Reliable and Resilient Trust Management in Distributed Service Provision Networks

ZHIYUAN SU, State Key Laboratory of High-end Server & Storage Technology, Dalian University of Technology, Georgia Institute of Technology
LING LIU, Georgia Institute of Technology
MINGCHU LI, Dalian University of Technology
XINXIN FAN, Dalian University of Technology
YANG ZHOU, Georgia Institute of Technology

Abstract: Distributed service networks are popular platforms for service providers to offer services to consumers and for service consumers to acquire services from unknown parties. eBay and Amazon are two well-known examples of enabling and hosting such service networks to connect service providers to service consumers. Trust management is a critical component for scaling such distributed service networks to a large and growing number of participants. In this paper, we present ServiceTrust^{++}, a feedback quality sensitive and attack resilient trust management scheme for empowering distributed service networks with effective trust management capability. Comparing with existing trust models, ServiceTrust^{++} has several novel features. First, we present six attack models to capture both independent and colluding attacks with malicious cliques, malicious spies and malicious camouflages. Second, we aggregate the feedback ratings based on the variances of participants' feedback behaviors and incorporate feedback similarity as weight into the local trust algorithm. Third, we compute the global trust of a participant by employing conditional trust propagation based on feedback similarity threshold. This allows ServiceTrust^{++} to control and prevent malicious spies and malicious camouflage peers to boost their global trust scores by manipulating the feedback ratings of good peers and by taking advantage of the uniform trust propagation. Finally, we systematically combine trust decaying strategy with threshold-value based conditional trust propagation to further strengthen the robustness of our global trust computation against sophisticated malicious feedbacks. Experimental evaluation with both simulation-based networks and real network dataset Epinion show that ServiceTrust^{++} is highly resilient against all six attack models and highly effective compared to EigenTrust, the most popular and representative trust propagation model to date.

Keywords: Trust, Feedback rating quality, Reliability, Attack resilience, Distributed service network.

1. INTRODUCTION

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 the service providers. This ranking is based primarily on consumers' feedback ratings obtained through their prior transaction experiences with the service providers. With the increased popularity of social media, such social rankings are often used 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 unique opportunities for consumers to be connected to unfamiliar service providers and for providers to reach out to a large and growing customer base, the opportunity to interact with unknown providers or consumers also opens doors for potential risks of dishonest ratings and malicious manipulations. eBay, Amazon and numerous other service provisioning network systems have demonstrated that incorporating trust management can be an effective way to improve the trustworthiness of their system. Many service providers consider feedback based trust as one of the most important measures for sharing information and developing new consumer-provider relationships. Reputation based

Author's addresses: Zhiyuan Su, State Key Laboratory of High-end Server & Storage Technology, Jinan, Shandong, China, email:suzhiyuan2006@gmail.com; MingChu Li, Xinxin Fan, Software school, Dalian University of Technology, Liaoning, China; Ling Liu, Yang Zhou, 266 Ferst drive, Atlanta GA, USA.

trust often refers to trust management in a large network of participants where the trust of a participant is computed based on the feedback ratings that this participant receives from the rest of the participants.

Trust management has attracted active research in several areas of computer science, ranging from e-Commerce [Xiong, et al. 2004] and e-Government [Nepal, et al. 2014], mobile networks, peer to peer networks, sensor networks, to social networks and applications, to name a few [Dwyer, et al. 2007, Jøsang, et al. 2007, Resnick, et al. 2000, Sherchan, et al. 2013]. Interestingly, most of the trust models proposed to date are based on per-transaction feedbacks. 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, which this provider has received from the consumers in the service provision network. However, existing trust 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 are against dishonest feedbacks and malicious manipulations, and (iii) how trust is computed in the presence of sparse feedbacks and cold start (zero feedbacks). 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 in anticipation of malicious or dishonest feedbacks, and whether a proposed trust model will be effective in the presence of some known attacks. We believe that answers to these questions are critical for building a reliable and resilient trust management for service provision networks.

These observations motivate us to develop ServiceTrust^{++}, a feedback quality aware and attack resilient trust management model and a suite of algorithms, for improving the effectiveness and robustness of distributed service networks. We aim to address the above questions from several perspectives: (i) Trust takes time to build but can be dropped or destroyed drastically due to malicious manipulations; (ii) Trust is based on direct and indirect experiences (circle of friends) but not symmetric and not transitive; (iii) Feedback ratings used in building trust can be vulnerable to strategically malicious manipulations; and (iv) Two players with low mutual trust are unlikely to interact with one another. We develop ServiceTrust^{++} with a number of novel features for establishing and managing trust in a distributed service network with strong attack resilience:

- First, we introduce six attack models to cover the most common forms of attacks in trust and social rating systems, including the four well-know basic attacks (independently malicious, malicious collective, malicious collective with camouflage, malicious spies), and the two strategically malicious attacks (Malicious spies combined with camouflage, and Malicious spies combined with both camouflage and malicious collectives). We show that even the existing trust models that are robust again the four basic attack models to date are vulnerable to the strategically malicious attacks.

- Second, ServiceTrust^{++} employs two novel techniques to improve the quality and robustness of local trust computation: (i) We use a quality-sensitive variance based rating aggregation method to increase the quality-based sensitivity of feedback ratings. (ii) We employ pairwise feedback similarity to weight the trust from one peer to another. We show that these two techniques encourage finer granularity of quality differentiation, which can increase the level of resilience for ServiceTrust++ to against colluding attacks on both feedback rating manipulation and local trust manipulation.
- Third, ServiceTrust^{++} is the first to identify the vulnerability of uniform trust propagation and the detrimental effects through analyzing the strategically malicious attacks: Malicious spies combined with camouflage and Malicious spies combined with both camouflage and malicious collectives. ServiceTrust^{++} develops a conditional trust propagation kernel, which computes the global trust of a participant by employing the feedback similarity controlled trust propagation. significantly enhances the attack resilience of trust propagation and global trust computation against dishonest feedbacks and malicious manipulation of feedback ratings. This optimization empowers ServiceTrust^{++} to effectively handle sparse feedbacks and cold start problems, and more importantly to discredit those strategically malicious participants, and to control the amount of trust propagating from a good participant to a malicious participant and vice versa.
- Fourth but not the least, we further improve the attack resilience of ServiceTrust++ by strengthening the threshold-based conditional trust propagation with a set of configurable performance and quality tuning techniques, such as tunable decay factor, controlled randomness and jump strategy for handling malicious cliques.

We conduct an extensive experimental evaluation to show the effectiveness and efficiency of ServiceTrust^{++}.

It is worth noting that our preliminary work in [Su, et al 2013] is limited to countering the basic four attack models by employing feedback similarity into the local trust computation. Unfortunately, as we show in this paper that the trust model in [Su, et al 2013] is severely handicapped and is vulnerable to malicious spies with camouflage or colluding collectives. This observation motivates us to develop ServiceTrust^{++} to against the six threat models by incorporating the above-mentioned three unique features. To the best of our knowledge, ServiceTrust^{++} is the first trust model that is resilient to all six attack-models (see Section 2.2). In addition, ServiceTrust^{++} is the first to promote the conditional trust propagation over the uniform trust propagation, and combined it with decaying strategy, for reliable and resilient trust management in distributed service networks.

The rest of paper is organized as follows. Section 2 discussed the overview and problem statement of the ServiceTrust^{++} development. Section 3 presents the core components of ServiceTrust^{++}. We report our experimental evaluation results in Section 4. Section 5 describes the related work and Section 6 concludes the paper.

2. OVERVIEW AND PROBLEM STATEMENT

In this section, we first briefly describe the fundamentals of reputation and trust, such as how feedback ratings are collected, how to compute local trust based on feedback ratings, and how to compute global trust by utilizing transitive trust relationship. Then we describe six different attack models and analyze why existing trust models are vulnerable in the presence of both dishonest feedbacks and

strategically malicious participants. EigenTrust [Kamvar, et al. 2003] is chosen as the reference trust model in the discussion because EigenTrust is the most cited trust model in the literature and its Eigen vector based trust propagation and its attack analysis remain to be the most representative in the context of trust and reputation management.

2.1 Reference Trust Model

In a service 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 in a peer to peer network. Thus we refer to a member of the service network by either "participant" or "peer" in the rest of the paper.

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*.

2.1.1 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 [Kamvar, et al. 2003], when 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. $sat(i, j) = \sum_{tr(i,j)>0} tr(i, j)$ and $unsat(i, j) = \sum_{tr(i,j)<0} tr(i, j)$. 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, when $sat(i, j) > unsat(i, j)$, $s(i, j)$ is a positive integer, indicating that there are more positive feedback ratings from peer i to peer j; and $s(i, j)$ is negative otherwise.

We compute the local trust c_{ij} based on $s(i,j)$. In order to make the local trust comparison meaningful between peers with high volume of transactions and peers with low volume of transactions, we normalize the local trust values into the range of [0,1. 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:

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

In Formula (1), P denotes a set of pre-trusted seed participants (pre-trusted peers) and the pre-trust value for each peer in P is $1/P$. By default, pre-trusted peers are considered as the central authority in the network [Kamvar, et al. 2003] and are used as the seed peers to bootstrap the system initially and to allow new comers to be linked to some existing participants in the network. When $\sum_j \max(s(i,j),0) = 0$, it indicates two cases: (i) peer i just joins the network as a newcomer and has not had transactions with anyone; or (ii) peer i has not received services from and rated

anyone in the system though it may have provided services to others and may have been rated by others, i.e., $\sum_j \max(s(j,i),0) \neq 0$. When $s(i,j) \leq 0$, it implies that either there is no feedback rating from i to j or the feedback ratings from i to j are dominated by negative ratings. Thus, the local trust from i to j is zero, i.e., $c_{ij} = 0$. Only when $s(i,j) > 0$, peer i has positive and non-zero local trust value for peer j. We can model this local trust relationship among every pair of the n participants as a trust network graph with n vertices and every edge in this directed graph represents a local trust relationship from participant i to participant j. When the size of the network (n) is large, this trust network often becomes sparse since it is common that every participant will transact with only a small portion of the participants in a large scale network.

2.1.2 Computing global trust by trust propagation kernel

A popular way to compute global trust for a participant i is to aggregate all the local trust values that participant i has received from other participants [Resnick, et al. 2000, Xiong, et al. 2004]. We call this global trust computation model the *one-hop aggregation* of local trust values. Indeed, local trust values provide valuable reference for trust enabled selection of service providers when the local trust network graph is dense. Unfortunately, in real world, most of the operational service networks (e.g., eBay, amazon) are often very large in size and most of the participants only have transactions with a few other participants, resulting in a very sparse local trust network graph. This sparseness causes two types of problems: (i) the one-hop local trust aggregation may no longer produce an accurate global trust of a participant and can easily be manipulated by malicious collectives; and (ii) for each service request, the number of trusted providers is often quite small, making the trust-guided service selection less effective.

The trust propagation based global trust computation model, introduced initially in [Kamvar, et al. 2003], is designed to cope with such sparse rating problem. It aims at computing a global trust for each of the n participants based on the transaction experiences collected from the entire network. Concretely, assume that in a local trust network graph, peer i has a direct edge to peer j and peer j has a direct edge to peer k. Even though peer i does not have any prior direct experience with peer k, we can compute the trust value of peer i over peer k by asking the opinion of its neighbors who have local trust value over peer k, such as peer j. This process can be iteratively extended to the m-hop circle of friends (neighbors of neighbors in the m-hop range) ($1 \leq m < n$) in m iterations to produce a global trust for each participant in the service network. Typically m is determined by the convergence condition of the propagation model, such as the global trust values for a large percentage of peers are similar within a small threshold with as a system-supplied parameter.

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 initialize the global trust values for all participants based on a uniform probability distribution over all n peers and set $t^0_i = 1/n$. Let t^k_i denote the global trust value of peer i from its connected participants within k-hop circle in the service network. Thus, the one hop global trust value of peer i can be computed by normalized aggregation of its local trust values from other directly connected peers, namely $t^1_i = \sum_{j=1}^{n} c_{ji} \Big/ n = \sum_{j=1}^{n} c_{ji} t^0_i$. Let $\vec{t}^{\,k} = (t^k_1, ..., t^k_i, ..., t^k_n)$ denote the global trust vector of

size n at k^{th} iteration ($1 \le k<n$). The general formula for computing the global trust vector at the ($k+1$) iteration is:

$$\vec{t}^{k+1} = C^T \vec{t}^{k} \tag{2}$$

Formula (2) can be expanded in the matrix format as follows:

$$\begin{bmatrix} t_1^{k+1} \\ \vdots \\ t_i^{k+1} \\ \vdots \\ t_n^{k+1} \end{bmatrix} = \begin{bmatrix} c_{11} & \cdots & c_{k1} & \cdots & c_{n1} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ c_{1k} & \cdots & c_{kk} & \cdots & c_{nk} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ c_{1n} & \cdots & c_{kn} & \cdots & c_{nn} \end{bmatrix} \begin{bmatrix} t_1^{k} \\ \vdots \\ t_i^{k} \\ \vdots \\ t_n^{k} \end{bmatrix}$$

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. For each element of this global trust vector, t_i, it quantifies how much trust the system collectively places on the participant i.

To address the cold start problem in terms of new comers or participants with no feedback ratings from other participants, we utilize a set P of pre-trusted seed peers as the bootstrap peers. A participant can always trust one of the pre-trust peers with a probability of $1/|P|$. Thus, alternative to initializing the global trust vector using uniform distribution over all n participants, $\vec{t}^{0} = (1/n,...,1/n)$, we can also initialize the global trust vector $\vec{t}^{0}$ by a uniform distribution $\vec{p}$ over the set of pre-trust peers, namely each peer initializes its trust vector by placing its initial trust of $1/|P|$ on only the pre-trusted peers. We will study the effect of these two alternative trust initialization methods in terms of attack resilience and convergence rate in Section 3.4 and Section 4.

One potential weakness of Formula (2) is the vulnerability for chained colluding attacks in which a group of colluding peers give one another positive feedbacks to gain the global trust value from the network. A common way to break such chained malicious collectives is by having each participant, at some random probability, placing its trust on pre-trust seed peers that are definitely not a part of the collectives instead. Formula (3) below computes the global trust vector of size n at $(k+1)^{th}$ round of iterations ($1 \le k<n$) by taking into account of malicious collectives:

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

This formula implies that peer i's global trust at the $(k+1)^{th}$ 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

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

We use the damping factor a to allow a participant with some probability to trust only some pre-trusted peers instead of establishing trust only through the participants with whom it has direct links in the service network. This damping factor introduces some controlled randomness in trust establishment such that the trust benefit for a participant j by another participant i through direct linkage in the service network is to some extent reduced.

