TrustGuard: Countering Vulnerabilities in Reputation Management for Decentralized Overlay Networks

Mudhakar Srivatsa, Li Xiong, Ling Liu
College of Computing
Georgia Institute of Technology
{mudhakar, lxiong, lingliu}@cc.gatech.edu

Abstract

Reputation systems have been popular in estimating the trustworthiness and predicting the future behavior of nodes in a large-scale distributed system where nodes may transact with one another without prior knowledge or experience. One of the fundamental challenges in distributed reputation management is to understand vulnerabilities and develop mechanisms that can minimize the potential damages to a system by malicious nodes. In this paper, we identify three vulnerabilities that are detrimental to decentralized reputation management and propose TrustGuard − a *safeguard* framework for providing a highly dependable and yet efficient reputation system. First, we provide a dependable trust model and a set of formal methods to handle strategic malicious nodes that continuously change their behavior to gain unfair advantages in the system. A unique feature of our approach is to incorporate historical reputations and behavioral fluctuations of nodes into the estimation of their trustworthiness, guaranteeing that reputation should be built gradually, but drop quickly when a node starts to behave maliciously. Second, a transaction based reputation system must cope with the vulnerability that malicious nodes may misuse the system by flooding feedbacks with fake transactions. We propose a feedback admission control mechanism to ensure that only transactions with secure proofs can be used to file feedbacks. Third but not least, we identify the importance of filtering out dishonest feedbacks when computing reputation-based trust of a node, including the feedbacks filed by malicious nodes through collusion. We propose an effective mechanism to rate the feedback credibility of nodes, and demonstrate its effectiveness in discounting dishonest feedbacks. Our experiments show that, comparing with existing reputation systems, our framework is highly dependable and effective in countering malicious nodes regarding strategic oscillating behavior, flooding malevolent feedbacks with fake transactions, and dishonest feedbacks.

1. INTRODUCTION

A variety of electronic markets and online communities have reputation system built in, such as eBay, Amazon, Yahoo! Auction, Edeal, Slashdot, Entrepreneur. Recent works [4, 1, 3, 10, 16] suggested reputation based trust systems as an effective way for nodes to identify and avoid malicious nodes in order to minimize the threat and protect the system from possible misuses and abuses by malicious nodes in a decentralized overlay networks. Such systems typically assign each node a trust value based on the transactions it has performed with others and the feedbacks it has received.

WWW2005, May 10–14, 2005, Chiba, Japan.

For example, XRep [4] provides a protocol complementing current Gnutella protocol by allowing peers to keep track of and share information about the reputation of other peers and resources. EigenTrust [10] presents an algorithm similar to PageRank [13] that computes a trust value by assuming trust is transitive and demonstrated its benefits in addressing fake file downloads in a peer-to-peer file sharing network.

However, few of the reputation management work so far have focused on the vulnerabilities of a reputation system itself. One of the detrimental vulnerabilities is that a malicious node may strategically alter its behavior in a way that benefits itself such as starting to behave maliciously after it attains a high reputation. Another widely recognized vulnerability is the shilling attack [11] where malicious nodes submit dishonest feedback and collude with each other to boost their own ratings or bad-mouth non-malicious nodes. Last but not the least, malicious nodes can flood numerous fake feedbacks through fake transactions in a transaction-based feedback system. We believe that a highly dependable reputation system is needed to safeguard the system itself from malicious attacks through strategic oscillation, fake transactions, and dishonest feedbacks. For example, a node's reputation (represented by its trust value) should drop quickly as soon as it misbehaves [17], while it should be hard for any node to boost its reputation within a short period of time. In addition, the trust building techniques based on reputation should be robust and effective in countering attacks through fake transactions and dishonest feedbacks.

With these issues in mind, we present TrustGuard − a highly dependable reputation-based trust building framework. The paper has a number of unique contributions. First, we introduce a highly dependable trust model to effectively handle strategic oscillations by malicious nodes (Section 3). Second, we propose a feedback admission control mechanism to ensure that only transactions with secure proofs can be used to file feedbacks (Section 4). Third, we propose feedback credibility based algorithms for effectively filtering out dishonest feedbacks (Section 5). We also present a set of simulation based experiments, showing the effectiveness of the TrustGuard approach in guarding against each of the above vulnerabilities with minimal overhead. We conclude the paper with a brief overview of the related work (Section 7), and a conclusion (Section 8).

