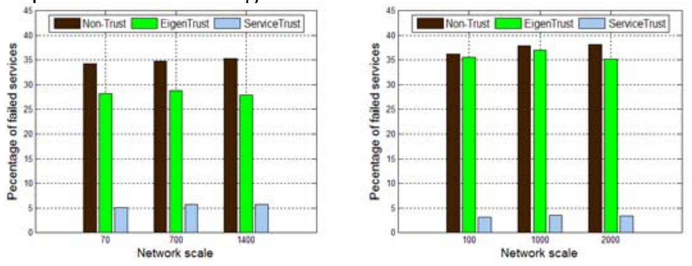


Figure 4. Time Complexity for trust propagation

Figure 4 shows the time complexity of global trust computation in ServiceTrust. As the network size increases from 100 to 10,000, the time needed to compute global trust for all participants through trust propagation will increase. Compare with the network of size 10,000, the time complexity of network size of 100 participants is very tiny. In 7 hops, the time complexity of trust propagation over a network of 100 participants is 0.00282 seconds and the time complexity increases to 20 seconds when the size of the service network reaches 10000.

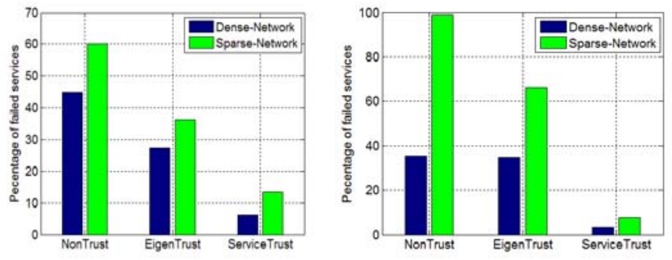
Now we compare ServiceTrust with EigenTrust and Non_trust system in terms of how the percentage of inauthentic downloads varies in Threat model C and Threat model D, when we varying the sizes of networks. From Figure 5 we observe two facts. First, the percentage of inauthentic downloads (failed services) in all three systems is less sensitive to the increase of network sizes. Second, as the size of the network increases, the number of inauthentic downloads (failed services) remains consistently low in ServiceTrust for different sizes of networks in both Threat models C and D. In contrast, EigenTrust has much higher number of failed services in comparison to ServiceTrust. Also in Threat model D, the number of failed services in EigenTrust is close to that of Non_trust system, showing again that EigenTrust is inadequate for managing trust in the presence of colluding malicious collectives.



(a)Threat model C f=0.4 (b)Threat model D type D=20

Figure 5. The effect of network scale on trust models.

Next we compare the performance of Non_Trust, EigenTrust and ServiceTrust in sparse network and dense network. Figure 12(a) shows that in Threat model C, the percentage of failed services in the sparse network is much higher than the result in a dense network for all three systems, about 15% higher in Non_Trust case, 8% higher in EigenTrust and 8% higher in ServiceTrust. In Threat model D, for the Non_Trust case, the percentage of failed services in the sparse network is extremely high about 98% than it is in the dense network. In short, ServiceTrust is more robust than EigenTrust in sparse or dense rating network in all scenarios.



(a)Threat model C f=0.4 (b)Threat model D type D=20

Figure 6. Percentage of failed services in sparse v.s. dense rating network

VI. CONCLUSION

We have presented ServiceTrust, a quality sensitive and attack resilient trust management facility for service provision networks. ServiceTrust offers attack resilience through three novel trust establishment techniques. First, we use multi-scale feedback rating scheme to enable providers offering high quality of services to be rewarded with high local trust values. Second, we incorporate the variances of user's rating behaviors into the local trust algorithm. Third, we exploit pairwise feedback similarity scores to weight the contributions of local trust values towards the global trust of a participant. We show that the pairwise feedback similarity weighted trust propagation can strengthen the robustness of global trust computation against malicious collectives in addition to sparse feedbacks. Experimental evaluation with independent and colluding attack models show that ServiceTrust is highly resilient to various attacks and highly effective compared to existing state of art trust models, such as EigenTrust.

ACKNOWLEDGEMENT

This work is partially sponsored by grants from NSF CISE NetSE program and SaTC program, Intel ISTC on Cloud Computing, National Nature Science foundation of China under grant No.s: 61272173, Liaoning Nature Science fund under grant no.:201102038 and Basic Research fund for higher universities (DUT12ZD104). The first author conducted this work as a visiting PhD student at school of Computer Science in Georgia Tech, funded by China Scholarship Council.

REFERENCES

[1] T. Beth, M. Borcherding, and B. Klein. Valuation of trust in open networks. In *Proc. 3rd European Symposium on Research in Computer Security – ESORICS '94*, pages 3–18, 1994.

[2] Catherine Dwyer, Starr R.Hiltz, Katia Passerini. Trust and privacy concern within social networking sites: A comparison of Facebook and MySpace. In *Proceedings of the Thirteenth Americas Conference on Information Systems.* 2007.

[3] X.X. Fan, L. Liu, M.Ch. Li, Zh.Y. Su. EigenTrust^{++}: Attack Resilient Trust Management. In Proceeding of 8th IEEE International Conference on Collaborative Computing: Networking, Applications and Work sharing, page

[4] J. Golbeck and J. Hendler. Inferring binary trust relationships in Web-based social networks. *ACM Transactions on Internet Technology, ACM,* 6(4):497-529,2006.

[5] Jennifer Golbeck. Trust and nuanced profile similarity in online social networks. *ACM Transactions on the Web (TWEB),* 3(4):1-33,2009.

[6] S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651.ACM, 2003.

[7] A. Y. Ng, A. Zheng and Jordan. Link analysis, eigenvectors and stability. *In Proceedings of 17th International Joint conference on Artificial Intelligence.* page 903-910, 2001

[8] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara. Reputation Systems. *Communications of the ACM,* 43(12):45–48, 2000.

[9] Amir Seyedi, Rachid Saadi and Valérie Issarny. Proximity-Based Trust Inference for Mobile Social Networking. *In Proceedings of 5th IFIP WG 11.11 International Conference on Trust Management, vol.358: 253-264,2011.*

[10] S. Song, K. Hwang, R. Zhou, and Y.K. Kwok. Trusted p2p transactions with fuzzy reputation aggregation. *Internet Computing, IEEE,* 9(6):24–34, 2005.

[11] L. Xiong and L. Liu. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. IEEE Transactions on Knowledge and Data Engineering, 16(7):843–857, 2004.

[12] H.B. Zhang, Y. Wang and X. Zh. Zhang. Transaction Similarity-Based Contextual Trust Evaluation in E-Commerce and E-Service Environments. In *Proceedings of IEEE international conference on Web Services(ICWS),* page 500-507,2011.