One instance of this reference trust model is the EigenTrust model [Kamvar, et al. 2003], in which α is set to be 0.1 in their experiment. However, there is no study on how different settings of α may influence the performance of a trust model and what other factors may impact on the effectiveness of a trust model. In this paper we argue that a number of factors are critical for trust establishment through a trust

propagation kernel (see Section 3 and Section 4). In the subsequent two sections, we first present the six attack models that are common in social and trust based rating systems and then analyze the inherent vulnerabilities of this basic reference trust model.

2.2 Threat models

In decentralized service provision networks, the following threat models are commonly used to characterize malicious behaviors of different forms.

Thread 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 collective by giving a high local trust value, 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 (type D and type A). 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 A) who only provide bad (inauthentic) services when selected as service providers but they do not form the malicious chain, i.e., they are independently malicious.

Threat model E (Malicious spies with camouflage)

Malicious participants are also strategically organized into two groups (type D and type B). Compared with type D peers in Threat model D, type D peers also provide honest feedbacks to good peers at $m\%$ after finishing the transaction. In addition, type B peers not only provide bad (inauthentic) services but also form a chain of malicious collective as described in Threat model B.

Threat model F (Malicious spies with camouflage and collective)

Malicious participants are strategically organized into two groups (type D and type B). Compared with the type D peers in Threat model E, type D peers in threat model F will provide honest feedbacks to good peers with $m\%$ and form a malicious

collective among all type D peers. The type B peers behave the same as they do in Threat model E.

There are other types of threat models, such as multiple independent or colluding malicious collectives, which can be represented as a generalized form of the five 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 corresponds 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 five threat models. Thus we argue that a highly effective trust model for a service network should be resilient to all of the above five threat models..

2.3 Trust based service selection schemes

We discuss the following four commonly adopted 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 selects 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% [Kamvar, et al. 2003].

2.4 Vulnerabilities in the Reference Model

Although EigenTrust by design has incorporated several decisions to increase its attack resilience, which is also one of the main factors for its huge popularity and citation in the literature, EigenTrust has reported [Kamvar, et al. 2003] some inherent vulnerabilities as shown in Figure 1.

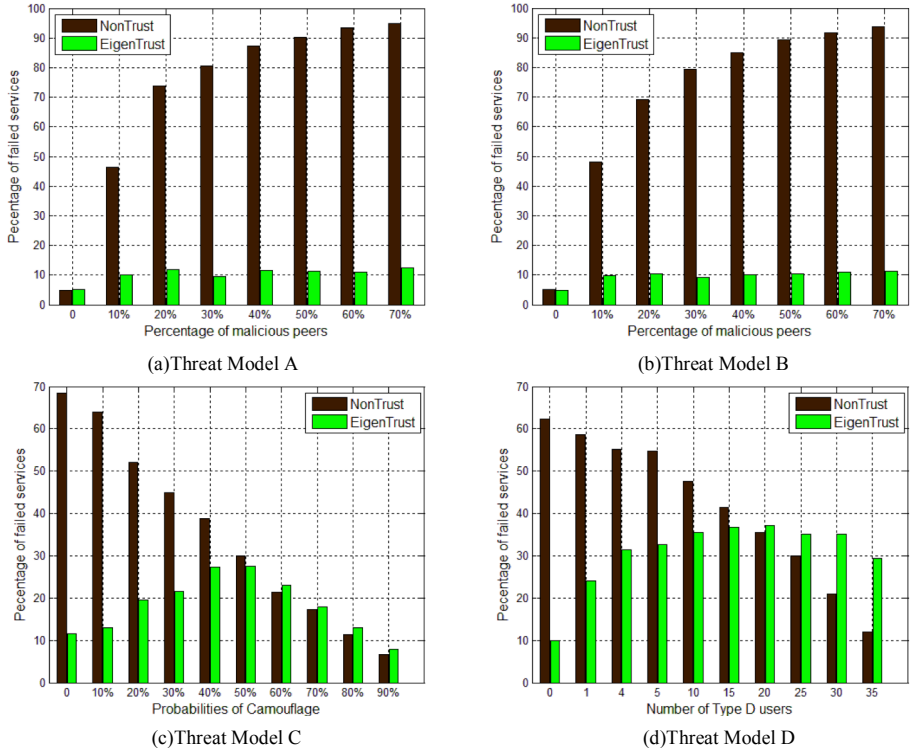


Fig.1. Attack resilience of EigenTrust model

In this Figure, 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. We produced these experiments used the identical setup as reported in [Kamvar, et al. 2003]. 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.

Next, 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 (i.e., Threat models C and D).

First, the feedback aggregation formula (1) is 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, Formula (3) for computing global trust vector uses a weighted trust propagation model where the local trust values 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 peers are acting as spy by providing good services and accumulate high global trust values. Then type D peers utilize their high global trust values to boost the global trusts of all type A malicious peers in the collective. This can easily subvert the system as the number of spy peers increases, as shown in Figure 1(d). Figure 2 shows the global trust values of all participants under Threat model D. 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.

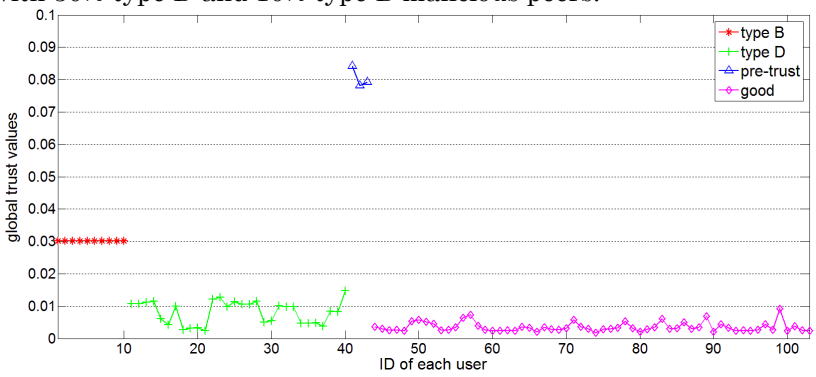


Fig.2. Global trust value of each participant with 60% good peers and 40% malicious peers (30% spy + 10% type B in threat model D).

The analysis of the above vulnerabilities motivates us to carefully revisit all core components of the reference trust model, including the transaction rating model, the local trust computation, the uniform trust propagating kernel, and to develop the ServiceTrust^{++} model for providing feedback quality aware and attack resilient trust management in large scale service network systems.

3. ATTACK RESILIENT TRUST MANAGEMENT IN SERVICETRUST^{++}

The design of ServiceTrust^{++} embodies two innovative techniques: (i) we develop a feedback similarity weighted local trust computation algorithm to constrain the trust manipulation by malicious collectives; and (ii) we develop a conditional global trust computation algorithm by combining pairwise feedback similarity with decaying rate, damping strategy and initialization strategy to maintain reliable and resilient trust propagation.

3.1 Feedback Similarity Weighted Local Trust Computation

Recall the vulnerability analysis in Section 2, the feedback aggregation scheme used in EigenTrust suffers from a number of inherent vulnerabilities. To address the detrimental effect of those vulnerabilities, the design of ServiceTrust^{++} needs to meet a number of objectives.

First, we advocate the multi-scale rating scheme over the binary rating scheme. By replacing binary rating with multi-scale rating in ServiceTrust^{++}, we can use fine-grained and quality sensitive metrics to differentiate dishonest ratings by malicious participants from feedback ratings by non-malicious participants by introducing feedback similarity measure.

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

Second, 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 Thread model D or camouflage peers in Threat model C. This implies that we need to provide capability of identifying such malicious behaviors in ServiceTrust^{++}.

Third, a peer who wants to get a good local trust value from other peers must provide consistently good services. Also peers who always get high scale ratings, such as 5 or 4 score in the multi-scale rating scheme (5) should have higher local trust values than those who always receive positive but low feedback ratings of 1 or 2.

In addition, malicious behavior, be it a bad service or dishonest feedback rating, should be punished in ServiceTrust^{++}. One way to meet this objective is to make the trust of a peer hard to build but allow the trust of a peer to drop dramatically once detected being malicious.

3.1.1 With these design goals in mind, we propose two new algorithms, one for aggregation of feedback ratings and the other for location trust computation. *Variance based Rating Aggregations*

The local trust of peer i for peer j is computed primarily based on the aggregate of the feedback ratings that peer i has given to peer j. Instead of using a simple algorithm to perform the aggregation, we propose to aggregate the multi-scale ratings by incorporating the rating variance. Concretely, let J denote the set of participants that i gives rating to and let $\max(tr(i,J))$ denote the highest rating that peer i gives to others in the network. We define $s(i, j)$ as follows:

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

In Formula (6), $v(i, j)$ denotes the variance of all the ratings that i gives j. 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:

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

In Formula (7), we assume that there are L transactions in one spectacular interval. $tr_l(i, j)$ is the lth 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:

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

3.1.2 Computing Local Trust based on Rating Similarity

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:

$$\begin{aligned} pos\_sim(i,j) &= \begin{cases} 1 - \sqrt{\dfrac{\sum_{k \in pos\_comm(i,j)} (\mu(i,k) - \mu(j,k))^2}{|pos\_comm(i,j)|}} & |pos\_comm| > 0 \\ 0 & otherwise \end{cases} \\ \mu(i,k) &= \sum_{m=1}^{|R(i,k)|} tr_m(i,k) / (|R(i,k)| \times \max(S)) \\ \mu(j,k) &= \sum_{m=1}^{|R(j,k)|} tr_m(j,k) / (|R(j,k)| \times \max(S)) \end{aligned} \tag{9}$$

$R(i,k)$ 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. For each peer i in the network of n participants, we refer to the highest rating that peer i has received as the maximum rating of peer i. Thus, *max_mean(S)* can be computed by taking the mean from the n maximum ratings from the n participants.

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

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

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.

$$sim(i,j) = w_n \times neg\_sim(i,j) + w_p \times pos\_sim(i,j) \tag{11}$$

In the experiments reported in this paper, w_p and w_n are set to 0.5 respectively. 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:

$$sc_{ij} = c_{ij} \times (sim(i,j) / \sum_{k \in R(i)} sim(i,k)) \tag{12}$$

By utilizing pairwise feedback similarity, a peer i can assign higher weight to its local trust on those neighboring peers with similar feedback behavior and lower weight to its local trust on those peers with very dissimilar rating behavior.

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

$$l_{ij} = \begin{cases} \dfrac{\max(sc_{ij}, 0)}{\sum_{k \in R(i)} \max(sc_{ik}, 0)} \\ 0 \end{cases} \tag{13}$$

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:

$$t_i^{k+1} = (1-\alpha)(l_{1i}t_1^{k} + l_{2i}t_2^{k} + ... + l_{ni}t_n^{k}) + \alpha p_i \tag{14}$$

By combining feedback similarity and local trust to weight the amount of trust propagation from one participant to another in the trust network, we can constrain the effect of bad mouth from malicious peers on the trust of good participants via trust propagation. Similarly, we can limit the amount of trust from good peers to be propagated to malicious peers when their feedback similarity score is low.

3.2 Global Trust Computation by Controlled Propagation

Most of the trust models that utilize the propagating kernel to compute global trust are based on the flooding model with uniform distribution. Propagation with the uniform distribution treats all neighboring peers equally and fails to distinguish those strategically malicious peers from good peers. Even though, by introducing feedback similarity based local trust, a peer can decrease its local trust on those with dissimilar rating behavior, it cannot prevent or stop the detrimental effect of the positive ratings generated by malicious peers acting as spies or camouflage. Moreover, by using trust propagation with uniform distribution in global trust computation, the benefit of feedback similarity weighted local trust is further decreased because malicious peers acting as spies or camouflage can continue to gain higher global trust by collecting local trust values from good peers. For example, our experiments in Section 4.5 show that trust propagation with uniform distribution are vulnerable under Threat model E and Threat model F even for the trust model in [Su, et al. 2013] where feedback similarity is utilized in local trust computation.

We argue in this paper that ServiceTrust^{++} should develop the conditional trust propagation kernel instead of using uniform distribution based trust propagation. In

this section, we describe a number of technical strategies used in ServiceTrust++ to control the trust propagation from several dimensions, including the similarity threshold controlled trust propagation, decaying factor, controlled jump strategy for breaking the malicious clique.

3.2.1 Conditional Trust Propagation based on Rating Similarity Threshold

Following the uniform trust propagation model, if there exists a directed link from participant i to participant j in the trust network, i would always propagates certain amount of its global trust value to j no matter how weak this trust relationship is. Most of the existing works, even those in recent years [Kuter, et al. 2010, Guha, et al. 2004, Ortega, et al. 2012, Su et al. 2013], are based on the uniform trust propagation model for addressing the rating sparsity problems and malicious collective problems in trust and reputation management. However, we argue that trust and reputation in real world never follow the uniform propagation model. Intuitively, if the trust relationship between two participants i and j is too weak, participant i usually has the choice of not considering this weak trust relationship and thus stop to propagate its global trust value to j until its trust relationship with j improves. In ServiceTrust++, we introduce active and inactive state for each participant in the trust propagation network and set j to be in an inactive state with respect to participant i when their pairwise feedback similarity value on the directed link, which represents the strength of the trust relationship from i to j, is lower than a specified threshold value θ. i could propagate its global trust value to j only when the strength of trust relationship from i to j is higher than the threshold value θ, and j is in an active state with respect to j. We call this model of trust propagation the *similarity threshold based conditional trust propagation*.

We formulized the conditional trust propagation process as follows.

$$\vec{t}^{\,k+1} = (1-a)L^T\vec{t}^{\,k} + a\vec{p}$$
$$t_i^{k+1} = (1-\alpha)(l_{1i}t_1^k + l_{2i}t_2^k + ... + l_{ni}t_n^k) + \alpha p_i \quad (15)$$
$$l_{ji} = \begin{cases} l_{ji} & \text{if } \mathrm{sim}_{ji} > \theta \\ 0 & otherwise \end{cases} \quad j \in [1,2,\cdots,n]\ \theta \in (0,1)$$

Formula (15) makes the following statements:

(1) If the threshold value θ is 0 and the decay factor is not turned on, the conditional propagation described in Formula 15 becomes the uniform propagation. If the threshold value θ is 1, the trust propagation process could not be executed and no participant would propagate any trust value to others in the network. This is not desirable for a reputation system. That is why we set $0<\theta<1$.