2. TRUSTGUARD: AN OVERVIEW

2.1 System Architecture

We first present a high level overview of the TrustGuard framework. Figure 1 shows a sketch of the decentralized

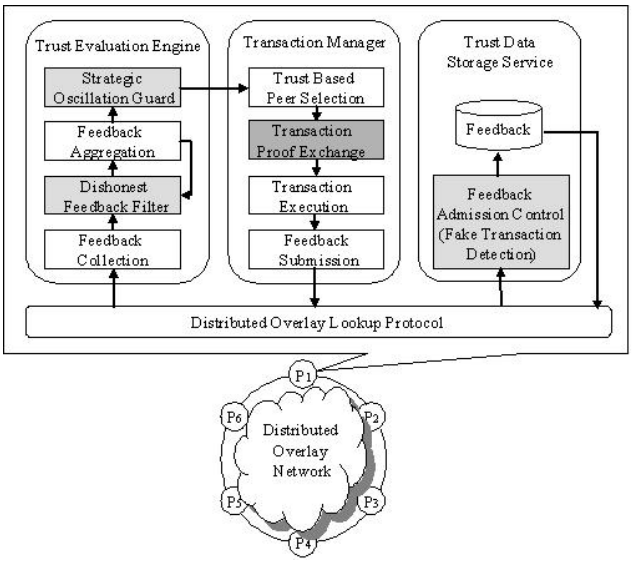

Figure 1: TrustGuard's Architecture

architecture of the dependable reputation management system. The callout shows that each node has a transaction manager, a trust evaluation engine and a feedback data storage service. Whenever a node n wants to transact with another node m, it calls the *Trust Evaluation Engine* to perform a trust evaluation of node m. It collects feedback about node m from the network through the overlay protocol and aggregates them into a trust value. Such computation is guarded by strategic oscillation guard and dishonest feedback filters. The *Transaction Manager* consists of four components. The trust-based node selection component uses the trust value output from the trust evaluation engine to make trust decisions before calling the transaction execution component. Before performing a transaction, the transaction proof exchange component is responsible for generating and exchanging transaction proofs. Once the transaction is completed, the feedbacks are entered by the transacting nodes. These feedbacks are routed to designated nodes on the overlay network for storage through a decentralized overlay protocol (e.g. DHT based protocol). The designated nodes then invoke their *data storage service* and admit a feedback only if it passes the feedback admission control where fake transactions are detected. The feedback storage service is also responsible for storing reputation and trust data on the overlay network securely, including maintaining replicas for feedbacks and trust values. We build the TrustGuard storage service on top of PeerTrust [16].

Although we implement the TrustGuard framework using a decentralized implementation that distributes the storage and computation of the trust values of the nodes, it is important to note that one could implement TrustGuard using different degrees of centralization. At one extremity, third-party trusted servers could be used for both trust evaluation and feedback storage. One can also utilize the trusted servers to support only selected functionality, for example, the transaction proof exchange (Section 4).

Finally, we assume that TrustGuard architecture is built on top of a secure overlay network. Thus, the overlay network should be capable of routing messages despite the presence of some malicious nodes and ensure that all nodes can be identified through some digital certification based mechanism. Readers may refer to [15, 7] for a detailed discussion on security issues in overlay networks.

2.2 Problem Statement and Solution Approach

The TrustGuard framework is equipped with several important safeguard components which are critical for providing a highly dependable reputation system that minimizes the potential threats and vulnerabilities in the reputation system itself. In the rest of the paper, we focus on the following three types of vulnerabilities, analyze the potential threats and describe countermeasures against such vulnerabilities using TrustGuard.

Strategic Oscillation Guard. Most existing reputation systems such as eBay use a combination of average feedbacks and the number of transactions performed by a node as indicators of its trust value. Our experiments show that using a simple average does not guard the reputation system against oscillating behavior or dishonest feedbacks. For example, a bad node may behave non-maliciously until it attains a good reputation (reflected in its trust value) and then behave maliciously. Or it could oscillate between building and milking reputation. A dependable reputation system should be able to penalize malicious nodes for such dynamic and strategic behavioral changes.

