The ISC Int'l Journal of Information Security

July 2015, Volume 7, Number 2 (pp. 151–166)

http://www.isecure-journal.org

A Collusion Mitigation Scheme for Reputation Systems

Mina Niknafs [1,*], Sadegh Dorri Nogoorani [1], and Rasool Jalili [1]

[1] *Data and Network Security Lab. (DNSL), Department of Computer Engineering, Sharif University of Technology, Azadi Ave., Tehran, I.R. Iran*

ARTICLE INFO.

Article history:
Received: 5 September 2014
Revised: 13 June 2015
Accepted: 1 December 2015
Published Online: 7 December 2015

Keywords:
Attack resistance, Collusion, Reputation, Trust.

ABSTRACT

Reputation management systems are in wide-spread use to regulate collaborations in cooperative systems. Collusion is one of the most destructive malicious behaviors in which colluders seek to affect a reputation management system in an unfair manner. Many reputation systems are vulnerable to collusion, and some model-specific mitigation methods are proposed to combat collusion. Detection of colluders is shown to be an NP-complete problem. In this paper, we propose the *Colluders Similarity Measure* (CSM) which is used by a heuristic clustering algorithm (the *Colluders Detection Algorithm* (CDA)) to detect colluders in $O(n^2m + n^4)$ in which m and n are the total number of nodes and colluders, respectively. Furthermore, we propose an architecture to implement the algorithm in a distributed manner which can be used together with compatible reputation management systems. Implementation results and comparison with other mitigation methods show that our scheme prevents colluders from unfairly increasing their reputation and decreasing the reputation of the other nodes.

1 Introduction

Reputation management in distributed systems is a new security solution for situations where there is not enough information about the members of the system. In particular, the nodes are autonomous entities interacting with each other according to their reputation information. An important challenge in these systems is the choice of trustworthy interaction partners. The reputation values are usually computed based on the opinions of the other nodes as well as the history of local satisfaction values.

Attackers aim to abuse the trust and reputation system and subvert it in order to be promoted falsely, or manipulate reputations of the other nodes. At the same time they do not want to provide a good service, and may report dishonest feedbacks (unfair rating) or recommendations. Malicious nodes may operate individually or collectively. Collusion is a crucial challenge and one of the most destructive malicious behaviors in most reputation systems. In a collusion, a group of malicious nodes which know each other, create a collective. In general, they give high ratings to the nodes in their group and low ratings to the other ones. Trust and reputation evaluation methods are designed to detect and boycott individual mis-behaving nodes. However, colluders cooperatively misrepresent themselves to converse the effects of their bad behavior and manipulate the system to gain high reputation values. In order to mitigate collusion, they should not be allowed to promote themselves easily and unfairly, and the reputation system should have a mechanism to identify and isolate them [1].

* Corresponding author.
Email addresses: m.niknafs@vru.ac.ir (M. Niknafs), dorri@ce.sharif.edu (S. Dorri Nogoorani), jalili@sharif.edu (R. Jalili)
ISSN: 2008-2045

Some mechanisms have been proposed to combat collusion, so far. For instance, the ratings received from third party nodes are weighted [2], and/or higher weights are assigned to the ratings from *pretrusted* nodes [1]. Moreover, parameters like agent similarity, feedback credibility, and different types of trust are integrated into evaluations to mitigate collusion [3]. However, these mechanisms have not been so effective. One of the deficiencies is that functional and referral trust are usually not distinguished from each other. Furthermore, the mitigations focus on individuals and do not explicitly target collections.

Detection of colluders can be modeled as finding a *clique* in a graph [4] which is an NP-complete problem. Therefore, finding colluders in a large reputation network is very time-consuming and impractical. Some case-specific approaches are also proposed, which will be investigated in Section 3.1. In this paper, we propose a practical anti-collusion scheme. We mark nodes with activities suspicious to collusion in a distributed manner. Then, all such nodes are further investigated in a central authority, and the collectives are detected by a heuristic clustering algorithm. The algorithm is based on a similarity measure specially tuned to give high values to colluders, and its runtime complexity is $O(n^2m + n^4)$ in which m and n are the total number of nodes and colluders, respectively.

1.1 Background

We briefly review the concepts and definitions in the field of reputation and trust research according to [5].

Definition 1 (Reputation). Reputation is what is generally said or believed about a person's or thing's character or standing.

Definition 2 (Trust). Trust is the subjective probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends.

The term *subjective probability* indicates that trust is a measure of belief in the $[0, 1]$ scale. There is a reciprocal relation between reputation and trust. Each node in a reputation system can base its trust evaluation on the opinions provided by third-party nodes. For instance, reputation information can be used to initialize the trust evaluation for the first time [6]. According to [5], reputation can be deemed as a united measure of trustworthiness based on the ratings or referrals from members in a group. In other words, many reputation systems calculate the reputation value of a node by aggregating the trust values received from different nodes (see [1] and [2], for example.) It is also important to distinguish between *functional* and *referral* trust [7]:

Definition 3 (Functional trust). Functional trust of A towards B is the A's opinion in B's ability to provide certain services in a trust scope.

Definition 4 (Referral trust). Referral trust of A towards B is A's opinion in B's ability to provide good recommendations in a trust scope because node B is knowledgeable in that trust scope.

While the two concepts are related, but should be differentiated from each other. This is because it is probable that a node provides a good service and at the same time, gives dishonest recommendation about the other nodes. Hence, a reputation system should distinguish functional from referral trust, in order to counteract collusion.

1.2 The Structure of the Paper

In the remainder of the paper, we describe the most common security threats that are applicable to reputation and trust systems in Section 2. Then, we review the related works in Section 3. In Section 4, we explain our scheme, and analyze its complexity from different aspects. Then, its implementation is studied in Section 5. Our simulation results are presented in Section 6 and the paper is concluded with Section 7.

2 Security Threat Scenarios

Attackers aim to misuse the system in general, have diverse goals, and use different mechanisms. The primitive goals of attacks are as follows [8]:

- **Self-promoting**: Attackers increase their own reputation through illegal actions like manipulation of trust values.
- **Slandering**: To lower the reputation of other nodes, attackers manipulate their reputation by reporting false data .
- **Subverting** the reputation system: Attackers disrupt the availability of the system and manipulate the system, in general.

At the most basic level, an attacker targets one of the above-mentioned attacks in a malicious scenario. However, in more advanced scenarios, more than one goal are followed. In essence, an attacker may provide a bad service, give dishonest feedbacks (unfair rating), or recommendations about the others. The three mechanisms can also be mixed and matched to implement the following attack scenarios [9]:

- **Individual malicious nodes:** Individual malicious nodes always provide bad service. The main goal in this scenario is to subvert the reputation system.
- **Malicious collectives:** Collusion is a union between two or more nodes to limit open competi-

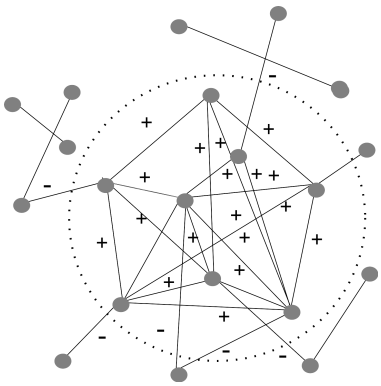

Figure 1. A collusion scenario in a trust network. The circles represent the nodes of the system, and those in the dotted region are colluders. The links indicate interactions among the nodes. The positive and negative labels indicate positive and negative feedbacks, respectively.

tion by deceiving to gain unfair benefits that colluders would not be able to gain as individuals [10]. They have the following behaviors [1, 2, 9, 11–13] (see Figure 1):

- Frequently provide bad services when they are selected as service providers, so receive low ratings form other nodes,
- Assign high trust values to their group-mates, and
- Frequently give dishonest feedbacks and recommendations about the others, indeed, they give other nodes low reputation values.

Sometimes, the colluders try to look good and hide their bad behavior, by providing high-quality services (with some probability). This behavior is called *camouflaging*. A malicious collective may be used to implement any of the self-promoting, slandering, or subverting attacks.