(2) We need to carefully select the appropriate threshold value in the conditional trust propagation process. If the threshold value θ is too low, it cannot prevent the trust values being propagated from good participants to malicious ones. If the threshold value θ is too high, there might be the danger that too many trust propagation paths from good ones to good ones are cut out. This may lead to the trust propagation process to be collapsed or meaningless.

In the first prototype of ServiceTrust++, we recommend settings of θ between 0.1 to 0.5, such as 0.1, 0.2, 0.3, 0.4, 0.5. As we have analyzed, smaller θ implies more propagation paths and faster convergence rate, and higher θ implies fewer propagation paths and slower convergence rate. We conduct a series of experiments in Section 4 to show the impact of θ values on attack resilience and propagation rate of ServiceTrust++.

3.2.2 Decay factors

One serious threat to existing trust propagation models is the malicious clique. On one hand, malicious cliques can manipulate the trust propagation and unfairly bring down the global trust scores of good peers. On the other hand, the existence of malicious cliques can also affect the convergence speed of trust propagation based global trust computation.

We argue that although employing pairwise feedback similarity to weight the local trust and threshold-based conditional propagation to control the global trust computation during the trust propagation are effective against all six attacks, when the number of strategically malicious peers (type D peers) becomes large and dominating in numbers, it will be harder to break the malicious cliques by simply relying on the rating similarity based techniques, which rely on the majority voting principle. This motivates us to incorporate the decay factor with tunable decay rates to the trust propagation kernel and the global trust computation algorithm to further improve the attack resilience of ServiceTrust^{++}.

Concretely, let $d \in (0,1]$ be the decay factor supplied by the trust management system. By incorporating the decay factor in the trust propagation formula (15), we can compute the global trust at the $(k+1)^{th}$ iteration as follows:

$$\vec{t}^{\,k+1} = d \times (1-a) L^T \vec{t}^{\,k} + a\vec{p} \tag{16}$$

By examining closer at the matrix computation, each element t_i in the global trust vector is computed as follows:

$$t_i^{k+1} = d \times (1-\alpha)(l_{1i} t_1^k + l_{2i} t_2^k + \ldots + l_{mi} t_n^k) + \alpha p_i \tag{17}$$

We illustrate the effect of decay factor by example. By setting d=1/2 in Formula 16 and Formula 17 above, it implies that as the trust propagation continues from hop 1 to hop k, the decaying factor is changing from ½ to $(½)^2$, …, $(½)^k$, simulating the effect of continuous fading effect as the hop distance increases in each trust propagation path. In the first prototype of ServiceTrust^{++}, we choose *decay factors* ranging from 1/2, 1/3, 1/5, 1/10. Smaller d value implies faster decaying rate in the trust propagation and thus relatively lower global trust value overall. Also smaller decay factor d often leads to faster rate of convergence and better resilience to attacks. By setting d=1, the decaying effect will be zero in the propagation based global trust computation process. We conduct extensive experiments in Section 4 to analyze the impact of different settings of the decay factor d on both attack resilience and convergence speed of our global trust propagation algorithm.

3.2.3 Other Parameters involved in Propagation based Global Trust Computation

Formula 16 also shows that, in addition to decay factor d, the global trust computation in ServiceTrust^{++} also involves the damping parameter α, representing the probability of jumping out of the link-based propagation mode (hop-based circle of friends) during the trust propagation process. This damping factor involves both the setting of α value and the jumping strategy in terms of how the jump destination will be selected. Combining the damping factor α and the jump strategy offers a number of advantages in our trust model. First, it enables the ServiceTrust^{++} model to break the malicious cliques during a trust propagation process. Second, it supports the use of pre-trust peers with some given probability. Third, it also guarantees that the computation of matrix L is converging at some given probability. Another important parameter is the initialization of global trust value vector at the time step of k = 0.

Initialization Strategies

The most commonly used initialization strategy is the uniform distribution based initialization, in which every peer in the network of size n will be initialized with the trust value of $1/n$ such that the sum of the global trust values for all n participants equals to 1. In this case, the initialization vector is set to $t^0 = [1/n, \cdots, 1/n, \cdots, 1/n]$. In this paper we refer to this initialization strategy as the average initialization (***AI***).

Alternatively, one may also wonder how the pre-trust based initialization strategy works in comparison to the uniform distribution based initialization. In pre-trust based initialization, each peer, except the pre-trust peers, starts with a global trust value of zero and all the pre-trust peers in P will have a global trust value of $1/|P|$ such that the sum of all n participants equals to 1. The initialization vector is $t^0 = [t_1^0, t_2^0, ..., t_n^0]$, $t_i^0 = 1/|P|$ *if* $i \in P$ and $t_i^0 = 0$ *if* $i \notin P$. We refer to this initialization strategy as the pre-trust based initialization (***PI***). In order for the pre-trust based initialization to work, it requires a bootstrapping (warm-up) procedure in the feedback based trust system. For example, in our simulation-based experiments, prior to starting the trust propagation, pre-trust peers will send a certain number of service requests to the network in order to give other peers chance to obtain feedback ratings and global trust values. Without this warm-up process, the trust propagation will not be meaningful. Clearly, the PI strategy can cause extra burden for pre-trust peers.

We observe through experiments on both simulation-based and real dataset (Epinions) that both initialization strategies deliver the similar effect in terms of robustness of trust against attacks, assuming the same jump probability α to propagate one's global trust to a randomly selected participant in the network.

3.2.4 Jump Strategies

The *jump strategies* are related to the decision of where in the network the global trust propagation should be jump to given a damping probability α There are two commonly used jump strategies: *uniformly random jump* or *controlled random jump*. The uniform random jump strategy implies that everyone in the network has an equal probability to be chosen as the jump destination. In contrast, the controlled random jump strategy constrains to a subset of peers as the candidate destinations of a random jump. For instance, in ServiceTrust^{++}, we could use pre-trust peers as the subset of participants that serve as the destinations for the jump strategy. Thus, each of the pre-trust peers has the equal probability to be chosen as a jump destination. Alternatively we could also choose *top K* most trusted peers as the candidate destinations of the controlled jump strategy. Clearly, by restricting jump to only a subset of peers, be it pre-trusted peers or *top K* trusted peers, we provide better resilience to attacks compared to uniformly random jump strategy. Note that when $K > |P|$, top K based jump strategy may converge faster, but if a malicious peer obtained a sufficiently high global trust value to get into the top K peers, then the trust system will no longer be resilient to malicious cliques in Threat models C, D, E. Thus, in ServiceTrust^{++}, we use the pre-trust based jump strategy as the default.

Figure 3 gives a small trust network of 15 participants (peers). We use this example network to illustrate how different initialization strategies and different jump strategies may influence the global trust values of each peer in ServiceTrust^{++} and also in EigenTrust for the sake of comparison. In this example network, the local trust value on each edge is assigned by the *zipf* distribution and we allow some trust links from good users to malicious users. 30% of malicious users are included in this example network, which forms a malicious clique.

We first compare the time complexity of different combinations of initialization and jump strategies under EigenTrust and ServiceTrust^{++} using this example network. Figure 4 shows that different initialization and jump combinations do not affect the convergence time of EigenTrust but for ServiceTrust^{++}, the controlled random jump to only pre-trusted peers consumes more iteration rounds to reach the convergence. This is primarily because the global trust in ServiceTrust^{++} is computed iteratively and the computation in each iteration will compute the pairwise rating similarity, leading to more number of iterations to reach the convergence condition.

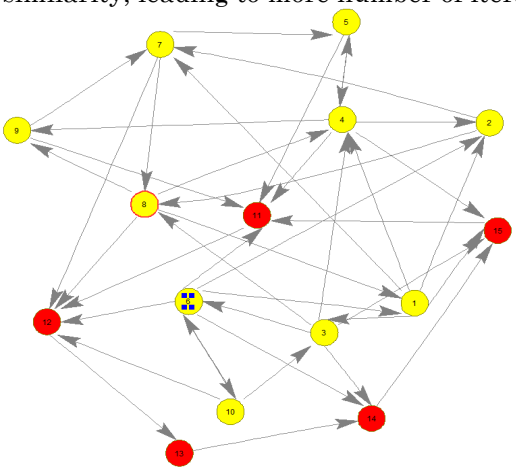


Fig.3. An example trust network topology under Threat model C

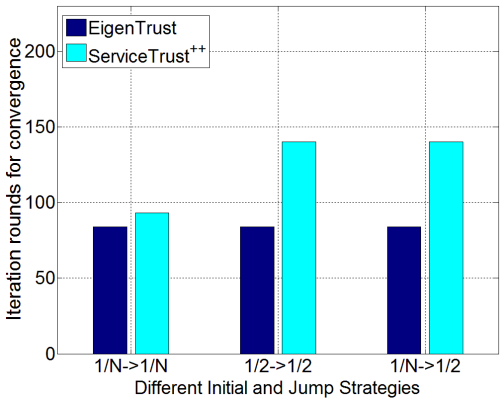


Fig.4. Time complexity of different initial and jump strategies

Figure 5 and Figure 6 show the global trust values for the 15 users after the global trust vector converges in EigenTrust and ServiceTrust^{++} respectively. We observe that the choice of the initialization vector does not influence the global trust values but the choice of jump strategies does for both trust models. If the uniformly random jump strategy is chosen, we can see that malicious users (id 11-15) obtain much higher global trust values in both EigenTrust and ServiceTrust^{++}. This demonstrates that the controlled random jump strategies, as expected, could provide better attack resilience.

Last but not the least, we would like to show how different α may impact on the attack resilience in Threat model C. Figure 7 shows that with the pre-trust initial and pre-trust jump strategy, the higher α is, the lower the global trust values of malicious users are. However, with higher α value, the high global trust values are concentrated in pre-trust users. This may overload the pre-trust users. Since α mostly aims to break the malicious collective, in our experiments, we set α value typically between 0.1 and 0.25 with default value of 0.1.

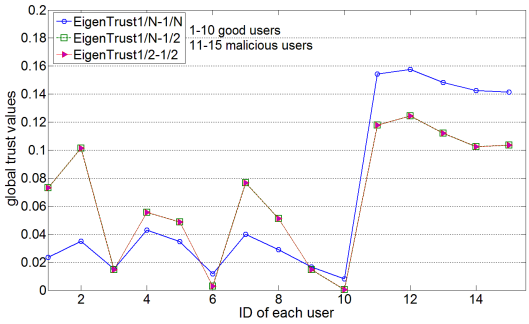


Fig.5. Impact of initialization and jump strategies in EigenTrust

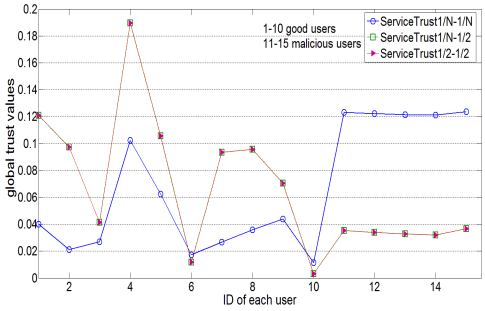


Fig.6. Impact of initialization and jump strategies in ServiceTrust++

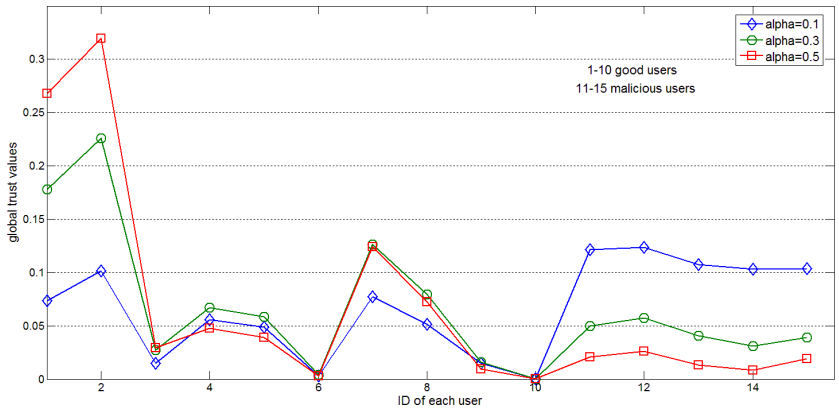


Fig.7. Global trust values of 15 peers in with different α values

Although the initialization strategy is not essential to the trust propagation models, the jump strategy chosen for a given trust model is critical in terms of its effectiveness in breaking the malicious cliques.

4. EXPERIMENTAL EVALUATION

In this section we evaluate the performance of ServiceTrust++ in terms of both attack resilience and service quality. We compare ServiceTrust++ with EigenTrust and ServiceTrust in a service network system without the support of trust under the six attack models. We evaluate the effectiveness of ServiceTrust++ in terms of attack resilience, convergence rate, time complexity by varying the sizes of the service networks, varying the decaying factors and propagation control threshold values, and by different ways of combining jump strategies and initialization strategies. We evaluate the effectiveness of ServiceTrust++ by comparing with two existing representative trust models for a number of reasons. First, EigenTrust is the No. 1 most cited trust model in the literature of trust and reputation management. Second, although there are many existing trust models in the literature, none to date except ServiceTrust [Su, et al 2013] has shown to prevail over EigenTrust [Kamvar, et al. 2003] in terms of attack resilience against the four attack models used by EigenTrust. Third, ServiceTrust in [Su, et al 2013] is a result of our early research on attack resilient trust management. Even though it improves over EigenTrust on the attack model C (malicious collectives with camouflage but no spies) and attack model D (malicious spies without camouflage), but it fails to handle more sophisticated attack models, such as malicious spies with camouflage or malicious collectives with camouflage and spies, like Threat models E and F. We show analytically and experimentally in this paper that there are a couple of reasons that these two

representative earlier trust models perform poorly against complex attack models. First, although the use of Eigen vector based trust propagation remains to be the leading method for managing trust in large networked systems with rating sparsity, both EigenTrust and ServiceTrust directly employ eigenvector based uniform trust propagation with no consideration of any form of propagation control. Second, the controlled propagation in ServiceTrust^{++} should combine feedback similarity threshold based conditional trust propagation with decaying strategy and jumping strategy to conquer the inherent vulnerabilities due to uncontrolled uniform trust propagation.

We organize the experimental results into three parts: (1) Evaluating attack resilience of ServiceTrust^{++} under all six attack-models in Section 4.2 and Section 4.3; (2) Evaluating the scalability and time complexity of ServiceTrust^{++} from two perspectives: varying the scale and the number of participants of the service networks in Section 4.4 and studying different parameter settings and their impact on the effectiveness of ServiceTrust^{++} in Section 4.5; and (3) Evaluate the performance of ServiceTrust^{++} using Epinions, a real dataset from http://snap.stanford.edu/data/soc-Epinions1.html. We show that ServiceTrust^{++} significantly outperforms EigenTrust and ServiceTrust with high attack resilience.