In TrustGuard, we promote the incorporation of the reputation history and behavior fluctuations of nodes into the estimation of their trustworthiness. We use adaptive parameters to allow different weighting functions to be applied to current reputation, reputation history, and reputation fluctuations. The current reputation of a node is computed from aggregating feedbacks about this node over the recent period. The reputation history is computed with an incremental aggregation of past reputation values of a node with a bias on recent history. The reputation fluctuation is computed using a derivative of the reputation value. We also develop a fading memories based optimization technique to reduce the cost of maintaining historical information of nodes.

Fake Transaction Detection. In a typical transaction-based feedback system, after each transaction, the two participating nodes have an opportunity to submit feedbacks about each other. This brings two vulnerabilities. First, a malicious node may flood numerous ratings on another node with fake transactions. Second, a malicious node may submit dishonest feedback about a transaction. A dependable trust model should be equipped with mechanisms to handle malicious manipulation of feedbacks to guard the system against such fake transactions, and to differentiate dishonest feedback from honest ones. In TrustGuard approach, we propose to bind a feedback to a transaction through transaction proofs. In other words, a feedback between nodes n and m on a given transaction is stored *if and only if* n and m indeed transacted with each other. Concretely, before two nodes perform a transaction, they exchange an unforgeable transaction proof. Once the transaction proof is fairly exchanged, both the nodes can submit feedback about the transaction using the proof. The feedback is then routed to appropriate nodes for storage and only those feedbacks with valid transaction proof will be admitted into the feedback database.

Dishonest Feedback Filter. While the fake transaction

detection guarantees that a feedback is associated with a real transaction, a malicious node may submit dishonest feedbacks in order to boost the ratings of other malicious nodes or bad-mouth non-malicious nodes. The situation is made much worse when a group of malicious nodes make collusive attempts to manipulate the ratings. In this paper, we build a dishonest feedback filter to differentiate dishonest feedbacks from honest ones. The filter essentially assigns a credibility value to a feedback source and weights a feedback in proportion with its credibility. We study two such credibility measures and their effectiveness in filtering out dishonest feedbacks in both non-collusive and collusive settings.

3. GUARDING FROM STRATEGIC MALICIOUS NODES

We define a strategic malicious node as a node that adapts its behavioral pattern (with time) so as to maximize its malicious goals. Consider a scenario wherein a bad node does not misbehave until it earns a high trust value. The scenario becomes more complicated when bad nodes decide to alternate between good and bad behavior at regular or arbitrary frequencies. In this paper, we primarily focus on *strategic oscillations* by malicious nodes and describe concrete and systematic techniques taken by TrustGuard to address both steady and sudden changes in the behavioral pattern of a node without adding heavy overheads to the system. Other possible behavioral strategies that could be employed by malicious nodes are not considered in this paper.

A dependable trust model should be capable of handling the following four important issues: ($P1$) sudden fluctuations in node behavior, ($P2$) distinguish an increase and decrease in node behavior, ($P3$) tolerate unintentional errors, and ($P4$) reflect consistent node behavior. We propose a dependable trust model that computes reputation-based trust of a node by taking into consideration: current feedback reports about the node, its historical reputation, and the fluctuations in the node's current behavior. First, we present an optimization theory based cost metric (Section 3.1) to formalize our design goals and then present TrustGuard's dependable trust model (Section 3.2).

3.1 Cost Model

The primary goal of our safeguard techniques is to maximize the cost that the malicious nodes have to pay in order to gain advantage of the trust system. We first formally define the behavior of a non-malicious and a malicious node in the system using the game theory approach [5]. A non-malicious node is the commitment type and a long-run player who would consistently behave well, because cooperation is the action that maximizes the player's lifetime payoffs. In contrast a strategic malicious node corresponds to an opportunistic player who cheats whenever it is advantageous for him to do so. Now we formally describe a cost model for building reputation-based trust and use this cost model to illustrate the basic ideas of maximizing the cost (penalty) to be paid by anyone behaving maliciously. Let $TV_n(t)$ denote the trust value as evaluated by the system for node n at time t ($0 \leq TV_n(t) \leq 1$). Let $BH_n(t)$ denote the actual behavior of node n at time t ($0 \leq BH_n(t) \leq 1$), modeled as the fraction of transactions that would be honestly executed by node n between an infinitesimally small time interval t and $t + dt$. Then, we define the cost function for a node b