- **Sybil:** An adversary can introduce a substantial set of corrupt participants and control all of them. They can be used to compromise the reputation system. For example, each time one of them is selected as a service provider, it provides bad service. After that, it disappears from the system and reappears with a new identity. In the worst case, all of these participants concurrently try for a common goal. Hence, this case can also be considered to be a collusion.
- **White washing:** An attacker abuses the system using some system vulnerability to repair its reputation. In this scenario, the attacker tries to subvert the reputation system.

Since malicious collectives use all the three mechanisms, collusion is one of the most destructive scenarios in reputation systems. Although collusion mitigation mechanisms are proposed in the literature, they are either NP-complete or specific to a system and cannot be used in other systems.

3 Related Works

A statistical reliability metric is proposed in [14] to mitigate collusion. The metric is based on the distribution of transactions among the set of partners of a node. The reputation value of a node is called *reliable* if all its related transactions are distributed uniformly among many distinct partners. In a similar manner, a reputation value is not reliable if a considerable fraction of the transactions are performed with a small number of partners. Statistical analysis were proposed to assess the reliability of the reputation value of a node, and detect collusions accordingly. Das and Islam [3] propose a dynamic trust computation model for secure communication in multi-agent systems. They integrate parameters like agent similarity, feedback credibility, and different types of trust (direct, indirect, recent, and historical) into computation to mitigate collusion.

In [13, 15], collusion is detected by focusing on suspicious relationships among nodes according to the reputation values of nodes as well as the frequency of their interactions. In addition to frequency of relationships, social distance is checked for collusion detection in [15]. As an example, providing frequent *high* ratings by low-reputed nodes with short social distance is detected as a suspicious relationship. In [16] the k-means clustering is used to partition the population into a set of clusters. Consequently, each clusters is investigated for being a collusive cluster. The idea for identifying the collusion is that colluders not only try to be more self-serving (self promoting) but also damage outsiders more than a normal cluster. Other approaches, such as game theory, have also been proposed to detect collusion [17, 18].

In what follows, we review some notable reputation models which aimed to detect and mitigate collusion. Since our scheme is based on a clustering technique, we will have a brief review of clustering and the related concepts, as well.

3.1 Reputation Models

Not all the reputation models have taken collusion into consideration. Although resistance of trust and reputation models have been analyzed against some malicious scenarios [9], our focus is on collusion. In this section, we study the ways that some of the most assertive models deal with collusion and analyze their solutions. EigenTrust and PeerTrust are two of the most well known collusion-resistant models which are used in our evaluations. Hence, we will analyze them in more details.

3.1.1 EigenTrust

In EigenTrust [1] a unique global trust value is calculated for each node which is equivalent to reputation notion that we defined in this paper. In this model, a few nodes are considered as *pretrusted* nodes and have an important role in the convergence of the EigenTrust algorithm. The reputation value of a node i is evaluated based on the local trust values that other nodes assign to it. In calculating the reputation value of i, local trust values of the others are weighted by their global reputation. In particular, the reputation of each node is calculated as follows:

$$t_i^{(k+1)} = (1-\alpha)(c_{1i}t_1^{(k)} + \ldots + c_{ni}t_n^k) + \alpha p_i \quad (1)$$

where c_{ji} denotes the local trust value of j towards i, $t_j^{(k)}$ is reputation value of j at the k^{th} time interval, p_i is the priori trust value towards the pretrusted nodes, and $\alpha \in [0,1]$ is a parameter.

In the matrix notation, the local trust values form a matrix C:

$$C = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1j} & \cdots & c_{1n} \\ c_{21} & c_{12} & \cdots & c_{2j} & \cdots & c_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ c_{i1} & c_{i2} & \cdots & c_{ij} & \cdots & c_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nj} & \cdots & c_{nn} \end{pmatrix}, \quad \vec{c_i} = \begin{pmatrix} c_{i1} \\ c_{i2} \\ \vdots \\ c_{ii} \\ \vdots \\ c_{in} \end{pmatrix} \quad (2)$$

The local trust values are based on the satisfaction of nodes about the services they received from other nodes. In particular, the local trust of i towards j is calculated as follows:

$$c_{ij} = \frac{\max(s_{ij}, 0)}{\sum_j \max(s_{ij}, 0)}, \quad (3)$$

$$s_{ij} = Sat(i,j) - Unsat(i,j) \quad (4)$$

where s_{ij} is the difference between the number of satisfactory and unsatisfactory interactions between the nodes. In other words, c_{ij} is the normalized value of s_{ij}.

In a distributed environment a node i can ask its acquaintances about their opinions about other nodes:

$$c_{ik} = \sum_j c_{ij}c_{jk} \quad (5)$$

By repeatedly traversing longer paths, after a large enough number of iterations, all nodes in the system will have the same trust vector.

EigenTrust uses these two mechanisms to mitigate collusion:

- A higher weight is assigned to the ratings from the pretrusted nodes.
- The system assigns weights to the ratings according to the raters' global reputation value.

The main deficiency of this model in collusion mitigation is that it does not distinguish referral trust from functional trust. Dependency on specific formulation is another disadvantage of the model.

3.1.2 PeerTrust

In PeerTrust, a node's trustworthiness is evaluated by other nodes in the community based on their past experiences. Hence, a node's trustworthiness is equivalent to the reputation notion that we defined in this paper. PeerTrust [2] utilizes multiple factors in its reputation evaluation and is more resilient than EigenTrust against collusion. The reputation of a node u is formulated as follows:

$$T(u) = \alpha \frac{\sum_{i=1}^{I(u)} s(u,i)\, Cr(p(u,i))\, TF(u,i)}{\sum_{i=1}^{I(u)} Cr(p(u,i))\, TF(u,i)} + \beta CF(u) \quad (6)$$

where $I(u)$ denotes the total number of transactions performed by node u with all the other nodes, $p(u,i)$ denotes the transaction partner in the i^{th} transaction, $S(u,i)$ denotes the normalized amount of satisfaction the node receives from $p(u,i)$ in the i^{th} transaction, $Cr(v)$ denotes the credibility of the feedback submitted by a node v, $TF(u,i)$ denotes the adaptive transaction context factor in the i^{th} transaction, and $CF(u)$ denotes the adaptive community context factor for node u. As an example, the size of a transaction can be considered as its context factor. The credibility of v from the view point of another node w, is computed as:

$$Cr\left(p\left(u,i\right)\right) = \frac{Sim(p\left(u,i\right), w)}{\sum_{i=1}^{I(u)} Sim(p\left(u,i\right), w)} \quad (7)$$

and

$$Sim\left(v,w\right) = 1 - \sqrt{\frac{\sum_{x \in IS(v) \cap IS(w)} \left(\frac{\sum_{i=1}^{I(v,x)} S(x,i)}{I(v,x)} - \frac{\sum_{i=1}^{I(w,x)} S(x,i)}{I(w,x)} \right)^2}{\left| IS\left(v\right) \cap IS\left(w\right) \right|}} \quad (8)$$

where $I(u,v)$ denotes the total number of transactions performed by the node u with another node v, and $IS(v)$ denotes the set of nodes that have interacted with the node v. The nodes which submit feedbacks after their transactions can be awarded by $CF(u)$. One way of implementing $CF(u)$ can be as follows:

$$CF\left(u\right) = \frac{F(u)}{I(u)} \quad (9)$$

where $F(u)$ denotes the total number of feedbacks node u gives to the others.

PeerTrust uses the following mechanisms to mitigate collusion:

- The accurate management of the credibility of a node as a recommender helps PeerTrust to effectively overcome many of the security threats.
- This model uses the Personal Similarity Measure (PSM) for evaluation of referral trust. Distinction between functional trust and referral trust is the main advantage of this model in over EigenTrust.

In Section 6, PeerTrust will be shown to be more resistant to collusion than EigenTrust, in our simulations. Even so, it is not completely immune to collusion.