4.1 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 a peer will be broadcasted hop by hop within the radius of the chosen hop-count horizon of the network, say 7 hop-horizons.

Nodes in each service network consist 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 the popularity of files follows *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 the 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 a satisfied service. The probabilistic-based selection method is used. 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 metric used in this paper includes 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 also report the time complexity and the iteration rounds used for trust propagation. To make a fair comparison with EigenTrust and ServiceTrust, we choose the same set of configuration parameters as those used in EigenTrust [Kamvar, et al. 2003] as shown in Table I. We set pre-trusted peers to have 10 initial neighbors. To simulate the same hostile environment as in [Kamvar, et al. 2003], we also set malicious peers to have at least 10 initial neighbors and normal peers with at least 2 initial neighbors, which allows malicious peers to be connected to the highly connected peers and to other malicious peers.

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	0%-10%

In the second group of experiments, we extend the size of service networks with up to 10,000 peers. We show that the ServiceTrust^{++} schemes not only perform well in small scale networks but also perform proportionally well in large networks.

4.2 Effectiveness of ServiceTrust^{++}

In this section, we evaluate the effectiveness of ServiceTrust^{++} by comparing it with EigenTrust and ServiceTrust in terms of attack resilience. All simulations are executed under threat models A, B, C and D. For ServiceTrust^{++}, we set the decay factor $d = 0.5$ and the rating similarity threshold $\theta = 0.5$ for conditional trust propagation. The experiment results are depicted in Figure 8. We will report the evaluation result for Threat models E and F in Section 4.3.

For Threat models A and B, all three trust models 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%. This is primarily because in the simulation based experiments, we have allowed normal peers performing unintentionally bad (unsatisfactory) service through the new comer policy configuration, which allows a peer with global trust value 0 to be selected as a provider at the maximum 10% probability. This also gives the malicious peers with zero trust values at most 10% probability of being selected as a provider.

For Threat models C and D, ServiceTrust^{++} consistently performs well with only 5%-10% of failed services (malicious downloads) as the camouflage probability of

malicious users is increased from zero percentage to 90% in the network. So does ServiceTrust model due to the use of feedback similarity weighted local trust computation. However, EigenTrust starts failing in comparison when the camouflage probability *f%* reaches 10% in Threat model C. For Thread model D, under the same experimental setting as in EigenTrust, with 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),

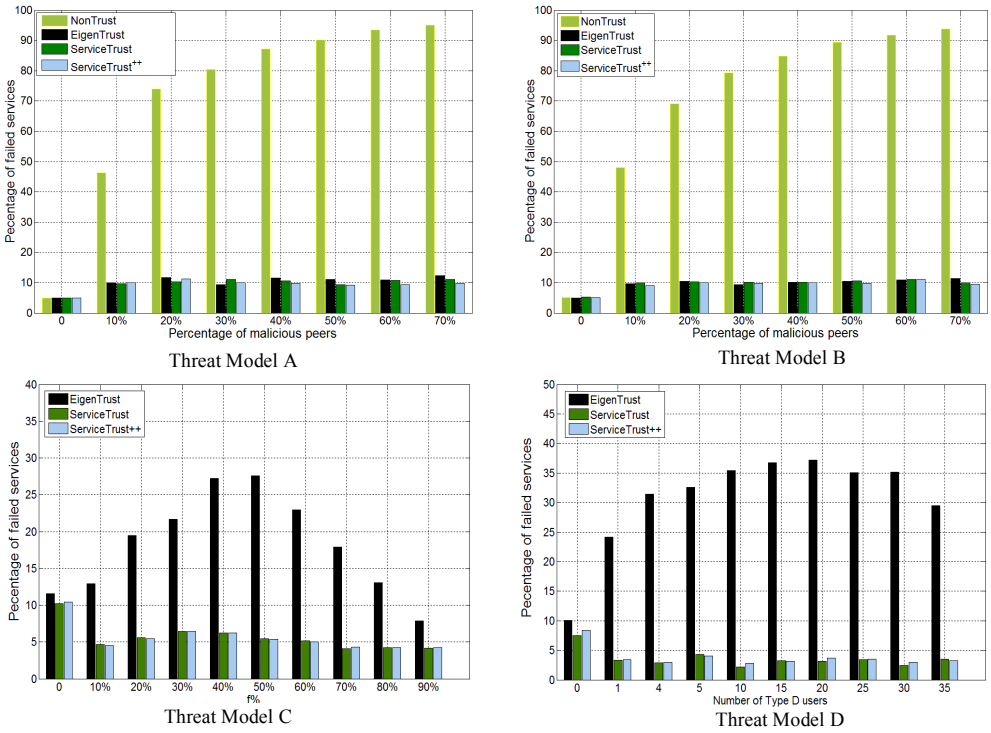
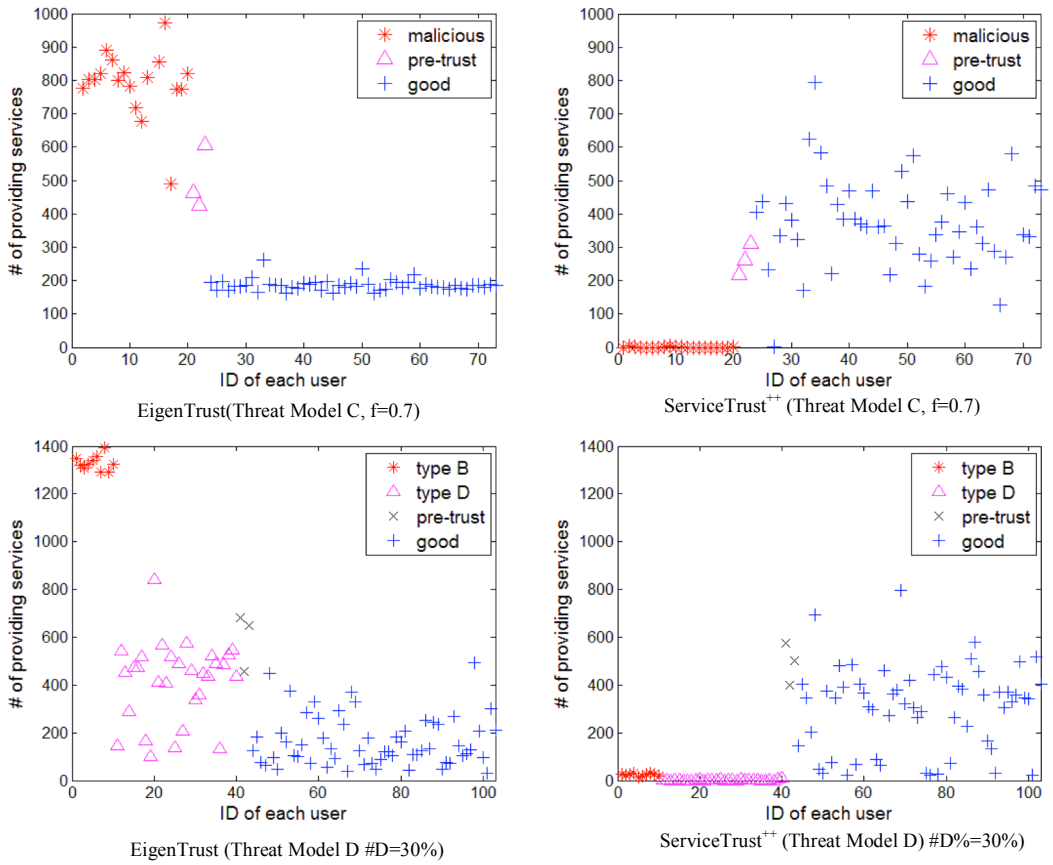


Fig.8. Fraction of failed services in threat model A, B, C and D (d=0.5, θ = 0.5 in ServiceTrust^{++}

Figure 8 shows the results measured when 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. Again ServiceTrust^{++} performs consistently better than EigenTrust in all cases. When the number type D peers (spy) reaches 50% of the malicious peers, EigenTrust is getting the highest number of failed services, which is about 38% of failed services, while ServiceTrust^{++} constantly performs well with only about 5% failed services due to the 10% probabilistic new comer selection policy. The strength of ServiceTrust^{++} against strategic malicious behaviors is attributed 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.

To further illustrate the performance difference between EigenTrust and ServiceTrust^{++} under Threat models C and D, Figure 9 provides visual display of the number of services provided by three types of participants in a service network: normal peers, pre-trust peers and malicious peers under Threat models C and D. The

setting for this set of experiments is the same as that in Figure 8. In Threat model C (camouflage probability is 70%), we see that the number of services provided by malicious peers (red stars) are almost zero in ServiceTrust^{++} but very high in EigenTrust (800 times on average). In Threat model D (# of spy users are 30), the red star and the pink triangle represent the number of services provided by type B users and type D spy peers respectively, both of which are very low in ServiceTrust^{++}, less than 50 but much higher in EigenTrust with on average 1300 for type B peers and 400 for type D peers. This shows that EigenTrust fails to defend the reputation management against the colluding behavior of type D spy peers and type B camouflage peers.



EigenTrust(Threat Model C, f=0.7)

ServiceTrust^{++} (Threat Model C, f=0.7)

EigenTrust (Threat Model D #D=30%)

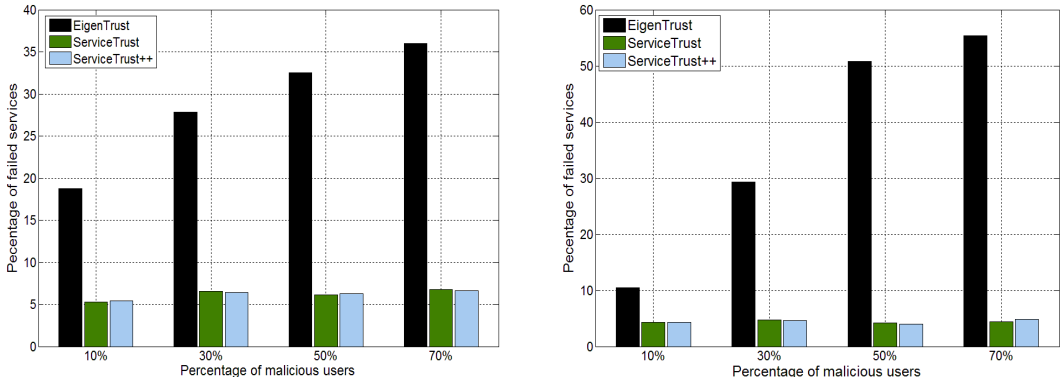
ServiceTrust^{++} (Threat Model D) #D%=30%)

Fig.9. Number of services provided by different kind of users (d=0.5, θ = 0.5 in ServiceTrust^{++})

To make a fair comparison with EigenTrust and ServiceTrust, for the set of simulations above, we take the same experiment settings in ServiceTrust^{++} as in EigenTrust and ServiceTrust. In Threat models C and D, the percentage of malicious peers is only 27% and 40%, respectively. In the following experiments, we vary the percentage of malicious peers in Threat models C and D to evaluate the performance of ServiceTrust^{++} and to compare it with EigenTrust.

For threat model C and D, we vary the percentage of malicious users from 10%, 30%, 50% and 70% to show the attack resilience of EigenTrust, ServiceTrust and ServiceTrust^{++}. The camouflage probability of malicious peers is 50% in Figure 10(a). In Figure 10(b), we split all the malicious users equally into two groups. Namely, the number of type D peers and the number of type B users are the same in Threat model

D. Figure 10 shows that ServiceTrust^{++} consistently performs well even when the percentage of malicious peers reaches to 70%. But the attack resilience of EigenTrust declines linearly with the increasing percentage of malicious users. In Threat model C, when the percentage of malicious peers reaches 70%, there are about 37% of failed services in EigenTrust. Similarly, for Threat model D, the percentage of failed services reaches to 55% in EigenTrust, while ServiceTrust^{++} remains to have very low percentage of failed services.



(a)Threat Model C with f%=50% (b)Threat Model D with number of spy users =20

Fig.10. Fraction of failed services in threat model C and D with different percentage of malicious users (d=0.5, θ = 0.5 in ServiceTrust^{++})

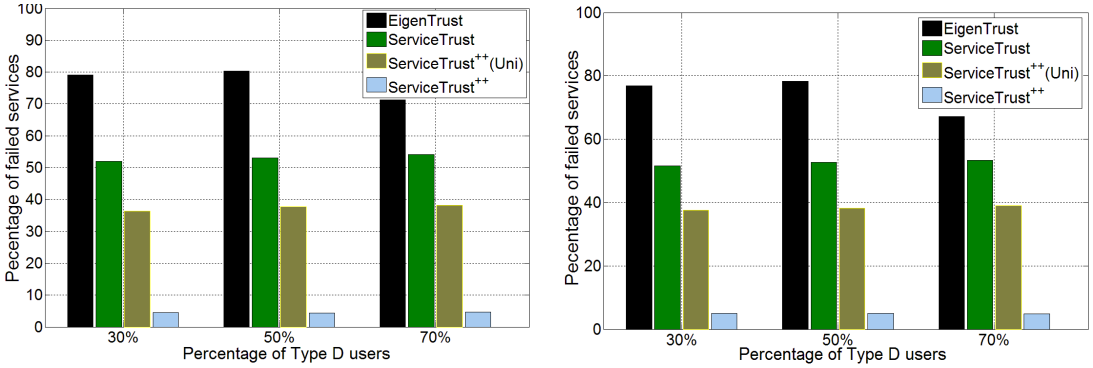
The overall design ServiceTrust^{++} is quite different from that of ServiceTrust in [Su, et al. 2013]. ServiceTrust^{++} advocates two novel and significant features: using conditional trust propagation kernel with feedback similarity based threshold and combining with decaying strategy and jumping strategy. These innovative techniques enable ServiceTrust^{++} to be highly resilient and effective against all six attack models.

We include ServiceTrust as the third model in this set of experimental comparison to show that for attacks under the categories of Threat models C and D, only relying on the feedback similarity weighted local trust computation together with the rating variance to aggregate individual transaction ratings is sufficient. However, relying on uniform trust propagation without exercising any form of controlled trust propagation, ServiceTrust will suffer from inherent vulnerabilities under complex attacks like Threat models E and F, as we will show in the next section.