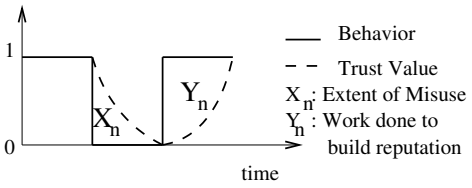

Figure 2: Cost of Building Reputation

as shown in Equation 1.

$$cost(b) \quad = \quad \lim_{t \to \infty} \frac{1}{t} * \int_0^t BH_b(x) - TV_b(x) \qquad (1)$$

Let G be the set of good nodes and B be the set of bad nodes. The objective is $\forall g \in G : TV_g(t) \approx 1$ and $\forall b \in B : cost(b)$ is maximized. Figure 2 provides an intuitive illustration of the above cost function for a strategic malicious node oscillating between acting good and bad. Referring to Figure 2, observe that the problem of maximizing the *cost* paid by the malicious nodes can be reduced to maximizing the area under $Y_n(t) - X_n(t)$, that is, minimizing the extent of misuse ($X_n(t) = max(TV_n() - BH_n(t), 0)$) and maximizing the cost of building reputation ($Y_n(t) = max(BH_n(t) - TV_n(t), 0)$).

In addition to maximizing the cost metric, we require TrustGuard to ensure that any node behaving well for an extended period of time attains a good reputation. However, we should ensure that the cost of increasing a node's reputation depends on the extent to which the node misbehaved in the past. For example a node that misbehaved for an extended period of time in the past should find it very hard to build reputation in a short span of time (although it is eventually possible).

3.2 Dependable Trust Model

Bearing the above analysis in mind, we present TrustGuard's dependable trust model in this section. Let $R(t)$ denote the raw trust value of node n at time t. Any of the existing trust evaluation mechanisms such as [16, 10] can be used to calculate $R(t)$. The simplest form can be an average of the ratings over the recent period of time. Let $TV(t)$ denote the dependable trust value of node n at time t and we compute $TV(t)$ using Equation 2.

$$TV(t) = \alpha * R(t) + \beta * \frac{1}{t} * \int_0^t R(x)dx + \gamma * \frac{d}{dx} R(x) \mid_{x=t} \quad (2)$$

Equation 2 resembles a Proportional-Integral-Derivative (PID) controller used in control systems [12]. The first component (*proportional*) refers to the contribution of the current reports received at time t. The second component (*integral*) represents the past performance of the node (*history information*). The third component (*derivative*) reflects the sudden changes in the trust value of a node in the very recent past. Choosing a larger value for α biases the trust value of a node n to the reports currently received about n. A larger value of β gives heavier weight to the performance of the node n in the past. The averaging nature of the proportional and integral components enables our model to tolerates errors in raw trust values $R_n(t)$ ($P3$) and reflect consistent node behavior ($P4$). A larger value of γ *amplifies sudden changes* in behavior of the node in the recent past (as indicated by the derivative of the trust value) and handles sudden fluctuations in node behavior ($P1$). We discuss techniques to distinguish increase and decrease in node

behavior ($P2$) later in this Section.

We now describe a simple implementation of the dependable trust model described above. For simplicity, we assume that the trust values of nodes are updated periodically within each time period T. Let successive time periods (intervals) be numbered with consecutive integers starting from zero. We call $TV[i]$ the dependable trust value of node n in the interval i. $TV[i]$ can be viewed as a function of three parameters: (1) the feedback reports received in the interval i, (2) the integral over the set of the past trust values of node n, and (3) the current derivative of the trust value of node n.

Incorporating feedbacks by computing $R[i]$. Let $R[i]$ denote the raw reputation value of node n computed as an aggregation of the feedbacks received by node n in interval i. Let us for now assume that all the feedbacks in the system are honest and transactions are not faked. In such a scenario, $R[i]$ can be computed by using a simple average over all the feedback ratings received by node n in time interval i. We defer the extension of our safeguard to handle dishonest feedbacks and fake transactions later to sections 4 and 5 respectively.