3.1.3 TrustGuard

In TrustGuard [11], the trust model is designed to be capable of handling four important issues in a node's behavior namely detecting sudden fluctuations, distinguishing between declines and rises, tolerating unintentional failures, and reflecting consistent behavior. The trust model in TrustGuard resembles a Proportional-Integral-Derivative (PID) controller which is used in control systems. In particular, the trust value at time t is calculated by the following formula:

$$TV(t) = \alpha R(t) + \beta \frac{1}{t} \int_0^t R(t)\,dx + \gamma \frac{d}{dx} R(t)|_{x=t} \quad (10)$$

The first term in (Equation 10) uses the latest rating of the behavior of the node (at time t; proportional gain); the second term incorporates its past performance (integral gain); and the third term is responsible for the sudden changes in the performance (derivative gain). TrustGuard uses the following mechanisms to mitigate collusion:

- Eliminating fake transactions by making use of transaction proofs.
- Camouflage resistance by the derivative term in (Equation 10).

In order to resolve the need to keep a long history, fading memories can be utilized. They summarize the last $2^m - 1$ trust values of a node by maintaining just m trust values. Fading memories again have logarithmic growth as time passes. A better solution is put forward in [19]. Another important problem with TrustGuard is that it does not distinguish referral from functional trust.

3.1.4 Fuzzy Reputation-Based Trust Model

This model uses a trust hierarchy to evaluate reputations, which is made up of four categories of nodes: self, neighbors, friends, and strangers [20]. Reputation of node Y (or trust in our terminology) is evaluated by another node X as follows:

$$repY/X_{before\_int} = ArepY/X_i + B\frac{\sum_i \alpha_i repY/X_i}{\sum_i \alpha_i} + C\frac{\sum_j \beta_j repY/X_j}{\sum_j \beta_j} + D\frac{\sum_l \delta_l repY/X_l}{\sum_l \delta_l} \quad (11)$$

where A, B, C, D are parameters ($A+B+C+D=1$) and the reputation values are restricted to be in $[0, k]$, where k is the predefined maximum reputation value. We do not go into the details of (Equation 11) and refer the interested reader to [20]. $repY/X_{before\_int}$ denotes the reputation of Y at X before their interaction, and $repY/X$ is the latest calculated reputation of Y by X. It is adjusted to account for the time passed since the last time node X was interested in finding the reputation of Y. The multiplicands of B, C, and D are the weighted sums of the reputation of Y as reported to X by the neighbors, friends, and strangers, respectively.

Referral trusts towards the neighbors are somehow considered in the model by calculating α_i as follows:

$$\alpha_i = a \times Sim + b \times Act + c \times Pop \quad (12)$$

in which $a + b + c = 1$, and Sim, Act, and Pop are activity, similarity, and popularity factors. Similarity of two nodes indicates how these nodes resemble each other in their reputation values and evaluation procedures. This is evaluated as follows:

$$Sim = 1 - \sqrt{\frac{\sum_n (u_i - v_i)^2}{25n}} \quad (13)$$

where u_i is the reputation value of node i in the reputation vector of the initiator node, v_i is the reputation value of node i in the reputation vector of the target node, and n is the total number of nodes appearing in the reputation vectors of both the initiator and the target nodes. Note that the factor 25 is used for normalization.

The activity of node X is evaluated by using:

$$Act = \frac{\sum int\_from_X}{\sum int\_from_{nodes}} \quad (14)$$

where $\sum int\_from_X$ denotes the sum of all interactions done by node X in the past time interval, which are reported by other nodes, and $\sum int\_from_{nodes}$ is the sum of all interactions done by all nodes in this interval.

The popularity value of node X is as follows:

$$Pop = \frac{\sum int\_with_X}{\sum int\_with_{nodes}} \quad (15)$$

where $\sum int\_with_X$ denotes the sum of all interactions done with node X in the past time interval, and $\sum int\_with_{nodes}$ the sum of all interactions done by all nodes in this interval.

The deficiency of the above-mentioned formulas is that colluders can easily affect the activity and popularity values of each other by interacting with a high frequency.

3.1.5 Discussion

Most of the models and collusion mitigation methods suffer from one or more of the following problems:

- Functional and referral trust are not distinguished from each other. It is necessary to distinguish these two concepts in order to detect dishonest recommenders. For instance, [1] and [11] suffer from this problem.
- Try to detect collusion in individual nodes. Therefore, all nodes need to know the whole trust network, which is impractical.
- Some other mitigation approaches are proposed in the literature. However, they are again specific to a model and cannot be used by the other models like [1] and [11].

3.2 Detection of Colluders via Clustering

Finding colluders can be modeled as discovering *unconventional subgraphs* of a trust network. Colluders create a group and interact with each other most of the time in order to gain maximum local trust values and boost themselves unfairly. In the trust networks, *density* is an indicator of significance. Actually dense regions indicate high degrees of interactions. The task of discovering specific subgraphs is studied in three different categories of algorithms in the literature: *clustering*, *dense subgraph detection* (see [4, 21]), and *community detection* algorithms (see [22–25]). The three concepts are similar to each other in many aspects. Clustering is broader than community detection, and clustering algorithms like hierarchical clustering can be used to detect communities in a social network, as well [26]. Dense subgraph detection is similar to clustering, too. Colluders actually create a dense subgraph in the trust network, and dense components can be discovered by clustering algorithms [21].

Clustering is the task of accumulating similar nodes in the same groups. The grouping can be based on a distance or similarity measure. The Mikowski, Euclidean, City-block, Mahalanobis distances and cosine similarity are notable measures in clustering algorithms [27]. The algorithms are divided into two categories: supervised and unsupervised. In contrast to the supervised algorithms, the number of clusters is not determined before running an unsupervised algorithm. From another point of view, clustering algorithms are divided into two categories: (hard) partitioning and hierarchical. In hierarchical algorithms, a weight is calculated for each pair of nodes in the network. The weight represents the degree of similarity, distance, or how closely connected the nodes are. Then, the edges between the pairs are established in the same order as the weights, starting by the pair with the greatest weight [23, 26]. These edges have no direct connection with the edges of the original network [26].

3.2.1 Discussion

According to the analysis of different classes of detection methods, we focus on clustering algorithms to detect collusion. We do not aim to detect the precise structure of colluder communities, and it is enough to flag suspicious nodes to be colluders. In addition, we do not want to extract all the existing clusters or put all nodes in a cluster. Hierarchical algorithms build a tree of clusters (dendogram) but the full hierarchy is not useful to us. In fact, the existing clustering algorithms are inefficient for collusion detection.

4 Our Proposed Scheme

In this section, we introduce our collusion mitigation scheme. Colluders use all mechanisms to achieve their goals. They include providing bad services, and giving dishonest feedbacks and recommendations. Hence, collusion is the most complicated attack against a reputation system, and its mitigation is a big step towards detecting and relieving other attack scenarios. First, we introduce our attacker model and assumptions. Then, we describe the details of our scheme.

4.1 Attacker Model

We assume to be faced with powerful attackers [8] with the following characteristics:

- **Location:** All attackers are insiders. They are authenticated and have legitimate access to the system (contrary to the outsiders).
- **Activity:** All attackers are active. They actively participate in the system and may inject false information, modify trust information, or attempt to interrupt the availability of the system.
- **Population:** Attackers are part of a coalition.

4.2 Collusion Mitigation

In order to mitigate collusion, we propose a similarity measure, and a similarity-based algorithm. In the following, first we discuss our measure. Then, we propose our collusion mitigation algorithm, and analyze it.

4.2.1 Colluders Similarity Measure (CSM)

Similarity measures can be used to cluster the nodes in a graph. Based on the behavior of colluders in a trust

network, we define the Colluders Similarity Measure (CSM). CSM is inspired by the Personality Similarity Measure (PSM) [2]. However, it is totally different from PSM, and does not rely on global information. Moreover, we use CSM along with an algorithm for collusion detection which leads to a better performance than PeerTrust. Two nodes will be similar according to CSM if their satisfaction values are in agreement with each other. In particular, the CSM between two nodes v and w is formulated as follows:

$$CSM(v,w) = 1 - \sqrt{\frac{\sum_{x \in IS(v) \cap IS(w)} \left(\frac{S^+(v,x) - S^-(v,x)}{I(v,x)} - \frac{S^+(w,x) - S^-(w,x)}{I(w,x)} \right)^2}{|IS(v) \cap IS(w)|}} \quad (16)$$

where $I(v,x)$ denotes the total number of transactions performed by v with another node x, and $IS(v)$ denotes the set of nodes that v has received services from. CSM is undefined for nodes without any shared interaction partners. In addition, $S^+(v,x)$ and $S^-(v,x)$ respectively denote the positive and negative satisfaction feedbacks that x has received from v. Colluders give dishonest positive feedbacks to their friends and dishonest negative feedbacks to the others. In order to be effective, they interact with each other with high frequencies. Therefore, if v and w are colluders, then a large portion of $IS(v) \cap IS(w)$ will be colluders too. Subsequently, v and w will have a high similarity according to CSM. In the following section, we use CSM in our clustering algorithm to discover colluders.