4.3 ServiceTrust^{++} under Threat Model E and Threat model F

In this section, we evaluate the effectiveness of ServiceTrust^{++} in terms of attack resilience under Threat models E and F, which are the two most sophisticated threat models. To demonstrate the superiority of ServiceTrust^{++} design as a whole, we compare the performance of ServiceTrust^{++} with EigenTrust, ServiceTrust in [Su, et al. 2013] and ServiceTrust^{++}(Uni) (with only decay and jump strategy incorporated but keeping uniform trust propagation kernel). In Threat model E, malicious type D users try to gain some feedback similarity with good peers by providing m% honest feedbacks to good users and malicious type B peers form a malicious chain to boost each other's global trust value. In Threat model F, except the malicious behaviors depicted in Threat model E, the type D peers also form a malicious chain to boost each other's trust. The experiment results are shown in Figure 11 with 20% percentage of type D users and 20% type B users. Figure 11(a) shows the attack

resilience under varying camouflage m%, with decay factor d=1/2 in Threat model E. Figure 11(b) shows the attack resilience under varying camouflage m%, with decay factor d=1/2. For both Figure 11(a) and (b), the threshold for ServiceTrust^{++} is 0.5.



(a) Threat model E (b) Threat model F

Fig.11. the attack resilience under Threat model E and Threat model F (d=1/2, θ =0.5)

From Fig.11, ServiceTrust^{++} has very low percentage of failed services and is the smallest, about 5%, compared with almost 40% failed services in ServiceTrust^{++}(Uni), slightly over 50% failed services in ServiceTrust, and about 80% failed services for all cases in EigenTrust. We would like to make two important observations. First, although uniform trust propagation kernel is used in both ServiceTrust and ServiceTrust^{++}(Uni), by simply incorporating the combination of decaying strategy and jumping strategy in ServiceTrust^{++}(Uni), we can weaken the effect of malicious chain to some extent, enabling ServiceTrust^{++}(Uni) to have 10% less failed services. Second and more importantly, by introducing rating similarity threshold based conditional trust propagation kernel combined with decaying strategy, ServiceTrust^{++} can successfully control and prevent the global trust values propagated from good peers to malicious peers, leading to high attack resilience of ServiceTrust^{++} against all six attack models. In fact, the difference in attack resilience between ServiceTrust^{++} and EigenTrust will be more pronounced when the size of the service network is bigger and the trust network gets sparser.

Similar observations are found in Figure 11(b) under Threat model F. Only EigenTrust under Threat model F performs slightly better, about 2%, than under Threat model E. This is because the malicious chain between type D users could help improve the global trust values of good peers to which type D participants always provide good/authentic ratings.

From Figure 11 (a) and (b), we also observe that except EigenTrust, all the rest three trust models are not sensitive to the changes of m% from 30%, 50% to 70%. For EigenTrust, the percentage of failed services is as high as 80% when the m% is less than 50%, but when the m% is 70%, the percentage of failed services in EigenTrust declines to about 70%. In comparison, for ServiceTrust and ServiceTrust^{++}(Uni), the percentage of failed services does not change too much. This is because in EigenTrust, the good peers could also obtain some positive global trust values from type D peers with the increasing percentage of honest feedbacks of type D participants, which boost good peers' global trust values to some extent. In contrast, due to the variance based rating aggregation and rating similarity weighted local trust computation used in ServiceTrust^{++} and ServiceTrust, we prevent good users obtain high positive global trust values from type D participants even type D users provide honest feedbacks with a high probability.

Figure 12 illustrates why ServiceTrust^{++}(Uni) could significantly outperform ServiceTrust naïve model with d=1 (no decaying strategy) under Threat model E compared with ServiceTrust^{++}(Uni), where the percentage of type D and type B malicious peers are both 20% and the decay factor d=1/2. We observe from Figure 12, type B malicious peers in ServiceTrust^{++}(Uni) have much lower global trust values than those in ServiceTrust. Also the global trust values of malicious type B peers (id 1-20) in ServiceTrust are about 0.02, which is much higher than all the other non-pre-trust peers. In contrast, the global trust values of type B peers (*id 1-20*) in ServiceTrust^{++} are only about 0.01, which is lower than or same as good peers.

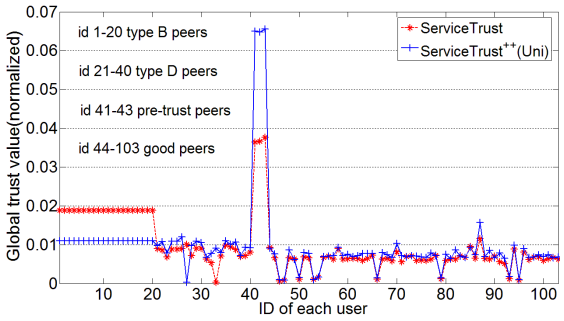


Fig.12. Global trust value of each user in ServiceTrust and ServiceTrust^{++}(Uni) under Threat model E

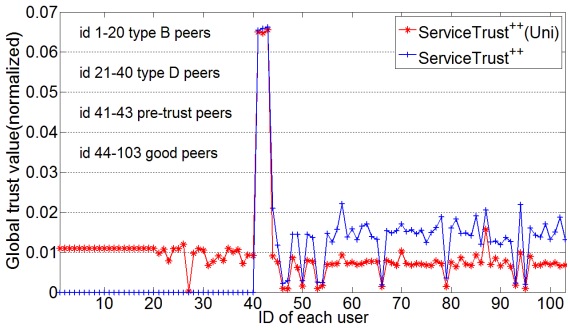


Fig.13. Global trust value of each user in ServiceTrust^{++}(Uni) and ServiceTrust^{++} under Threat model E (d=1/2, $\theta = 0.5$)

For type D peers and normal good peers, there is no significant difference between global trust values in both ServiceTrust and ServiceTrust^{++}(Uni). However, pre-trust peers in ServiceTrust^{++}(Uni) have much higher global trust values than those in ServiceTrust. We also observe that incorporating the decay factor does not improve the global trust values of normal peers but it will prevent or constrain the global trust values to be propagated to type B malicious peers from pre-trust peers and good peers, which alleviates the detrimental effect of the malicious chain formed by type B participants.

Figure 13 illustrates why ServiceTrust^{++} performs significantly better than ServiceTrust^{++}(Uni) under Threat model E. The percentage of type D and type B malicious peers are both 20% with decay factor d=1/2 in ServiceTrust^{++}. The rating similarity threshold value is set to $\theta = 0.5$. From Figure 13, we observe type B malicious peers in ServiceTrust^{++} could hardly get any positive global trust values but in ServiceTrust^{++}(Uni), their global trust values are about 0.01. But the global trust values of normal good participants in ServiceTrust^{++} are much higher than they are in ServiceTrust^{++}(Uni) thanks to the utilization of conditional trust propagation kernel.

To make a better comprehension of how ServiceTrust^{++} prevents the global trust values propagating from good users to malicious users, we depicted the similarity matrix between good users and malicious participants of ServiceTrust^{++} under threat model E in Figure 14. We show the pair-wise similarity values between different types of participants under Threat model E for ServiceTrust^{++}. We can clearly see that almost all of the similarity values between good ones and malicious ones are below 0.5 in Figure 14(a) and most of the similarity values between good ones and good ones are above 0.5 in Figure 14(b). Therefore, if we take the threshold value of 0.5 as the threshold condition for controlled trust propagation, it will help us prevent

trust values of good peers to be propagated to malicious ones. At the same time, this conditional trust propagation will continue to enable the trust propagation process between good peers, making ServiceTrust++ an effective trust model for encouraging good service and feedback rating behavior and punishing against strategic attacks.

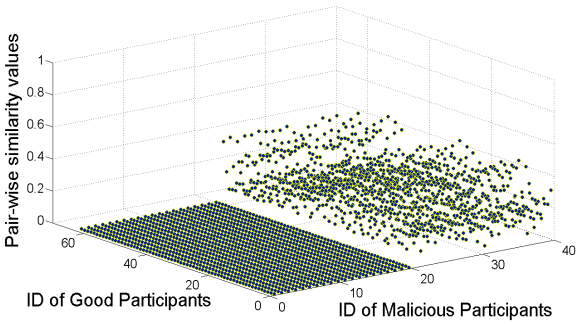


Fig.14. (a) Pair-wise similarity value between good participants and malicious participants (*d*=1/2, θ = 1/2)

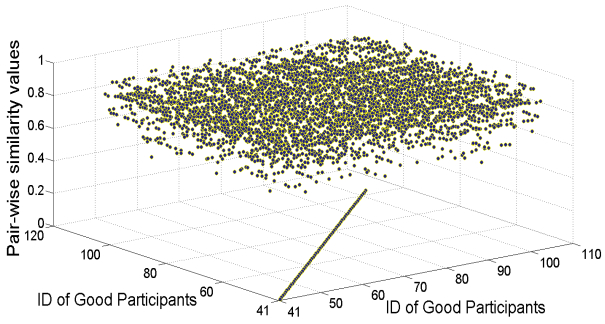


Fig.14. (b) Pair-wise similarity value between good participants and good participants (*d*=1/2, θ = *0.5)*

In order to gain a better understanding the role of decaying strategy in Eigenvalue based uniform trust propagation models, in the next set of experiments shown in Fig.15, we evaluate how different decay factors affect the performance of trust models in terms of attack resilience under Threat model E for four different trust models: EigenTrust, EigenTrust extended with a decaying strategy (EigenTrust+decay), ServiceTrust^{++}(Uni) and ServiceTrust^{++}.

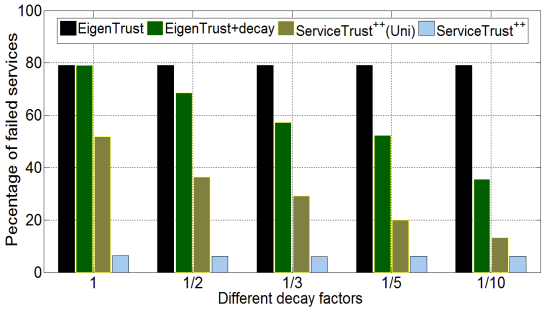


Fig.15. Attack resilience of EigenTrust and ServiceTrust^{++}(Uni) with different decay factors under threat model E (θ = 0.5)

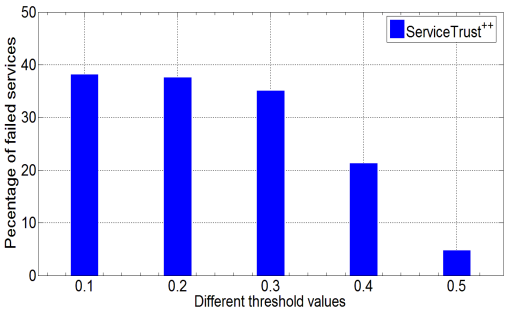


Fig.16. Attack resilience of ServiceTrust^{++} with different threshold values under threat model E (*d*=1/2)

The camouflage probability for type D users is 0.3 and there are 20% type D users and 20% type B users. The threshold value used in ServiceTrust^{++} is θ =1/2. From Fig. 15, the first interesting observation is that ServiceTrust++ with θ =0.5 threshold value delivers consistently good performance under varying decay rates from 1, 1/2, 1/3, 1/5 to 1/10 in terms of attack resilience under Threat model E. This is because with the threshold value of 1/2, the malicious chain could hardly be effective and few of type D peers could obtain global trust values from good peers. However, by observing the effect of different decaying rates for EigenTrust+decay and ServiceTrust++(Uni), we can clearly see that smaller decay factor can significantly reduces the percentage of failed services for both EigenTrust+decay and ServiceTrust++(Uni) and making the two models more attack resilient than otherwise.

In all the experiments on ServiceTrust++ reported in this section, we set the default rating similarity threshold value as 0.5 for conditional trust propagation unless explicitly stated otherwise. In the set of experiments shown in Fig. 16, we will show how different threshold values affect the attack resilience of ServiceTrust++. The decay factor is set to d=1/2, and the camouflage probability for type D users is 0.7. There are 20% type D participants and 20% type B participants. From Fig.16, we observe that the percentage of failed services in ServiceTrust++ is around 35% when conditional trust propagation is based on the rating similarity threshold value of 0.3 and the percentage of failed services is even higher when $\theta = 0.1,\ 0.2$. This shows that with a low threshold value, ServiceTrust++ could not effectively prevent the malicious type D peers from using spies and camouflage to obtaining high global trust values from good peers. Only when the θ value is set to 0.4, we observe a sharp reduction in the percentage of failed services and the reduction continues when we increase the θ value from 0.4 to 0.5. We can illustrate this result by recalling Figure 14, where we can see that most of the pair-wise similarities between good peers and malicious type D peers are above 0.3 and below 0.5.

4.4 Time Complexity and Impact of Network Scale

In order to reflect the best results given in [Kamvar, et al. 2003] on EigenTrust, in all previous experiments, we use the same size of networks as those in [Kamvar, et al. 2003] to provide a fair comparison with the best of EigenTrust. In this section, we would like to point out that the network density in EigenTrust is fairly dense with the degree of 10 for malicious peers for the small-scale networks. In this section we evaluate ServiceTrust++ by comparing with EigenTrust using varying sizes of service networks, ranging from 100, 500, 1000, 5000 to 10,000 participants, all with node degree on average of 5.

Figure 17 shows that when the scale of network is 100, only 4 iteration rounds are needed to reach to all other peers from every peer through a trust propagation kernel. However, we can only reach 14% and 7% of all peers in the network within 4 iteration rounds when the scales of network are 5,000 and 10,000 respectively. As the size of the service network increases, the number of hops we need in order to reach other nodes in the network through the hop-based query broadcast mechanism also increases. Even with the very sparse networks, ServiceTrust++ can finish the hop-based process in 7 iteration rounds (hops) 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.