Incorporating History by Computing Integral. We now compute the integral (history) component of the trust value of node n at interval i, denoted as $H[i]$. Suppose the system stores the trust value of node n over the last $maxH$ (maximum history) intervals, $H[i]$ could be derived as a weighted sum over the last $maxH$ reputation values of node n using Equation 3.

$$H[i] = \sum_{k=1}^{maxH} R[i-k] * \frac{w_k}{\sum_{k=1}^{maxH} w_k} \tag{3}$$

The weights w_k could be chosen either *optimistically* or *pessimistically*. An example of an optimistic summarization is the exponentially weighted sum, that is, $w_k = \rho^{k-1}$ (typically, $\rho < 1$). Note that choosing $\rho = 1$ is equivalent to H being the average of the past $maxH$ reputation values of node n. Also, with $\rho < 1$, H gives more importance to the more recent reputation values of node n. We consider these evaluations of H optimistic since they allow nodes to attain higher trust values rather quickly. On the contrary, a pessimistic estimate of H could be obtained with $w_k = \frac{1}{R[i-k]}$. Such an evaluation assigns more importance to those intervals where the node behaved particularly badly.

Strengthening the dependability of $TV[i]$. Once we have calculated the feedback-based reputation ($R[i]$) for the node n in the interval i and its past reputation history ($H[i]$), we can use Equation 4 to compute the derivative component ($D[i]$). Note that Equation 4 uses $H[i]$ instead of $R[i-1]$ for stability reasons.

$$D[i] \quad = \quad R[i] - H[i] \tag{4}$$

We now compute the dependable trust value $TV[i]$ for node n in the interval i using Equation 5:

$$\begin{gathered} TV[i] = \alpha * R[i] + \beta * H[i] + \gamma(D[i]) * D[i] \\ \text{where } \gamma(x) = \gamma_1 \text{ if } x \geq 0 \text{ and } \gamma(x) = \gamma_2 \text{ if } x < 0 \end{gathered} \tag{5}$$

In this equation, $TV[i]$ is derived by associating different weights γ_1 and γ_2 for a positive gradient and a negative gradient of the trust value respectively, enhancing the dependability of $TV[i]$ with respect to sudden behavioral changes of node n. One of the main motivations in doing so is to set $\gamma_1 < \beta < \gamma_2$, thereby increasing the strength of the derivative component (with respect to the integral component) when a node shows a fast degradation of its behavior, and lowering the strength of the derivative component when a node is building up its reputation (recall $P2$ in our design goal). Our experiments (see Section 6) show that one can use the rich set of tunable parameters provided by Equation 5 to handle both steady and sudden changes in the behavior of a strategic malicious node.

3.3 Performance Optimization Using Fading Memories

In TrustGuard, we compute the dependable trust value of a node n in interval i based on its current reputation, its reputation history prior to interval i and its reputation fluctuation. In computing reputation history, we assume that the system stores the reputation-based trust values of node n for the past $maxH$ number of intervals. By using a smaller value for $maxH$, we potentially let the wrong-doings by a malicious node to be *forgotten* in approximately $maxH$ time intervals. However, using a very large value for $maxH$ may not be a feasible solution for at least two reasons: (i) The number of trust values held on behalf of a long standing member of the system could become extremely large. (ii) The computation time for our trust model (Equations 3 and 5) increases with the amount of data to be processed. In the first prototype of TrustGuard, we introduce *fading memories* as a performance optimization technique to reduce the space and time complexity of computing $TV[i]$ by allowing a trade-off between the history size and the precision of the historical reputation estimate.

One simple technique to trade-off history size with precision would be to aggregate the trust value in each k consecutive intervals into one value. However, from our experiments we observed that it is vital to keep the trust values in the recent past very precise. Therefore, we propose to aggregate data over intervals of exponentially increasing length in the past $\{k^0, k^1, \cdots, k^{m-1}\}$ into m values (for some integer $k > 0$). Observe that the aggregates in the recent past are taken over a smaller number of intervals and are hence more precise. This permits the system to maintain more detailed information about the recent trust values of node n and retain *fading memories* (less detailed) about the older trust values of node n. Given a fixed value to the system-defined parameter m, one can trade-off the precision and the history size by adjusting the value of k.

Now we describe how we implement fading memories in TrustGuard. To simplify the discussion, let us assume that $k = 2$. With fading memory optimization, our goal is to summarize the last $2^m - 1$ ($\sum_{i=0}^{m-1} 2^i = 2^m - 1$) trust values of a node by maintaining just m (=$\log_2(2^m)$) values. This can be done in two steps. (i) we need a mechanism to aggregate $2^m - 1$ trust values into m values, and (ii) we need a mechanism to update these m values after each interval.