4.2.2 Colluders Discovery Algorithm (CDA)

Colluders create dense regions, which is a result of high number of interactions. Clustering algorithms can be used to detect unconventional subgraphs, but the problem is that investigating all parts of the trust network for discovering colluders is a very time-consuming task. To solve this problem, we use frequency checking as a heuristic to restrict the size of the graph which is investigated to find colluders. Our proposed Colluders Discovery Algorithm (CDA) is presented in Algorithm 1.

Before running CDA, the whole trust network must be prune, and the algorithm only processes a small set of suspicious nodes. In other words, CDA should be run on the list of suspicious nodes, and by a central component the system, which we name it the *Central Control Component (CCC)*. We consider that nodes are monitored by other nodes which we call *Trust Managers*. When a node receives positive feedbacks from another node with a frequency more than a threshold (th_1), their trust managers mark them to be suspicious, and report their identities to CCC. CCC periodically executes CDA on the suspicious nodes to find colluders. These components are described in more detail in Section 5.

Algorithm 1 The CDA pseudocode.

```
CDA(suspected)
 1  k ← 0
 2  for all i in suspected
 3       do cluster[i] ← 0
 4          for all j in suspected
 5               do if i ≠ j
 6                    then s ← CSM(i, j)
 7                         if s ≠ −1
 8                           then sim[i][j] ← s
 9                                sim[j][i] ← s
10                                k ← k + 1
11                                sim-sorted[k] ← ⟨i, j, s⟩
12                    else break
13  SORT-SIM(sim-sorted)
14  for c ← 1 to k
15       do ⟨i, j, max⟩ ← sim-sorted[k − c + 1]
16          l ← ∅
17          for all x in suspected
18               do if x ≠ i and x ≠ j and cluster[x] = 0 and
                       sim[x][i] > th2 and sim[x][j] > th2
19                    then cluster[x] ← c
20                         ADD(l, x)
21          for all x in l
22               do if cluster[x] ≠ 0
23                    then for all y in l
24                              do if cluster[y] ≠ 0 and x ≠ y
                                   and S⁻(x, y) > ε0
25                                   then cluster[y] ← 0
26  l ← ∅
27  for all x in suspected
28       do if cluster[x] ≠ 0
29             then ADD(l, x)
30  return l
```

The pseudocode of CDA is presented in Algorithm 1. At first, CDA calculates the similarity between all the nodes which are reported to have suspicious activity (lines 1-12). Then, selects the link with the highest CSM between nodes (i,j) provided that this link has not been chosen so far (lines 13-15). In the case that there is such link, it selects suspicious nodes having the following conditions, and mark them to be in the same cluster (lines 16-20):

- The node is not in the colluders list in this run of CDA, yet.
- $CSM(\text{selected node}, i) > th_2$.
- $CSM(\text{selected node}, j) > th_2$.

CDA clusters nodes with similarity more than a threshold (th_2). After that, CDA checks the interactions between the nodes in the created cluster and delete the end node of a link if it has negative feedbacks more than a threshold ϵ_0 (lines 21-25). This is done according to the assumption that the probability of a negative outgoing link from the colluders' cluster to the other nodes is higher. The before-mentioned steps are repeated until all suspicious links are processed. Finally, the nodes which are detected to be in some cluster are returned as the colluders list (lines 26-30).

As mentioned in Section 1, separation of functional and referral trust has an important role in mitigation of collusion. Adjustment of other nodes' opinions in evaluating the reputation of a node can be done by using their referral trust values. The colluder groups are assumed to be non-overlapping in CDA. This assumption does not prevent CDA from detecting a colluder because the returned list only specifies the colluder nodes, and not their communities.

4.3 Discussion

Some nodes might be falsely reported to be suspicious. Here we list some factors that affect the false positive in our scheme (ϵ_1):

- In some reputation models like EigenTrust, pretrusted nodes may cause false positives. This is because they have high reputation values, and are chosen with a high probability as service providers.
- Nodes with high reputation values can be other sources of false positive for the same reason.

Although the above-mentioned nodes may result in false positives, CDA does not detect them to be colluders because they have low CSM with the real colluders. Colluders interact with each other with high frequency and have high CSM among their friends. Therefore, CDA can detect them as colluders.

5 The Architecture

The architecture we present in this section implements CDA in an efficient manner in a distributed reputation system. This architecture can be considered to be a proof of concept to show the generality of our scheme in regard to different trust models and systems. The other goals of the architecture are as follows:

- **Embedding referral trust:** Consideration of referral trust is integrated into reputation systems via a black list of colluders identified by CDA. If a node is marked as a colluder, its recommendations will be ignored in the calculation of reputations.
- **Distribution:** We implement CDA in a distributed manner which incorporates a base trust model improved by the referral trust we talked about earlier. Most of the reputation and trust algorithms should be used in a distributed system. In addition, distributed systems are generally more robust against malicious behaviors.
- **Randomization and redundancy:** Trust managers (which will be introduced later) are chosen randomly so that the manipulation of trust information becomes harder for an attacker. In addition, several trust managers are used to compute the trust value of each node and if their opinions differ, the result will be based on their majority.

5.1 Components

Reputation systems are typically based on common information in order to reflect the community's impression in general. Therefore, we assume that the reputation value of each node is evaluated by aggregating local trust values of other nodes towards it. All the models which we reviewed in Section 3 use this mechanism. Our architecture adds some components to a reputation system to mitigate collusion. The system is composed of the following components:

- *Nodes:* They are at the lowest level and interact with each other.
- *Trust managers:* One or a specific number of trust managers are chosen among the nodes to be the trust manager(s) of a node. Not only each node monitors several nodes, but also is simultaneously monitored by others. Indeed, all nodes monitor each other in the system.
- *Central Control Component (CCC):* This preselected node (component) is the core of the architecture. It becomes active with a specific frequency and performs some regulatory jobs in the reputation system, including running CDA.

Our main goal is mitigating collusion in a distributed manner and by the cooperation of these components. In the following subsections, we describe their functionality.

5.1.1 Local Trust Models

Our scheme is applicable to reputation systems which evaluate reputation in a manner compatible with the Figure 2 (step 3 is introduced in our scheme). Trust- and reputation-relevant information is collected in step 1. Then, trust is evaluated locally by each node (step 2). The colluders are filtered out from the subsequent steps via the black list of colluders produced by running CDA in step 3. These local trust values are aggregated in step 4 and reputation is evaluated in step 5. Finally, the reputation information is used to choose a reputable partner in step 6.

Our scheme is not restricted to a specific local trust model, and we empower reputation evaluation by collusion mitigation. Our scheme refers to a base trust model to calculate local trust values (local trust model). The local trusts can be evaluated by various techniques such as simple summation or average of ratings, Bayesian models, fuzzy models, discrete trust models, belief models, and flow models [28]. Resistance to simple malicious behaviors is left to the base trust model. Hence, the more resistant the trust model is, the more effective is the overall system.

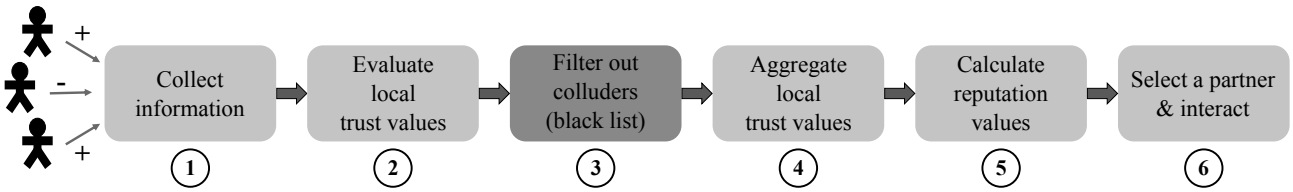

Figure 2. Steps in reputation evaluation.