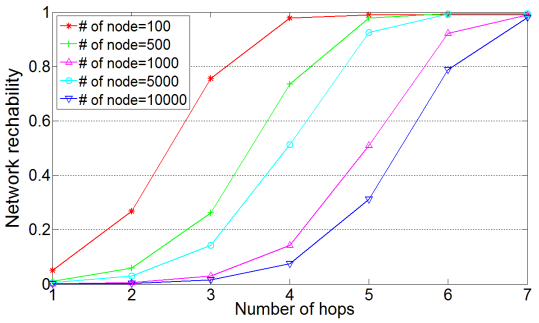


Fig.17. the ratio of reached peers in different iteration rounds with different network scales

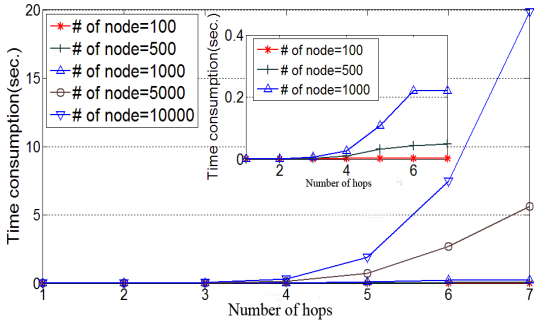


Fig.18. Time complexity of reachability for varying network size

Figure 18 shows the time complexity of hop-based broadcasting in ServiceTrust++. As the network size increases from 100 to 10,000, the time needed to reach to all

participants will increase. Compared with the network of size 10,000, the time complexity for the network 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 10,000 participants. From Figure 18, we also observe that when the number of participants reaches to 10000, it takes much longer time for a node to reach every other node in the network. However, when the number of nodes in the network is 10000, with only 6 hops we can reach to 80% of other nodes in the network, which are sufficient in terms of the number of participants to be involved in the global trust computation of a given peer. The time complexity for iterations of 6 hops is only 7 seconds, very reasonable and acceptable runtime, considering that we use a personal laptop with i5 CPU 2.5GHZ for the experiments. the time complexity will be much lower if the experiments were run on a much faster commercial server with higher CPU and memory capacity.

Figure 19 shows the time complexity of trust propagation when the propagation process converges for ServiceTrust^{++} by comparing it with EigenTrust, ServiceTrust. For ServiceTrust^{++}, we use the decay factor as 1/2 and the threshold value as 0.5. As the network size increases from 100 to 10000, the convergence time for the trust propagation will increase for all trust models. However, when the network size increases from 1000 to 5000 and 10,000, ServiceTrust^{++} consistently shows the best performance with lowest convergence time, whereas EigenTrust takes much longer time to converge for uniform trust propagation compared to ServiceTrust. There are two important factors that have significant impact on the time complexity of trust propagation process and its convergence speed: the length of malicious collective and the size of the network. In EigenTrust, the malicious peers could obtain positive global trust values from good peers by camouflage behavior or spy behavior. Both could make the malicious chain more effective in boosting the trust scores of malicious peers. Also the longer this malicious chain is, the harder it will take for the trust propagation process to converge since the global trust values of malicious peers in this malicious clique keep increasing. We can see that when the network size reaches to 10,000 participants, the time complexity of EigenTrust to converge its global trust computation is 15.43 seconds while it takes 6 seconds and 4 seconds respectively for ServiceTrust and ServiceTrust^{++} to reach the same convergence condition.

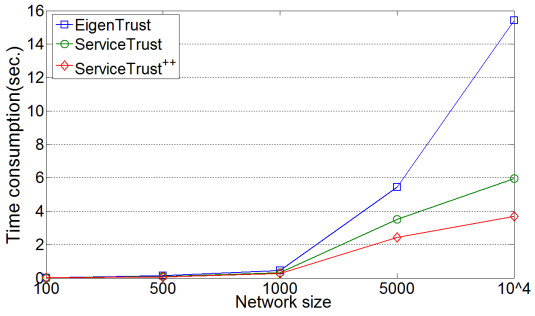


Fig.19. Time complexity of each trust model with different network scales(d=0.5, θ = 0.5 in ServiceTrust^{++})

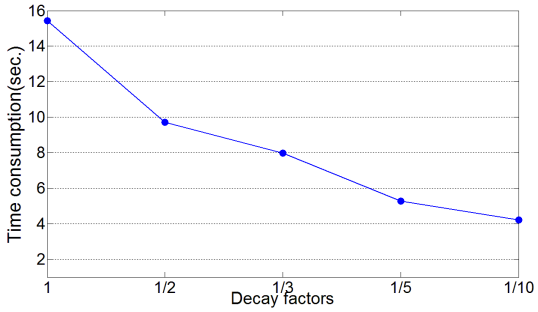


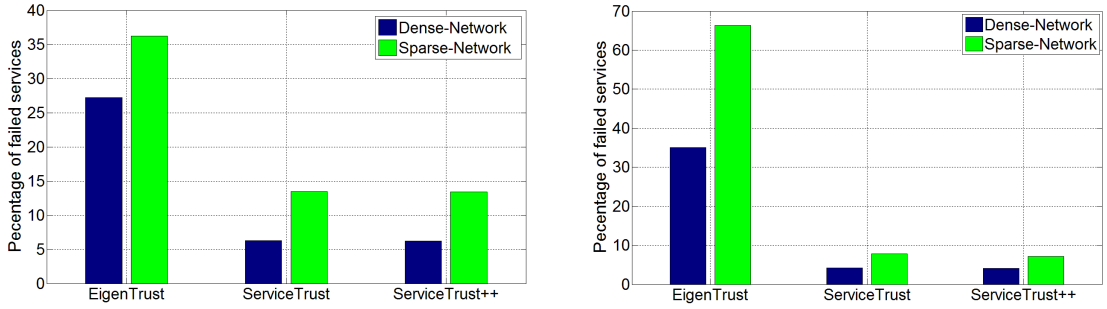
Fig.20. Time complexity of EigenTrust in 10000 nodes network with different decay factors

The result from Fig. 19 demonstrates that ServiceTrust^{++} is effective in breaking the malicious colluding cliques with spies and camouflages by exercising a suite of novel attack resilient techniques: variance based feedback aggregation, feedback similarity weighted local trust computation, conditional trust propagation by tunable

rating similarity threshold, and decaying strategy combined with jump strategy and initialization strategy. The ability to break the malicious cliques at early stage of the iterative global trust computation through Eigenvalue based propagation kernel, the faster the propagation process will converge, making ServiceTrust++ scalable to trust management in large scale service networks.

The next set of experiments shown in Fig. 20 is to study how decay factor may affect the time complexity of the uniform trust propagation process. We measure the convergence time for EigenTrust model with varying decay factors using the network of 10,000 participants. From Fig. 20, we see that the smaller the decay factor is, the faster the global trust computation will converge and the smaller the time complexity will be. When the decay factor is 1, there is no decaying strategy is employed, the time to converge is close to 16 seconds. When the decay factor is 1/10, the time complexity of EigenTrust for the network size of 10,000 participants is reduced to 4.22 seconds. This shows that, in addition to increase the attack resilience as shown in Fig. 15, the decaying factor can also significantly improve the convergence speed of the global trust computation even with uniform trust propagation kernel.

Next we evaluate the performance of ServiceTrust^{++} in sparse network and dense network respectively by comparing it with EigenTrust and ServiceTrust. Due to space constraint, we only show the experimental results under Threat model C with the camouflage probability $f\%$ set to 0.4 and Threat model D with 20 type D malicious peers. Figure 21 (a) and (b) show that ServiceTrust^{++} has the lowest percentage of failed services, ServiceTrust has slightly higher percentage of failed services for both sparse and dense networks but EigenTrust has significantly higher percentage of failed services, about 25% failed services for dense network and 36% for sparse networks, compared to 6% and 14% for ServiceTrust^{++} and 6.5% and 14.3% for ServiceTrust in dense network and sparse network respectively.



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

Fig.21. Percentage of failed services in the sparse feedback rating network under different trust models. (d=0.5, θ = 0.5 in ServiceTrust^{++})

Another observation is that for all three systems, the percentage of failed services in the sparse network is consistently higher (about 8%) than that in a dense network under both Threat models C and D. Also under Threat model D, the percentage of failed services in the sparse network is very high, about 67%, for EigenTrust, compared to 35% in the dense network. Also EigenTrust performs worse under Threat model D (67% failed services) than under Threat model C (36% failed services). In comparison, ServiceTrust^{++} performs better under Threat model D than under Threat model D for both dense and sparse networks. Namely 14% failed services under Threat model C versus only 8% failed services under Threat D for sparse networks and 6% failed services in sparse networks versus 4% failed services

in dense networks. One reason that all three trust models perform less well in sparse network than in dense network is because, compared with a dense network, a sparse network usually needs more rounds to complete the warm-up process and under the same experimental setting, trust models in sparse network could not get enough history transactions as well as similarity information. However, even with this situation, ServiceTrust^{++} shows much more attack resilience than EigenTrust.

4.5 Impact of Different Parameters

In this section, we study how different choices of decay factor, initialization strategy and jump strategy may impact on the effectiveness of ServiceTrust^{++} compared to EigenTrust. The first set of experiments evaluates how different settings of the decaying factor and different combinations of initialization strategy and jump strategy may impact on the effectiveness of both conditional trust propagation in ServiceTrust^{++} and uniform trust propagation in EigenTrust.

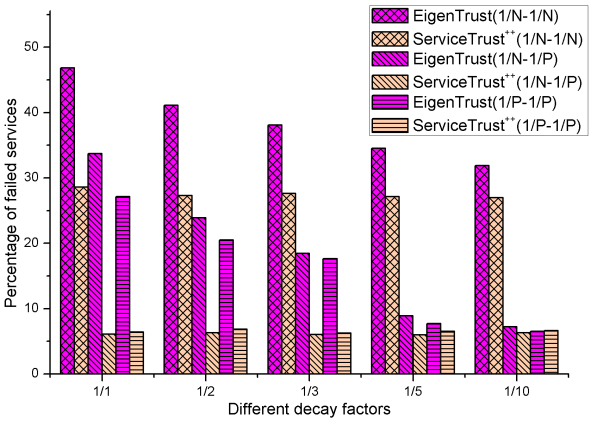


Fig.22 (a). Impact of decay factor and Initial-Jump strategies on Attack resilience with 27% malicious peers in Threat model C (θ =1/2 in ServiceTrust^{++})

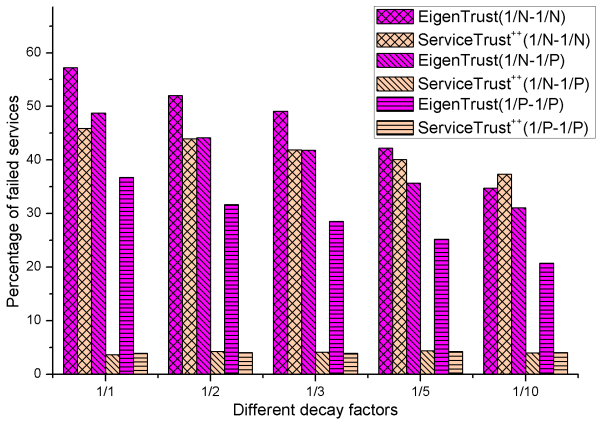


Fig.22 (b). Impact of decay factor and Initial-Jump strategies on Attack resilience with 20% type D peers in Threat model D (θ =1/2 in ServiceTrust^{++})

Figure 22 shows the impact of decay factors on attack resilience of EigenTrust and ServiceTrust^{++}. We studied the impact of varying decay factors under three combinations of initialization and jump strategies: 1/N:1/N, 1/N:1/|P|, 1/|P|:1/|P|. We omit the combination of 1/|P|:1/N because it has similar performance as 1/N:1/N in terms of attack resilience. By measuring the percentage of failed services with different settings of decay factor ranging from 1, 1/2, 1/3, 1/5 to 1/10, we observe that for both Threat model C and Threat model D, the decay factor can help EigenTrust to be more attack resilient. More interestingly, using decay factor, ServiceTrust^{++} can achieve higher attack resilience than EigenTrust with the pre-trust based controlled jump (JP) strategy, regardless whether the initialization strategy is AI (uniform initialization) or PI (pre-trust controlled initialization). However, combining the AI initialization strategy (uniform) with the uniform distribution based jump strategy helps neither ServiceTrust^{++} nor EigenTrust in terms of attack resilience.

In the next set of experiments, we study the effect of different settings of the decay factor on the convergence rate of trust propagation ServiceTrust^{++} and EigenTrust. Since we want to evaluate the effectiveness of decay factor under uniform trust propagation, we use ServiceTrust^{++}(Uni) for comparison in order to eliminate the influence of rating similarity threshold value in the trust propagation.

Figure 23 (a) and (b) show the iteration rounds of ServiceTrust^{++}(Uni) by varying the settings of the decay factor d under Threat models C and D respectively.

From Figure 23(a), we observe that the lower the decay factor is, the faster the propagation process converges under Threat model C. Also we observe that with the same decay factor, the higher the camouflage probability is, the slower the trust propagation converges. This is because malicious peers could get higher global trust values with higher camouflage probability and boost each other's trust values, which in turn slows down the speed to convergence in the global trust computation. Similarly, under Threat model D, we observe that under the same decay factor, the more type B peers are, the more iteration rounds will be needed to reach convergence. These observations are consistent with our analysis of decay factor in Section 3.3.

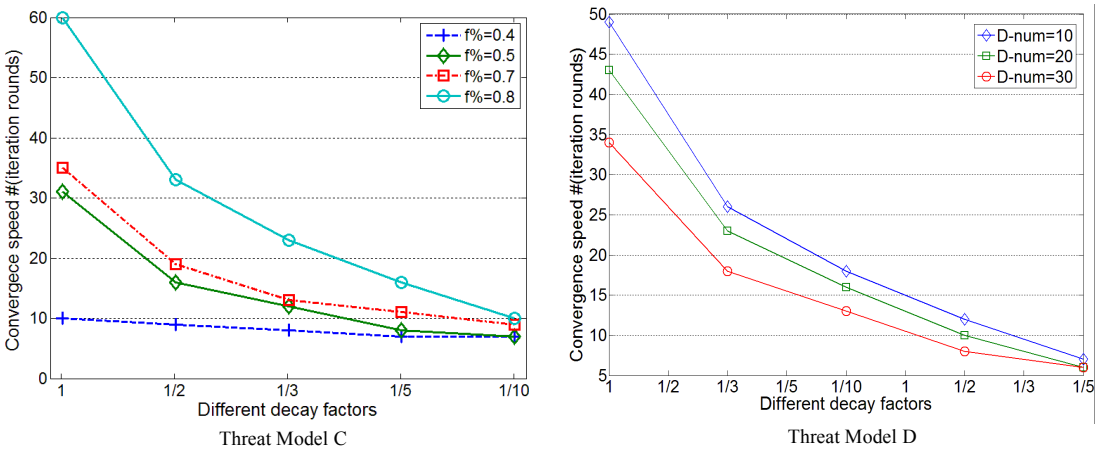


Fig.23. Convergence speed of ServiceTrust^{++}(Uni) w. different decay factors

Finally, we evaluate the impact of different ways of combining initial and jump strategies on the convergence rate of trust propagation without the impact of both decay factor and threshold value. Thus we conduct this set of experiments using ServiceTrust. Figure 24 shows the convergence speed of ServiceTrust for different combinations of initial and jump strategies under Threat model C when the camouflage probabilities are set to 0.3, 0.4, 0.5 and 0.6, and under Threat model D when the percentage of type D peers are set to 10% 20% and 30%, respectively.