TrustGuard performs Step 1 as follows. In the interval t, the system maintains trust values in intervals $t-1, t-2, \cdots, t-2^m$ in the form of m trust values by summarizing intervals $t-2^j, t-2^j-1, \cdots, t-2^{j+1}+1$ for every j ($j = 0, 1, \cdots, m-1$), instead of maintaining one trust value for each of the $2^m - 1$ time intervals. Figure 3 provides an example where $k = 2$ and $m = 3$.

Now we discuss how TrustGuard performs Step 2. Let $FTV^t[j]$ ($0 \leq j \leq m-1$) denote the faded trust val-

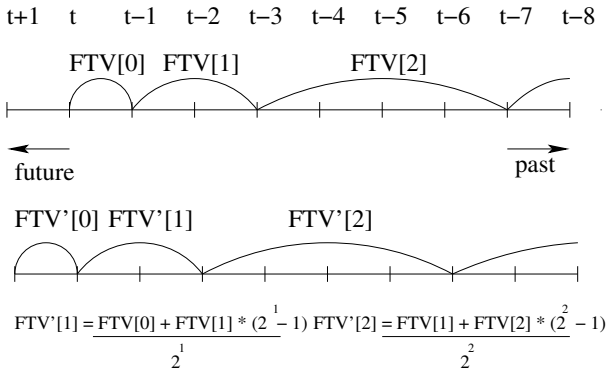

Figure 3: Updating Fading Memories: $FTV[i]$ denotes the faded values at time t and $FTV'[i]$ denotes the faded values at time $t+1$

ues of node n at interval t. Ideally, re-computing FTV for interval t requires all of the past $2^m - 1$ trust values. With fading memories we only store m summarization values instead of all the $2^m - 1$ trust values. Thus, at interval t we approximate the trust value for an interval $t - i$ $(1 \leq i \leq 2^m)$ by $FTV^t[\lfloor \log_2 i \rfloor]$. We use Equation 6 to approximate the updates to the faded trust values for interval j $(j = 0, 1, 2, \cdots, m-1)$ with the base case $FTV^{t+1}[0] = R[t]$. Figure 3 gives a graphical illustration of Equation 6 for $m = 3$.

$$FTV^{t+1}[j] = \frac{(FTV^t[j] * (2^j - 1) + FTV^t[j-1])}{2^j} \quad (6)$$

In summary, the fading memories approach closely resembles the way human beings remember their experiences. Our experiments show the vital role of fading memories in optimizing the performance of the trust system while maintaining its dependability at trust model level.

4. GUARDING FROM FAKE TRANSACTIONS

We have presented a dependable trust metric, focusing on incorporating reputation history and reputation fluctuation to guard a reputation system from strategic oscillation of malicious nodes. We dedicate this and the next section to vulnerabilities due to fake transactions and dishonest feedbacks and their TrustGuard countermeasures.

In TrustGuard, we tackle the problem of fake transactions by having a feedback bound to a transaction through a transaction proof such that a feedback can be successfully filed only if the node filing the feedback can show the proof of the transaction. Our transaction proofs satisfy the following properties: (i) Transaction proofs are unforgeable; and are hence generated only if the transacting nodes indeed wished to transact with each other, and (ii) Transaction proofs are always exchanged atomically; that is, a malicious node m cannot obtain a proof from a non-malicious node n without sending its own proof to node n. The atomicity property of the exchange of proofs guarantees fairness; that is, each of the transacting parties would be able to file feedbacks at the end of the transaction. In the absence of exchange atomicity a malicious node m could obtain a proof from node n but not provide its proof to the non-malicious node n; hence, a non-malicious node n may *never* be able to file complaints against the malicious node m.

We first present a technique to construct unforgeable proofs that curb a malicious node from flooding feedbacks on other non-malicious nodes. Then we employ techniques based on electronic fair-exchange protocol to ensure that transaction proofs are exchanged fairly (atomically). It is important to note that the proofs act as signed contracts and are thus exchanged before the actual transaction takes place. If the exchange fails, a good node would not perform the transaction. Nonetheless, if the exchange were unfair, a bad node could file a feedback for a transaction that *never actually* happened.