5.1.2 Trust Managers

Trust managers in our architecture are similar to score managers in [1]. The main functionality of a trust manager is to calculate the reputations of its *sub-nodes*, based on the feedbacks provided by other nodes. In addition, trust managers check the frequency of positive and negative feedbacks which their sub-nodes receive from others. If a trust manager perceives that the frequency of a specific feedback towards a sub-node is more than a threshold (standard), it will report the identities of the related nodes to CCC. Trust managers not only directly receive incoming feedbacks from other nodes to its sub-nodes, but also record feedbacks of their sub-nodes to others (local trust values.

A distributed hash table (DHT) can be used to assign trust managers to nodes. In this case, all nodes are members of an overlay network. The overlay network will facilitate the management, discovery, and query processing in the system. Moreover, multiple trust managers can be assigned to a node. Different hash functions should be used to assign multiple trust managers to nodes. An example with three hash functions (h_1, h_2, h_3) is depicted in Figure 3. Trust managers can use majority voting to avoid malicious results. For instance, if a trust manager of A wants to evaluate its reputation, and A has served another node B, then the trust manager should ask the trust managers of B about its satisfaction from A (local trust value). Then, the trust manager compares these local trust values and act based on the majority, in case of inconsistencies. The idea of assigning multiple trust managers will improve attack resistance (randomization), privacy (anonymity), and fault tolerance (redundancy) of our scheme. For increasing system reliability and eliminating effect of the malicious trust managers (who manipulate trust values), the only way is considering multiple trust managers to detect malicious trust managers. Since they manipulate trust values, their ideas and votes are inconsistent with other trust managers. Increasing the number of trust managers leads to have more resilient reputation management system against malicious trust managers. However, we leave it for future research, and only consider one trust manager for each node.

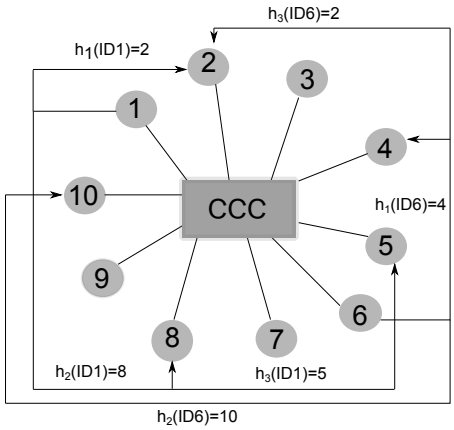

Figure 3. An example of our proposed architecture for eleven nodes.

5.1.3 Central Control Component (CCC)

The central control component (CCC) is the core of our architecture. If a trust manager perceives a suspicious activity (high frequency of positive or negative feedbacks), reports them to CCC. Periodically, CCC is activated and runs CDA on the reported nodes. A report from the trust manager of A about feedbacks by B has the following details:

- Identity of both nodes
- Positive and negative frequency feedbacks that A receives from B
- The satisfaction vector of A about the other nodes that interacts with them. The vector is required to calculate the CSM between A and B.

If CCC needs to know more satisfaction vectors (e.g. the satisfaction vector of B) to calculate similarities, then it will get the vectors from their respective trust managers.

Figure 3 depicts an example of the proposed architecture for eleven nodes. In this figure, the trust managers of nodes 1 and 6 are shown. We consider only one CCC in the system. This node is a special pre-selected node which is in direct communication with the trust managers. Thanks to the frequency heuristic, CCC is not responsible to mitigate all malicious behaviors, and only acts on suspicious nodes in a controlled frequency. Hence, CCC does not restrict scalability of the system.

Algorithm 2 The CSM pseudocode.

```
CSM(v, w)
1  d ← 0
2  n ← 0
3  for all x in IS(v)
4        do if CONTAINS(IS(w), x)
5              then d ← d + ( (S+(v,x) − S−(v,x)) / I(v,x) −
6                             (S+(w,x) − S−(w,x)) / I(w,x) )^2
7                   n ← n + 1
8  if n > 0
9     then return 1 − sqrt(d/n)
10    else return −1
```

6 Evaluation

First, we analyze the asymptotic complexity of our scheme. Then, compare its effectiveness with two other schemes.

6.1 Asymptotic Complexity Analysis

CDA can be classified to be a non-overlapping hierarchical (and unsupervised) clustering algorithm based on a similarity measure (see Algorithm 1). Hierarchical algorithms build a tree of clusters (dendogram). However, we build a one-level tree. Hence, CDA is a one-level agglomerative algorithm.

Our proposed scheme imposes computational and communicational overheads to the reputation system. The main computational overhead is on the *CCC* side to run CDA on suspicious nodes. The number of edges in a complete graph is $m(m-1)$ in which m is the total number of nodes. If the ratio of colluders to the number of nodes be $\alpha < 1$, the number of suspected nodes is $n = \alpha m$. Note that α does not take a high value. Otherwise, the majority of the nodes will be malicious, and the system will lose its useful functionality.

CSM can be efficiently calculated in $O(m)$ if the set of interaction partners ($IS(\cdot)$) is stored in a hash table (see Algorithm 2.) Therefore, calculation of similarity between all suspected nodes (lines 1-12) will take $O(n^2 m)$ time. The SORT-SIM procedure in line 13 will take $O(n^2 \log n^2) = O(n^2 \log n)$. The clustering loop (lines 16-20) and heuristic removal of nodes from the cluster (lines 21-25) will take $O(n)$ and $O(n^2)$, respectively. The maximum number of edges in the *sim-sorted* list is $n(n-1)$. Therefore, the running time of the clustering part (lines 13-25) is $O(n^3 + n^4) = O(n^4)$ Making up the colluders list (lines 26-30) will also take $O(n)$.

Therefore, the worst-case runtime complexity of CDA is $O(n^2 m + n^2 \log n + n^4) = O(n^2 m + n^4)$. As colluders form a small subset of nodes in practice, our scheme is practical. The amount of memory required to store the feedbacks (S^+, S^-) and the interaction partners (IS) is $O(m^2)$. In addition, the memory footprint of the CDA algorithm is $O(n^2)$.

The main communication overhead of the system is related to the messages exchanged between the trust managers and the *CCC*. In order for a trust manager to report a suspicious node to *CCC*, it should send the satisfaction vector of that node, too. At the worst case, the node has interacted with all nodes in the system, so the length of its satisfaction vector is m. As mentioned, there are n colluders in the system. Hence, the communication complexity of each report is $O(nm)$ *CCC* must know all satisfaction vectors of suspected nodes in order to run CDA. Most of them are reported by trust managers, but if *CCC* does not have a specific satisfaction vector, it should ask form the responsible manager.

6.2 Simulation Scenario

In this section, we evaluate our scheme and compare it with EigenTrust [1] and PeerTrust [2] in a simulation setting. The models are chosen for the following reasons:

- They are among the well-known trust and reputation models, and have strong theoretical bases. Researchers still try to improve them (see [29] and [15] as examples.)
- Both are compatible to be used as base local trust models in our scheme. In other words, they calculate the reputation of a node based on the local trust values of the other nodes towards it.
- Both models are designed with collusion-resistance in mind.

In this section, we evaluate the resistance of these models, with and without our scheme to prove its effectiveness.

6.3 Adoption of the Base Models

In order to adopt EigenTrust and PeerTrust by our scheme, slight modifications are necessary. EigenTrust does not originally distinguish referral from functional trust, and reputation value of each node is used as its referral trust, too. Therefore, we modify (Equation 1) to embed referral trust and filter out colluders from aggregation:

$$t_i^{(k+1)} = (1-\alpha)(\gamma_1 c_{1i} t_1^{(k)} + \ldots + \gamma_n c_{ni} t_n^k) + \alpha p_i, \quad (17)$$

where γ_j will be 0 if CCC recognizes node j as a colluder (j is in the black list), and 1 otherwise. Indeed, if CDA detects j to be a colluder, then its trust value towards node i will be ignored.