From Figure 24 (a), we observe that for the combination of uniform distribution based initialization and pre-trust based random jump strategy, it will take more time to converge, compared to other combinations. This is mainly because of the existence of malicious chain. When malicious peers with high trust value jump to pre-trust peers, it will cause more rounds to reach the convergence. Though 1/N:1/N converges faster, it is less resilient to attacks (recall Figure 22). With 1/|P|:1/|P| combination, it is more attack resilient and converges fast but it needs some warm-up time to boost-trap the trust model (recall Section 3.3).

Figure 24(b) shows that without the malicious chain, different combinations of initialization and jump strategies have similar influence on the convergence speed. This result also proves that when malicious peers forming a malicious collective, the attack to a trust model, such as those under Threat models E and F, can be much more detrimental. This motivates us to develop ServiceTrust^{++}.

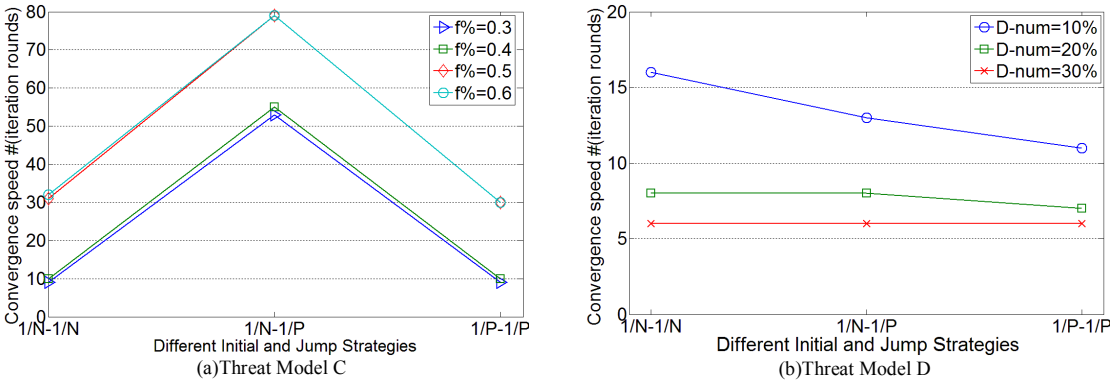


Fig.24. Convergence Speed of different combinations of initialization and jump strategies

4.6 Evaluating ServiceTrust^{++} using real dataset Epinions

Epinions is a platform for people to share their experiences, both good and bad, about a variety of topics, ranging from daily life consumptions (such as cars and coffees) to media objects (such as music and movies). Users could write reviews, rate the reviews of other authors and indicate trust or distrust for another user. The Epinions dataset (http://snap.stanford.edu/data/soc-Epinions1.html) represents a very sparse network with about 75,879 nodes. The degree distribution follows the power-law distribution. Most of the nodes only have limited trust links with other nodes. Thus, the Epinions dataset is an ideal sparse network for evaluating the effectiveness of ServiceTrust^{++}. Figure 25 shows the in-degree and out-degree distribution of Epinions dataset network.

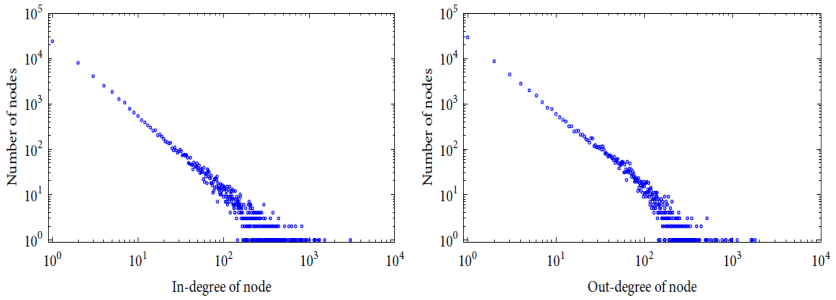


Fig.25. The degree distribution of Epinions Dataset.

In our experiments reported in this paper, we only consider the top 1000 nodes with highest degree. Among the 1000 nodes, there are 134 nodes without any in-degree, which means that they are not trusted by any of the 1000 nodes. Three nodes have no out-degree, which means that they do not trust anyone in the system composed of these 1000 nodes. The maximum out-degree for this network of 1000 nodes is 478 and the maximum in-degree is 366. The minimum out-degree and the minimum in-degree of the network are both 0. The average degree of the network is 40.24. The degree of this sample Epinions network remains to follow the power-law distribution and is a sparse network. The real pre-existing directed link from one node to another node in Epinions represents a direct trust relationship between the two nodes. In our experiments, we take the 134 nodes as the malicious nodes for launching attacks, including attempting to degrade or destroy the quality and performance of the service network system. We consider the top 30 users as our pre-trust peers whose initial global trust is greater than zero. All the rest are normal good users in the network for asking and responding to service requests. We simulate

the transactions by using the query-answering mechanism. Every node could send requests and we suppose that all 134 malicious users could answer any of the requests. Only when peer *i* has received a service provided by peer *j*, peer *i* is allowed to give *j* one feedback rating. Due to space constraint, we measure the effectiveness of ServiceTrust++ against Threat models B, C and E against malicious collectives with malicious chain, malicious camouflage and malicious spies.

Figure 26 how ServiceTrust++ performs over Epinions Dataset compared with EigenTrust and ServiceTrust. We observe the same performance (about 20% failed service requests) for all three trust models, which is consistent with the results of our simulation-based experiments. By carefully analyzing the global trust values and the topology of trust network, we see that the global trust values of all the malicious users are much lower in all three trust propagation models. But constrained by the network topology, there are several good users, which have only one other good neighbor to respond to their service requests. If the only good neighbor involuntarily provides unsatisfied services, the good users will keep trying all the other responders even they are all malicious users due to our experimental settings.

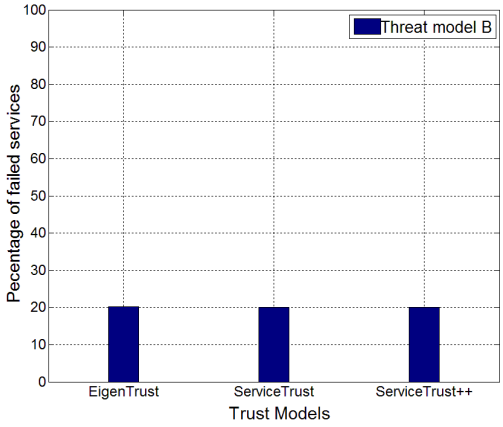


Fig. 26 Effectiveness of ServiceTrust++ under threat model B (1000 Epinions nodes, 134 malicious nodes) compared with EigenTrust and ServiceTrust

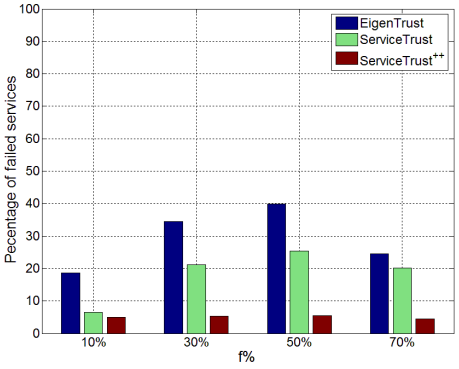


Fig. 27 Performance of ServiceTrust++ under Threat model C over Epinions dataset (1000 nodes, 134 malicious nodes with 10%, 30%, 50% and 70% camouflage probability)

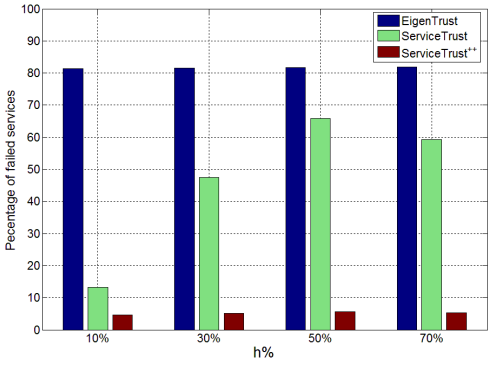


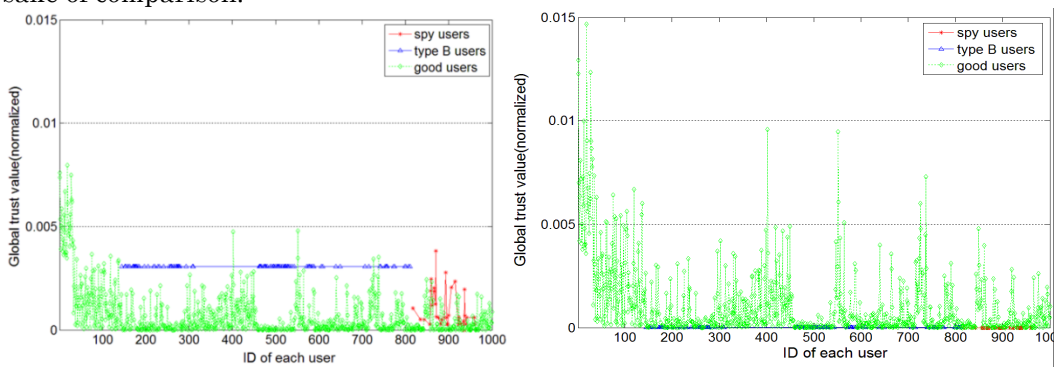
Fig. 28 Performance of ServiceTrust++ under Threat model E over Epinions dataset(1000 nodes, 134 malicious nodes with 34 type D users and 100 type B users)

Fig 27 shows that with a malicious clique formed by 134 malicious users with camouflage behavior, ServiceTrust++ significantly outperforms EigenTrust and ServiceTrust. This is because even the malicious users under Threat model C could

gain some small pair-wise similarity values with good peers, the conditional trust propagation mechanism designed in ServiceTrust++ can prevent positive global trust values being propagated from good users to malicious ones. In comparison, EigenTrust and ServiceTrust perform very poorly with significantly higher percentage of failed services. It is interesting to note that ServiceTrust performs much worse compared to the small and simulated dense network shown in Figure 8. This is due to the very sparse network for Epinions dataset and the fact that compared to the malicious users, too many good users have very small number of trusted neighbors, thus only relying on feedback similarity weighted location trust computation is no longer effective. This result once again demonstrates that ServiceTrust^{++} is highly effective in attack resilience and scalability in terms of both dense and spare networks of different sizes, thanks to the novel techniques, such as conditional trust propagation based on rating similarity threshold combined with decaying strategy and jump strategy, which prevents malicious users from boosting their global trust values from good users through camouflage.

Fig 28 shows that under Threat model E, the conditional trust propagation mechanism can successfully prevent the malicious spies or malicious type B users to boost their global trust values from good users. In comparison, both EigenTrust and ServiceTrust failed miserably under Threat model E because they cannot identify type D malicious spy users when the camouflage rating behavior is turned on with varying camouflage probability values ranging from 10%, 30%, 50% to 70%. As a result, the attack resilience of ServiceTrust under Threat model E is very poor, especially when the camouflage probability is 50%.

Fig.29 provides an illustration by plotting the statistics of global trust values under Threat model E for ServiceTrust++, with the threshold value=0.5 and the decay factor=0.5 (see the right figure) and ServiceTrust (see the left figure) for the sake of comparison.



ServiceTrust ServiceTrust^{++} with threshold value=0.5

Fig 29 Global trust value of each user in ServiceTrust and ServiceTrust++ under the attack model E on Epinions dataset (1000 nodes, 134 malicious nodes with 34 type D users and 100 type B users)

In the right figure of Figure 29, we can see that with a higher threshold value (0.5), the malicious users including spy users and malicious type B users have the lowest global trust values and thus they cannot use any high global trust values to subvert the trust management system. However, ServiceTrust model cannot effectively cut the propagation paths from good users to spy users. From the left figure in Fig. 29, both the malicious type B users and spy users get much higher global trust values under the uniform trust propagation model. This also illustrates why ServiceTrust and EigenTrust perform very poorly under Threat model E.

By evaluating ServiceTrust^{++} using Epinions dataset, we show that ServiceTrust^{++} significantly and consistently outperforms EigenTrust and

ServiceTrust in sparse and large network in terms of attack resilience, thanks to the variance-based rating aggregation, feedback similarity weighted local trust powered by the similarity threshold controlled conditional trust propagation combined with decaying strategy and jumping strategy.

5. RELATED WORK

Trust and reputation management research has attracted attentions from several areas of computer science over the last decade, ranging from eCommerce [Witkowski, et al. 2011, Liu, et al. 2011, Zhang, et al. 2011], mobile networks [Jin-Hee, et al. 2009], peer to peer networks, sensor networks to social networks and applications [Feng, et al 2012, Mobasher, et al. 2007, Albanese, et al. 2013, Lathia, et al. 2008]. An overview of some key issues in reputation management is given in [Dwyer, et al. 2007, Hoffman, et al. 2007, Resnick, et al. 2000, Sherchan, et al. 2013, Yu, et al. 2013]. Trust relationships in Web-based social networks are studied in [Caverlee, et al. 2010, Su, et al, 2013, Golbeck, et al. 2006, Page, 1998]. [Roy, et al. 2010] attempted a space efficient solution for trust management. The notion of transitive trust through trust propagation was initially presented in [Beth, et al. 1994] with trust metrics on graphs, and was made popular by [Kamvar, et al. 2003] through the use of Eigenvector based uniform propagation kernel in EigenTrust.

Trust management consists of both trust computation and maintenance, which establish and maintain trust from one participant to another, and trust evaluation, which measures and validate the effectiveness of trust modeling and trust computation in terms of attack resilience. The four basic attack models introduced by EigenTrust [Kamvar, et al. 2003] has been the main platform to date for evaluating attack resilience, though a majority of the trust models to date only evaluate their approaches against simple attacks, assuming malicious adversaries are independent and a small minority. Very few validate their approaches in terms of attack resilience to colluding collectives with different degrees of camouflage or different number of spies. The trust computation research in literature can be broadly classified into two categories: (1) Global trust computation based on feedback ratings and transactional contexts and (2) Global trust computation based on feedback ratings and trust propagation kernels.