Note that the fake transaction detection does **not** prevent two collusive malicious nodes from faking a large number of transactions between each other, and further give good ratings with exchanged transaction proofs. This type of collusion will be handled by our next safeguard - dishonest feedback filter. Additionally, one could also deploy a mechanism which associates a non-negligible monetary cost with each transaction so that the attackers may not be able to afford many fake transactions.

4.1 Constructing Unforgeable Transaction Proofs

A simple and secure way to construct proofs of transactions is to use a public key cryptography based scheme. Assume that every node n has an associated pair of public key and a private key pair, namely, $\langle PK_n, RK_n \rangle$. We assume that the public keys are tied to nodes using digital certificates that are notarized by trusted certification authorities. A transaction T is defined as $T = \langle Txn\ Descr \rangle \parallel \langle time\ stamp \rangle$, where $\langle Txn\ Descr \rangle$ is a description of the transaction and the symbol $\parallel$ denotes string concatenation. Node n signs the transaction with its private key to generate a transaction proof $PT = RK_n(T)$ and send it to node m. If the proofs are fairly exchanged then node n would obtain $RK_m(T)$ as a proof of transaction T with node m and vice versa. The key challenge now is how to exchange proofs atomically to guarantee fairness.

4.2 Fair Exchange of Transaction Proofs

Significant work has been done in the field of fair electronic exchange [14, 2], aiming at guaranteeing *exchange atomicity*. There are several trade-offs involved in employing a fair-exchange protocol in the context of reputation management. In this section we analyze the feasibility of trust value based fair-exchange protocol and optimistic fair-exchange protocol for TrustGuard.

Trust Value based Fair-Exchange Protocol. Intuitively, one could achieve fair exchange of proofs between nodes n and m ($TV_n > TV_m$) by enforcing that the lower trust value node m sends its proof first to the higher trust value node n; following which the higher trust value node n would send its proof to the lower trust value node m. However, this solution is flawed. For example, a malicious node m with a high trust value may always obtain a proof from non-malicious node n with a lower trust value, but not deliver its proof to node n. Hence, a malicious node may pursue its malicious activities *indefinitely* without being *detected* by the trust system.

Optimistic Fair-Exchange Protocol. In the first prototype of TrustGuard, we adopt an optimistic fair-exchange protocol for exchanging transaction proofs. Optimistic fair-exchange protocols guarantee fair-exchange of two electronic items between two mutually distrusting parties by utilizing

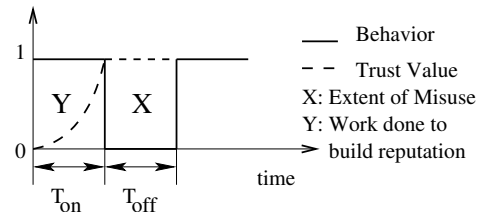

Figure 4: Cost of Building Reputation with Delayed Conflict Resolution

trusted third parties (*ttp*). However, they reduce the involvement of a *ttp* to only those exchanges that result in a *conflict*. Assuming that most of the parties in an open electronic commerce environment are good, the *ttp* is hopefully involved infrequently.

In particular, TrustGuard adopts the optimistic protocol for fair contract signing proposed by Micali [2]. The protocol assumes that the transacting parties n and m have already negotiated a would-be contract C. The nodes n and m now need to exchange the signed contracts, ($RK_n(C)$ and $RK_m(C)$) fairly. The protocol guarantees that if both the nodes commit to the contract then node n has a proof that node m has committed to the contract C and vice-versa; even if one of the parties does not commit to the contract C then neither party gets any proof of commitment from the other party. We map this protocol for fairly exchanging transaction proofs by using contract C as $C = T = \langle Txn\ Descr \rangle \parallel \langle time\ stamp \rangle$.

One of the major advantages of using such an optimistic fair-exchange protocol is that the *ttp* need not be always online. The *ttp* can infrequently come up online and resolve all outstanding conflicts before going offline. Note that minimizing the amount of time the *ttp* needs to be online significantly lowers its susceptibility to attackers. On the other hand, if the *ttp* comes up online very infrequently, the number of outstanding conflicts and the mean time to resolve a conflict would increase significantly. Nonetheless, all conflicts would be eventually resolved.