PeerTrust uses PSM to estimate referral trust. In our simulation results, we will show that PeerTrust performs better than EigenTrust in similar simulation

settings. Nevertheless, it is not totally immune to collusion. In order to filter out colluders, we modify the aggregation function in PeerTrust from (Equation 6) to the following:

$$T(u) = \alpha \frac{\sum_{i=1}^{I(u)} \gamma_i s(u,i)(Cr(p(u,i))\, TF(u,i))}{\sum_{i=1}^{I(u)} (Cr(p(u,i))\, TF(u,i))} + \beta CF(u) \tag{18}$$

where γ_j will be 0 if CCC recognizes j as a colluder, and 1 otherwise.

6.4 Types of Nodes

The behaviors of nodes in the system are very important in evaluations, and different behaviors can be assumed for them. In our evaluations, three types of nodes are simulated:

- **Normal:** Nodes with this behavior always give honest feedbacks to the others. They may provide a bad service with a small probability $\epsilon < 0.1$ because of failures. Hence, they provide good service with probability $P_{\text{Good service}} = 1 - \epsilon$.
- **Pretrusted:** These nodes are only simulated where EigenTrust is in use. They provide good services and give honest feedbacks with probability 1.
- **Colluder (with camouflage):** These nodes give dishonest positive feedbacks to their friends and dishonest negative feedbacks to the others. Hence, their probability of giving honest feedbacks is 0. They may have oscillation in their behavior and the probability of providing a good service in their case is $0 \leq P_{\text{Good service}} \leq 1$, in general.

6.5 Configuration and Settings

Threshold values in our scheme have a very important role in its performance and effectiveness. They can be set statically or dynamically. We set Th_1 dynamically in our simulations, and adjust it according to the mean value of interaction frequencies in the system. Specifically, if we assume the mean interaction frequency to be β, then Th_1 will be set to be $\beta + \mu$, where μ is a fixed tolerance parameter. In fact, μ declares that how much the frequency of a relation can be higher than the mean value while not being suspicious. We set $\mu = 0.3$ in our simulations. The other parameter of the system is the period of CCC activation. We set this value to 50—i.e. after each 50 query cycles, CCC updates the colluders list.

Each run of our simulations contains 600 query cycles; each simulation is repeated 25 times; and the average results are reported. In each query cycle, each node creates a query with probability 0.8. In order to select a service provider, our nodes use a probabilistic algorithm similar to the one in [1]. Particularly, each node chooses a service provider with probability 90% according to (Equation 19), and with probability 10%, it selects a service provider with reputation 0 (newcomer).

$$P_i = \frac{t_i}{\sum_{j=0}^{m} t_j} \tag{19}$$

where P_i is the probability that node i is chosen as a service provider, t_i is the reputation value of i, and m is the number of service providers.

We consider only one trust manager for each node as we evaluate resistance to collusion, not to other types of attack. The simulations are run with $n = 50, 125, 200$ nodes, and percentage of colluders is $c = 10\%, 25\%, 40\%$.

6.6 Results

Figure 4 to 6 depict our simulation results for EigenTrust. We have plotted the average reputation of each type of node versus the percentage of colluders c in these figures. There are all the three kinds of nodes in these figures: pretrusted, colluder, and normal nodes. The number of nodes n is 50, 125, and 200, respectively; and $P_{\text{Good service}} = 0.2$ for colluders. In these figures, we can compare the average reputation of each kind of node in two cases. The case in which the original EigenTrust model is used, and the one when the model is empowered by our scheme (EigenTrust+A). In the latter case, the average reputations of pretrusted and normal nodes have increased, and those of the colluders have decreased.

The bars representing each of the latter cases are overlapped (not accumulated) to save space and make comparisons easier. For example, in Figure 4, when the percentage of colluders is 10%, pretrusted nodes have earned an average reputation of about 0.5 with the original EigenTrust, and about 0.8 with EigenTrust+A. All the figures with overlapping bars must be interpreted in this way.

Figure 7 illustrates the individual reputations in either case for 125 nodes where 25% of which are colluders. It is evident in these figures that EigenTrust cannot mitigate colluders on its own, and colluders have very high reputation values compared to the other nodes in Figure 7 (a). However, by applying our scheme, the colluders have very low reputation values, and pretrusted nodes have reputations close to 1. Individual reputation values with different settings have the same pattern, and we omit their respective figures for the sake of brevity. Figure 8 depicts the simulation time of each query cycle for all values of n, where the percentage of colluders is 25%. According to this figure, the simulation time is increased by applying our scheme. By increasing n, the difference between

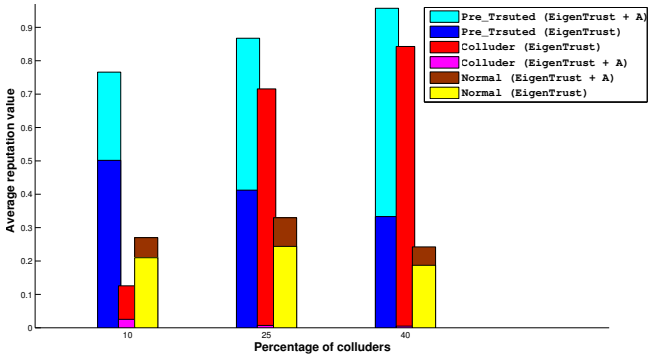

Figure 4. Average reputation value of different types of nodes vs. the percentage of colluders c in the EigenTrust case ($n = 50$).

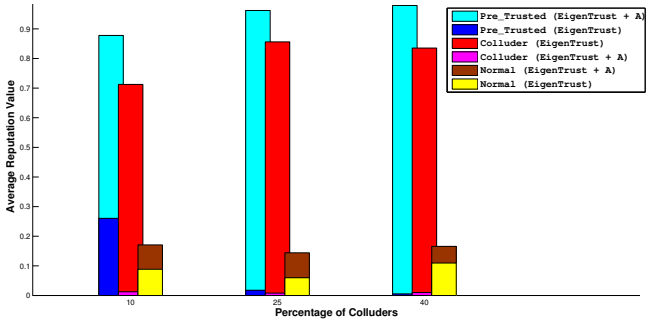

Figure 5. Average reputation value of different types of nodes vs. the percentage of colluders c in the EigenTrust case ($n = 125$).

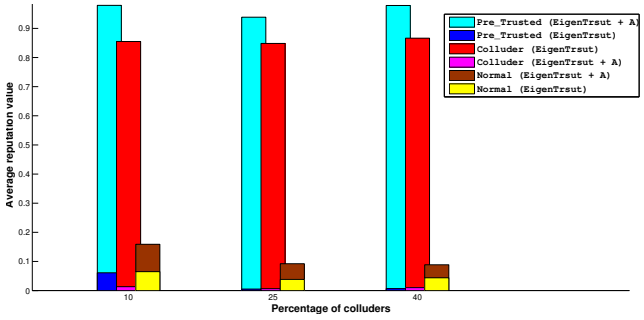

Figure 6. Average reputation value of different types of nodes vs. the percentage of colluders c in the EigenTrust case ($n = 200$).

simulation time in EigenTrust and EigenTrust+A is intensified.

Figure 9 to 11 depict our results in the PeerTrust case. All conditions in these simulations are similar to the EigenTrust ones except that we do not have pretrusted nodes in PeerTrust. In these figures, we can compare the average reputation of each kind of node in two cases. The case in which the original PeerTrust model is used, and the one when the model is empowered by our scheme (PeerTrust+A). Similar to the EigenTrust results, by applying our scheme, the average reputations of normal nodes have increased, and those of the colluders have decreased (to a very low value). The individual reputation values are depicted in Figure 12. Colluders have earned very high reputations when the original PeerTrust was in use. However, by applying our scheme, their reputation is decreased to a very low value, and normal users could reach high reputations. Figure 13 depicts the simulation time of each query cycle. Analogous to EigenTrust results, PeerTrust+A is more time-consuming than PeerTrust and their difference increases by n.