Research in the first category primarily focused on how to aggregate feedback ratings and how to use transactional contexts to compute trust. Many trust models have been proposed based on different trust computation techniques, ranging from statistical and machine learning techniques, heuristics-based techniques, behavior-based techniques, to probabilistic-based techniques. They also differ from the set of factors used for modeling and computing trust, such as personalized trust and collective trust, time window of trust modeling, types of trust relationships, and so forth. For example, a simple solution for measuring the credibility of the feedback rating from a participant is to use its reputation value [Kamvar, et al. 2003, Zhou et al. 2007]. However, a malicious peer could easily maintain a good reputation by providing good quality services and collecting good feedback ratings from good peers, but provide dishonest feedbacks, to subvert the trust system. PeerTrust in [Xiong, et al. 2004] is the first to introduce the feedback credibility concept and to use personalized feedback similarity as the credibility of feedbacks to increase the attack resilience of dishonest feedbacks and to differentiate trusts for different types of transactional contexts. [Can, et al. 2013] explicitly distinguishes the service context and recommendation context, and uses a personalized similarity metric augmented by the fading effect in the service context for trust computing such that historical

trust values are less valuable in computing the trust of a peer than the most recent trust values. [Li, et al. 2012] introduced the attenuation function with a threshold-based mechanism to discard dishonest feedbacks in trust computation. [Wang, et al. 2012] presented a general trust model RLM. RLM considers two kinds of values in trust computation: reputation value and reputation prediction variance, which could be considered as the credibility of reputation value. Kalman feedback aggregation method and Expectation Maximization algorithm are introduced for a robust and attack resilient trust evaluation. But the consistency of the reputation prediction is affected. [Witkowski, et al. 2011] argued that the reputation-based trust models rely on the seller being long-lived and thus susceptible to whitewashing. An 'escrow mechanisms' is proposed to avoid the problems by installing a trusted intermediary, which forwards the payment to the seller only if the buyer acknowledges that the good arrived in the promised condition. However, no consideration is given to malicious feedbacks and malicious collectives that bad-mouth their competitors in the system. Recently, [Jøsang, et al. 2013] proposed to use mappings of values produced by recommender systems and from scores produced by reputation systems to subjective opinions, and combine the resulting opinions within the framework of subjective logic. However, all the trust models in this category only work well in the presence of sufficient feedbacks from a large number of peers but fail drastically when the feedback ratings are sparse or when a participant is new or has not received any rating at all. Both sparseness and cold start are well known issues in any feedback rating based trust and quality management systems. Furthermore, these trust models show serious vulnerabilities when confronting sophisticated malicious attacks or when the number of malicious peers is not small. In comparison, ServiceTrust^{++} has shown the combination of variance based aggregation of multi-scale ratings with feedback similarity weighted local trust computation offers higher attack resilience than existing approaches.

The research efforts in the second category have focused more on addressing the feedback sparseness and cold start problems by utilizing Eigenvalue based propagation kernels to compute global trust values for every participant. [Kamvar, et al. 2003] introduced EigenTrust to show Eigenvalue based uniform trust propagation can be resilient to independently malicious peers and chained malicious collectives in all cases but only resilient to malicious camouflages and malicious spies when the malicious participants are a minority. [Guha, et al. 2004] described a trust system to study how distrust is propagated. [Ortega, et al. 2012] proposed a global trust system to propagate both positive and negative opinions of the users through the feedback network. [Su et al. 2013] used the feedback similarity as a credibility to weight the local trust value of each peer combined with uniform trust propagation, which displayed more robustness against the four basic attack models compared to EigenTrust. However, most existing trust propagation models in the literature rely on the Eigenvalue based uniform propagation kernel to compute global trust with some variation on how the feedback ratings are aggregated and how to use the aggregate rating to compute the local trust for each peer. To the best of our knowledge, ServiceTrust^{++} is the first trust management system that employs pairwise feedback similarity in both aggregating the local trust assessments and managing the trust propagation for global trust computation. By introducing rating similarity based conditional trust propagation kernel to replace the uniform propagation kernel, combined with tunable decay factor and jump strategy on top of feedback similarity weighted local trust, we show that ServiceTrust++ is more robust against all six Threat models, and offers superior performance in terms of attack resilience, compared to two existing representative trust systems ServiceTrust and

EigenTrust, both use uniform trust propagation kernels and both reported good performance against the four basic attack models.

Finally, we would like to note that although utilizing feedback similarity in both local trust assessment and propagation based global trust computation is effective for reliable and resilient trust management, computing and storing the pair-wise feedback similarity for every pair in the network of n peers can be expensive in terms of both computation and space efficiency, especially for a very big n. We are interested in incorporating optimization techniques that can speed up the pairwise similarity computation and at the same time reduce the storage consumption for large scale networks.

CONCLUSION

We have presented ServiceTrust^{++}, a feedback quality aware and attack resilient trust model and trust management mechanism for decentralized service networks. ServiceTrust^{++} is unique by its three novel trust establishment techniques. First, it incorporates the variances of user's rating behaviors to aggregate feedbacks into the local trust algorithm. Second, it utilizes the pairwise feedback similarity to weight the contributions of local trust values. Third, it introduces a threshold based conditional trust propagation kernel in the global trust computation algorithm to control and prevent malicious peers to boost their global trust scores from good peers. Finally, ServiceTrust^{++} combines the threshold based conditional trust propagation with decaying strategy and jump strategy to further strengthen the effectiveness of ServiceTrust^{++} against all six attack models. Experimental evaluation using both simulation and real dataset under six attack models has shown that ServiceTrust++ is highly resilient to both independent and strategically colluding attacks and highly effective in both dense and sparse networks of varying sizes, compared to existing representative trust propagation models.

Acknowledgement. This work is partially sponsored by NSF CISE grants IIS-0905493, CNS-1115375, IIP-1230740 and a grant from Intel ISTC on Cloud Computing, National Science foundation of China under grant No. 661100194, 61272173, Graduate Creative Talents Project of DUT(DUT12ZD104, DUT13LK38，DUT12RC(3) 80). The first author performed this work while visiting DiSL at School of CS in Georgia Institute of Technology, funded by China Scholarship Council and working at State Key Laboratory of High-end Server & Storage Technology as a researcher.

REFERENCES

M. Albanese, A. d'Acierno, V. Moscato, F. Persia and A. Picariello. 2013. A Multimedia Recommender System. *ACM Trans. Int. Tech. (TOIT).* 13,1, Artical 3.

T. Beth, M. Borcherding and B. Klein. 1994. Valuation of Trust in Open Networks. 1994. *In Proceedings of 3rd European Symposium on Research in Computer Security – ESORICS '94.* Springer Berlin Heidelberg, 1-18.

A. B. Can and B. Bhargava. 2013. SORT: A Self-Organizing Trust Model for Peer-to-Peer Systems. *IEEE Transactions on dependable and sercure computing*, 10(1): 14-27.

J. Caverlee, L. Liu and S. Webb. 2010. The SocialTrust Framework for Trusted Social Information Management: Architecture and Algorithms, *Information Science*, 180,1, 95-112.

C. Dwyer, S. R. Hiltz and K. Passerini. 2007. Trust and Privacy Concern within Social Networking Sites: A Comparison of Facebook and MySpace. In *Proceedings of the 13th Americas Conference on Information Systems*. Keystone, Colorado,339-351.

Q.-Y. Feng, L. Liu and Y. F. Dai. 2012. Vulnerabilities and Countermeasures in Context-Aware Social Rating Service. *Special Issue on Context-Aware Web Services for the Future Internet, ACM Trans. on Int.Tech. (TOIT)*,11,3.

K. K. Fullam, T. B. Klos, G. Muller, J. Sabater, A. Schlosser, Z. Topol, K. S. Barber, J. S. Rosenschein, L. Vercouter, M. Voss. 2005. A Speciation of the Agent Reputation and Trust (ART) Testbed: Experimentation and Competition for Trust in Agent Societies. In *Proceedings of 4th International Joint of Conference. On Autonomous Agents and Multiagent Systems*, Utrecht, The Netherlands, 512-518.

J. Golbeck. 2009. Trust and Nuanced Profile Similarity in Online Social Networks. *ACM Trans. on the Web (TWEB),* 3,4 :1-33.

J. Golbeck and J. Hendler. 2006. Inferring Binary Trust Relationships in Web-based Social Networks. *ACM Trans. Int. Tech. (TOIT)*, ACM, 6(4):497-529.

R. Guha, R. Kumar, P. Raghavan and A. Tomkins. 2004. Propagation of trust and distrust. In *Proceedings of the 13th International Conference on World Wide Web*, New York, USA, 403-412.

C. J. Hazard and M. P. Singh. Macau: A Basis for Evaluating Reputation Systems. 2013. *In Proceedings of the 23th International Joint Conference on Artificial Intelligence (IJCAI).* Beijing, China, 1–7.

K. Hoffman, D. Zage and C. Nita-Rotaru. 2007. A Survey of Attack and Defense Techniques for Reputation Systems. ACM Computing Surveys (CSUR), ACM, 42(1):1.

A. Jøsang, G.-B. Guo, M. S. Pini, F. Santini, and Y. Xu. 2013. Combining Recommender and Reputation Systems to Produce Better Online Advice. *The 10th International Conference on Modeling Decisions for Artificial Intelligence (MDAI 2013).* Barcelona, 126-138.

A. Jøsang, R. Ismail, and C. Boyd. 2007. A Survey of Trust and Reputation Systems for Online Service Provision. *Decision Support Systems*, 43(2), pages 618-644.

S. W. Jiang, J. Zhang and Y. S. Ong. 2012. A Multiagent Evolutionary Framework Based on Trust for Multiobjective Optimization. *In Proceeding of 11th International Conference on Autonomous Agents Multiagent Systems.* Valencia, Spain. *299-306.*

C. Jin-Hee, S. Ananthram Swami and I. Chen. 2009. Towards Trust-based Cognitive Networks: A Survey of Trust Management for Mobile ad hoc Networks. In Proceedings of the *14th International command and control research and technology symposium*. Washington DC, USA.

S. D. Kamvar, M. T. Schlosser and H. Garcia-Molina. 2003. The Eigentrust Algorithm for Reputation Management in P2P Networks. In *Proceedings of the 12th international conference on World Wide Web*, ACM. 640-651.

U. Kuter, J. Golbeck. 2010. Using probabilistic confidence models for trust reference in Web-based social networks. *ACM Trans. Int. Tech. (TOIT)*, ACM, 10(2):Article No. 8.

N. Lathia, S. Hailes and L. Capra. 2008. Trust-Based Collaborative Filtering. In Proceedings of *IFIPTM 2008 - Joint Trust and PST Conferences on Privacy, Trust Management and Security*, Springer, 119–134.

S.-Y. Liu, J. Zhang, C.-Y. Miao, Y.L. Theng, and A.C. Kot. 2011. iCLUB: An Integrated Clustering-Based Approach to Improve the Robustness of Reputation Systems. I*n Proceedings of 10th International. Conference on Autonomous Agents Multiagent Systems.* Taipei, Taiwan,1151-1152.

X.-Y. Li, F. Zhou and X.-D. Yang. 2012. Scalable Feedback Aggregating (SFA) Overlay for Large-Scale P2P Trust Management. *IEEE Trsansactions on Parallel and distributed systems,* 23(10): 1944-1957.

B. Mobasher, R. Burke, R. Bhaumik and C. Williams. 2007. Toward Trustworthy Recommender Systems: An Analysis of Attack Models and Algorithm Robustness. *ACM Trans. on Int. Tech. (TOIT).* 7,4, 23.

S. Nepal, W. Sherchan and A. Bouguettaya. 2014. Service Trust Management for E-Government Applications. *Advanced Web Services*. pp: 339-362

A. Y. Ng, A. X. Zheng and M. I. Jordan. 2001. Link Analysis, Eigenvectors and Stability. *In Proceedings of 17th International Joint conference on Artificial Intelligence*. Seattle, Washington, 903-910.

F. J. Ortega, J. A. Troyano, F. L. Cruz, C. G. Vallejo and F. Enríquez. 2012. Propagation of Trust and Distrust for the Detection of Trolls in a Social Network. *Elsevier Computer Networks*. 56,12, 2884–2895.

L. Page, S. Brin, R. Motwani and T. Winograd. 1998. The PageRank Citation Ranking: Bringing Order to the Web. *Technical report, Stanford Digital Library Technologies Project*.

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

S. B. Roy, S. Amer-Yahia, A. Chawla, G. Das, C. Yu. 2010. Space Efficiency in Group Recommendation. *The VLDB Journal*, 19(6):877-900.

A. Seyedi, R. Saadi and V. Issarny. 2011. Proximity-Based Trust Inference for Mobile Social Networking. *In Proceedings of 5th IFIP WG 11.11 International Conference on Trust Management*. Copenhagen, Denmark, 253-264.

W. Sherchan, S. Nepal and C. Paris. 2013. A Survey of Trust in Social Networks. *ACM Comput. Surv.* 45, 4, Article 47, 33 pages.

Z.-Y. Su, L. Liu, M.C. Li, X. X. Fan and Y. Zhou. 2013. ServiceTrust: Trust Management in Service Provision Networks. *In Proceedings of the IEEE 10th International Conference on Services Computing (SCC 2013),* Santa Clara Marriott, CA, USA.

X. F. Wang, L. Liu and J. S. Su. 2012. RLM: A General Model for Trust Representation and Aggregation. *IEEE Transaction on Service Computing*. 5(1): 131–143.

Y. H. Wang, Y and M. P. Singh. 2010. Evidence-Based Trust: A Mathematical Model Geared for Multiagent Systems. *ACM Trans. on Aut. and Adap. Systs. (TAAS)*. 5,4,14.

Y. H. Wang and M. P. Singh. 2007. Formal Trust Model for Multiagent Systems. *In Proceedings of 20th Internatioonal Joint Conference on Artificial Intelligence(IJCAI07)*. Beijing, China,1551-1556.

J. Witkowski. 2011. Trust Mechanisms for Online Systems. In *Proceedings of 22th International Joint Conference Artificial Intelligence(AAAI'11)*. San Francisco, CA, 2866-2867.

J. Witkowski, S. Seuken and D. C. Parkes. 2011. Incentive-Compatible Escrow Mechanisms. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI'11)*.

L. Xiong and L. Liu. 2004. Peertrust: Supporting Reputation-based Trust for Peer-to-Peer Electronic Communities. *IEEE Transactions on Knowledge and Data Engineering*. 16(7):843–857, 2004.

H. Yu, Z. Q. Shen, C. Leung, C. Miao and V. R. Lesser. 2013. A Survey of Multi-Agent Trust Management Systems. *Access IEEE*, 35-50.

R. F. Zhou and K. Hwang. 2007. PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing. *IEEE Trans. Parallel and Distributed Systems*, vol. 18, no. 4, pp. 460-473.