6.7 Discussion

According to the results presented earlier, the pretrusted nodes gain higher reputation when our scheme is applied to EgienTrust. According to (Equation 19), the probability of choosing a node as a service provider is proportional to its reputation. When colluders gain high reputation values, in the one hand, the probability of choosing them as service providers enhances. On the other hand, the probability of choosing pretrusted nodes declines. With our scheme, the average reputation values of colluders decrease. This is because their dishonest ideas and recommendations about the other colluders, as well as the other nodes are ignored. By using CDA, we detect colluders and put them in a black list. Therefore, they can neither promote themselves nor degrade others.

In our scheme, the average reputation values of normal nodes enhance. This is because when the reputation values of the colluders decrease, the probability of choosing normal nodes increases. Both the original EigenTrust and PeerTrust use weighted equations for reputation evaluation. For example, consider (Equation 6) for reputation evaluation in PeerTrust. In this case, when colluders assign low local trust values to the nodes that are out of their coalition, only the numerator diminishes, and the reputations of normal nodes decline, consequently. However, when we use (Equation 18) and apply the filtering according to CDA, the colluders are filtered, and both the numerator and denominator decline simultaneously. Therefore, the reputations of normal nodes do not decrease in the same manner as in the former case.

7 Conclusions and Future Works

We presented a method to minimize the impact of colluders on the performance of reputation systems. The system detects colluders by its Colluders Detection Algorithm (CDA), in which the Colluders Similarity Measure (CSM) is used to cluster colluders. We empower reputation systems by referral trust by filtering out unfair feedbacks and recommendations by colluders from trust aggregation. We proposed a distributed architecture to implement CDA by adopting a base

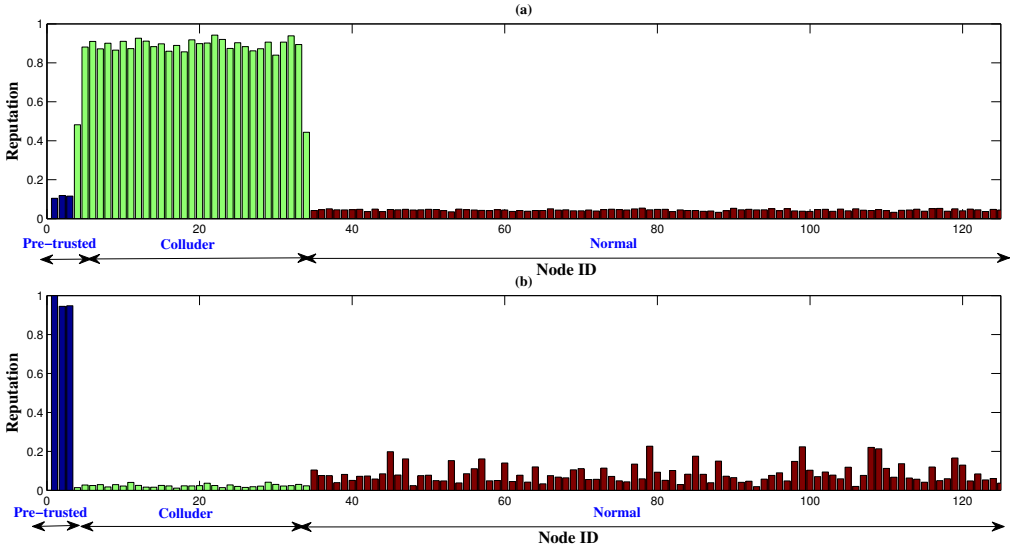

Figure 7. Individual reputation value of each node in (a) EigenTrust, and (b) EigenTrust+A ($n = 125$ and $c = 25\%$).

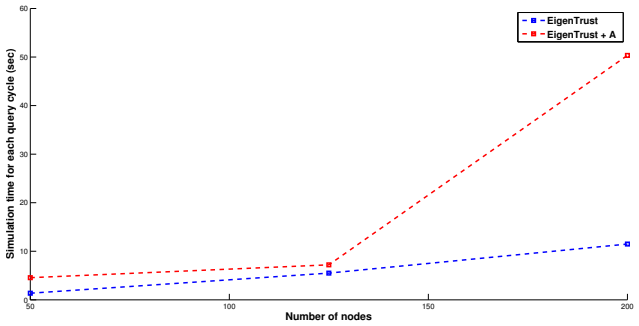

Figure 8. Simulation time for each query cycle vs. the number of nodes in the EigenTrust case ($c = 25\%$).

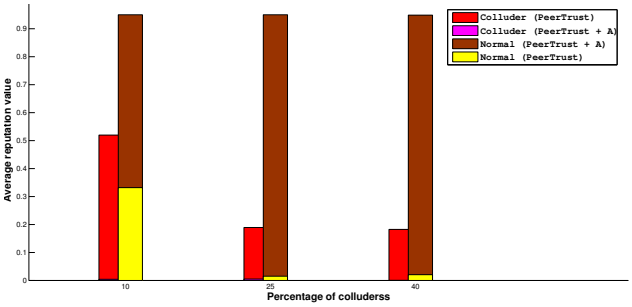

Figure 9. Average reputation value of different types of nodes vs. the percentage of colluders c in the PeerTrust case ($n = 50$).

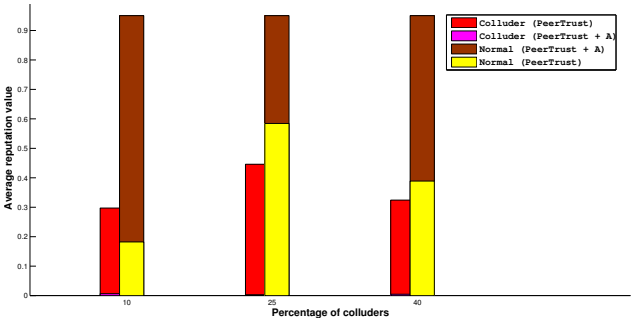

Figure 10. Average reputation value of different types of nodes vs. the percentage of colluders c in the PeerTrust case ($n = 125$).

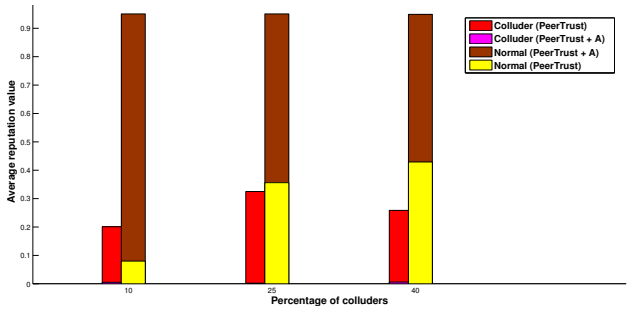

Figure 11. Average reputation value of different types of nodes vs. the percentage of colluders c in the PeerTrust case ($n = 200$).

local trust model. We showed the efficiency of our approach in simulation, and compared it to the original EigenTrust and PeerTrust models.

Two immediate future works our proposed scheme: graded filtering, and a more distributed architecture. In simulations, we filtered out colluders completely by assigning a zero weight to the information provider by them. One cannot claim to detect colluders accurately. Hence, in the future, we should assign a degree of belief to each node to be a colluder, and weight its information accordingly. The second future work is related to the Central Control Component (CCC) and Trust Managers in our proposal. Both can be points of failure in our architecture. An idea is to have more than one Trust Manager per node, and multiple CCCs each of which interacts with a subset of trust managers.

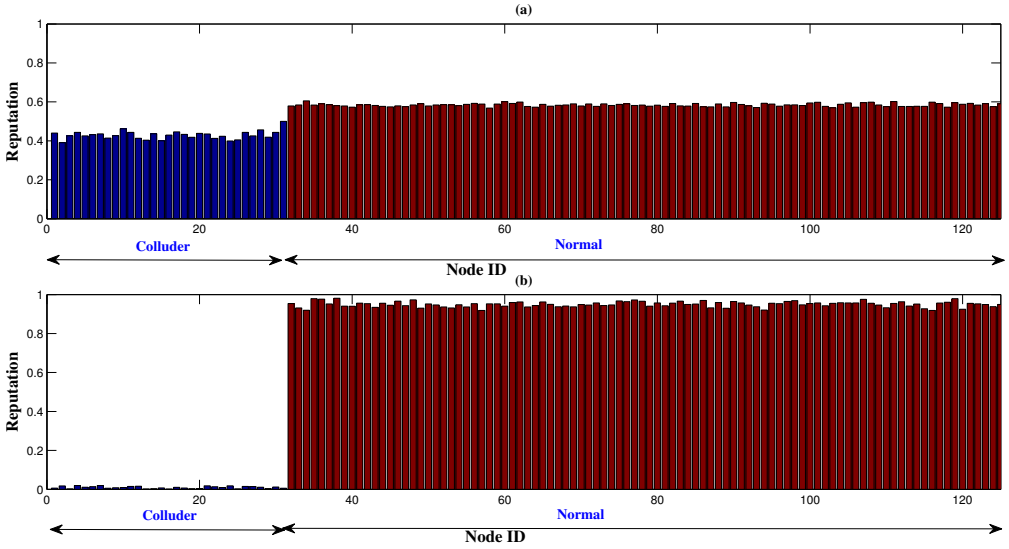

Figure 12. Individual reputation value of each node in (a) PeerTrust, and (b) PeerTrust+A ($n = 125$ and $c = 25\%$).

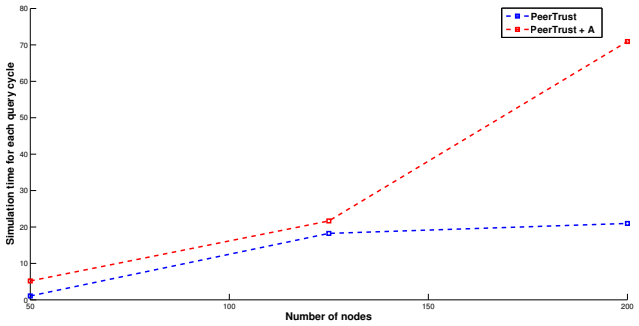

Figure 13. Simulation time for each query cycle vs. the number of nodes in the PeerTrust case ($c = 25\%$).

Acknowledgement

This research has been supported by a grant from the Research Institute for ICT (ITRC), Tehran, I.R. Iran under contract number T/500/12183.

References

[1] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651, Budapest, Hungary, 2003. ACMPress Press.

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

[3] Anupam Das and Mohammad Mahfuzul Islam. Securedtrust: a dynamic trust computation model for secured communication in multiagent systems. *Dependable and Secure Computing, IEEE Transactions on*, 9(2):261–274, 2012.

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*, volume 2. MIT Press, Cambridge, MA, USA, 2001.

[5] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision support systems*, 43(2):618–644, 2007.

[6] Xinlei Wang, Kannan Govindan, and Prasant Mohapatra. Collusion-resilient quality of information evaluation based on information provenance. In *Proceedings of the 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 395–403, Salt Lake City, UT, USA, 2011. IEEE.

[7] Audun Jøsang, Ross Hayward, and Simon Pope. Trust network analysis with subjective logic. In *Proceedings of the 29th Australasian Computer Science Conference-Volume 48*, pages 85–94. Australian Computer Society, Inc., 2006.

[8] Kevin Hoffman, David Zage, and Cristina Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Computing Surveys (CSUR)*, 42(1):1, 2009.

[9] Félix Gómez Mármol and Gregorio Martínez Pérez. Security threats scenarios in trust and reputation models for distributed systems. *Computers & Security*, 28(7):545–556, 2009.

[10] Xiaoning Jiang and Lingxiao Ye. Reputation-based trust model and anti-attack mechanism in p2p networks. In *Proceedings of the Second International Conference on Networks Security*

Wireless Communications and Trusted Computing (NSWCTC), volume 1, pages 498–501, Wuhan, Hubei, China, 2010. IEEE.

[11] Mudhakar Srivatsa, Li Xiong, and Ling Liu. TrustGuard: countering vulnerabilities in reputation management for decentralized overlay networks. In *Proceedings of the 14th International Conference on World Wide Web*, pages 422–431, Chiba, Japan, 2005. ACMPress Press.

[12] Runfang Zhou and Kai Hwang. Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing. *IEEE Transactions on Parallel and Distributed Systems*, 18(4):460–473, 2007.

[13] Ze Li, Haiying Shen, and K. Sapra. Collusion detection in reputation systems for peer-to-peer networks. In *Parallel Processing (ICPP), 2012 41st International Conference on*, pages 98–107, Sept 2012.

[14] Gayatri Swamynathan, KevinC. Almeroth, and BenY. Zhao. The design of a reliable reputation system. *Electronic Commerce Research*, 10(3-4):239–270, 2010.

[15] Ze Li, Haiying Shen, and K. Sapra. Leveraging social networks to combat collusion in reputation systems for peer-to-peer networks. In *Parallel Distributed Processing Symposium (IPDPS), 2011 IEEE International*, pages 532–543, May 2011.

[16] Reid Kerr and Robin Cohen. Detecting and identifying coalitions. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 3*, AAMAS '12, pages 1363–1364, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[17] Gianluca Ciccarelli and Renato Lo Cigno. Collusion in peer-to-peer systems. *Computer Networks*, 55(15):3517 – 3532, 2011.

[18] Bao Yu, Cao Tianjie, and Zeng Guosun. Resisting collusion by game in culture web. In *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2011 20th IEEE International Workshops on*, pages 262–267, June 2011.

[19] Roberto Aringhieri, Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, and Pierangelo Samarati. Fuzzy techniques for trust and reputation management in anonymous peer-to-peer systems. *Journal of the American Society for Information Science and Technology*, 57(4):528–537, 2006.

[20] Ayman Tajeddine, Ayman Kayssi, Ali Chehab, and Hassan Artail. Fuzzy reputation-based trust model. *Applied Soft Computing*, 11(1):345–355, 2011.

[21] Victor E. Lee, Ning Ruan, Ruoming Jin, and Charu Aggarwal. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*, pages 303–336. Springer, USA, 2010.

[22] Yunpeng Zhao, Elizaveta Levina, and Ji Zhu. Community extraction for social networks. *Proceedings of the National Academy of Sciences*, 108(18):7321–7326, 2011.

[23] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[24] Gaoxia Wang, Yi Shen, and Ming Ouyang. A vector partitioning approach to detecting community structure in complex networks. *Computers & Mathematics with Applications*, 55(12):2746–2752, 2008.

[25] Xutao Wang, Guanrong Chen, and Hongtao Lu. A very fast algorithm for detecting community structures in complex networks. *Physica A: Statistical Mechanics and its Applications*, 384(2):667–674, 2007.

[26] Mark E. J. Newman. Detecting community structure in networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):321–330, 2004.

[27] Rui Xu and Donald C. Wunsch II. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[28] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2007.

[29] Wang Miao, Xu Zhijun, Zhang Yujun, and Zhang Hongmei. Modeling and analysis of peertrust-like trust mechanisms in p2p networks. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 2689–2694. IEEE, 2012.

Mina Niknafs received her B.S. degree in Information Technology Engineering from Isfahan University of Technology in 2009 and her M.S. degree in Information Technology from Sharif University of Technology in 2012. Her research interests include security and trust management, attack tolerance of reputation systems, data and network security.

Sadegh Dorri Nogoorani received his B.S. degree in Software Engineering from University of Tehran in 2007 and his M.S. degree in Information Technology from Sharif University of Technology in 2009. He has been a Ph.D. student in Computer Engineering in Sharif University of Technology since 2009. His research interests include uncertainty theories, computational notions of trust, data and network security and privacy, risk management, and grid computing.

Rasool Jalili received his B.S. degree in Computer Science from Ferdowsi University of Mashhad in 1985, and M.S. degree in Computer Engineering from Sharif University of Technology in 1989. He received his Ph.D. in Computer Science from University of Sydney, Australia, in 1995. He then joined the Department of Computer Engineering, Sharif University of Technology in 1995. He has published more than 140 papers in international journals and conference proceedings. He is now an associate professor, doing research in the areas of computer dependability and security, access control, distributed systems, and database systems in his Data and Network Security Laboratory (DNSL).