A strategic malicious node could exploit the delay in conflict resolution as shown in Figure 4. Let us assume that the *ttp* stays online for a time period T_{on} and then stays offline for a time period T_{off}. When the malicious node is building reputation, it behaves honestly and exchanges transaction proofs fairly (Y in Figure 4). However, after the malicious node has attained high reputation, it unfairly exchanges several proofs with other nodes in the system. By synchronizing the schedule of the *ttp*, the malicious node can ensure that none of the conflicts caused by its malicious behavior is resolved within T_{off} time units (X in Figure 4). Hence, despite of the fact that the malicious node behaved badly over a period of T_{off} time units its reputation does not fall. However, the moment all outstanding conflicts are resolved, the malicious node's reputation falls very steeply. Observe that cost paid by a malicious node (see Equation 1) is much lower in Figure 4 when compared to Figure 2 (wherein $T_{off} = 0$).

In conclusion, one needs to choose T_{off} very carefully so that: (i) the attackers do not have enough time to compromise the trusted third party, and (ii) maximize the cost paid by those malicious nodes that strategically exploit delayed conflict resolution.

5. GUARDING FROM DISHONEST FEEDBACKS

In the previous section we have discussed techniques to ensure that both transacting nodes have a fair chance to submit feedbacks. In this section we extend our safeguard model to handle dishonest feedbacks. The goal of guarding from dishonest feedbacks is to develop algorithms that can effectively filter out dishonest feedbacks filed by malicious nodes in the system.

We propose to use a credibility factor as a filter in estimating the reputation-based trust value of a node in the presence of dishonest feedbacks. Recall that, we use TV_n to denote the dependable trust value of node n and R_n to denote the reputation-based trust value of node n without incorporating past history (Integral component) and fluctuations (Derivative component). The main idea of using a credibility-based feedback filter in computing R_n is to assign higher weight to the credible feedbacks about node n and lower weight to the dishonest ones.

Concretely, we first extend the naive average based computation of trust value described in section 3.2 into a weighted average. Let $I(n)$ denote the set of interactions (transactions) performed by node n. Let $F_n(u)$ denote the normalized feedback rating (between 0 and 1) that a node n receives after performing an interaction u with another node. Let $CR_n(u)$ denote the feedback credibility of the node $u.x$ who submitted the feedback about node n after interaction u. The reputation-based trust of node n can be computed as $R_n = \sum_{u \in I(n)} F_n(u) * CR_n(u)$. The information about the set of transactions performed ($I(n)$) and the feedbacks received ($F_n(u)$ for $u \in I(n)$) can be collected automatically [16]. Our goal is to design a credibility filter function that is most effective in ensuring that more credible feedbacks are weighted higher and vice-versa.

A simple and intuitive solution is to measure feedback credibility of a node n using its trust value TV_n. We call it the Trust-Value based credibility Measure (TVM for short). Let $TV_{u.x}$ denote the trust value of node $u.x$ who had interaction u with node n. We can compute the trust value based credibility measure of node $u.x$ in the interaction u, denoted by $CR_n^{TVM}(u)$, using Equation 7.

$$CR_n^{TVM}(u) = \frac{TV_{u.x}}{\sum_{u \in I(n)} TV_{u.x}} \tag{7}$$

Several existing reputation-based trust systems use TVM or its variant to measure feedback credibility [16, 10]. The TVM solution is based on two fundamental assumptions. First, untrustworthy nodes are more likely to submit false or misleading feedbacks in order to hide their own malicious behavior. Second, trustworthy nodes are more likely to be honest on the feedback they provide. It is widely recognized that the first assumption is generally true but the second assumption may not be true. For example, it is possible that a node may maintain a good reputation by providing high quality services but send malicious feedbacks to its competitors. This motivates us to design a more effective credibility measure.

We propose to use a *personalized similarity measure* (PSM for short) to rate the feedback credibility of another node x through node n's personalized experience. Concretely, a node n will use a personalized similarity between itself and node x to weigh all the feedbacks filed by node x on any other node (say y) in the system. Let $IJS(n, x)$ denote the set of common nodes with whom both node n and x have

interacted, and $I(n,r)$ denotes the collection of interactions between node n and node r. We compute similarity between node n and x based on the root mean square of the differences in their feedback over the nodes in $IJS(n,x)$. More specifically, given a node m and an interaction $u \in I(m)$ performed by node m with node $u.x$, node n computes a personalized similarity-based credibility factor for u, denoted as $CR_n^{PSM}(u)$, using Equation 8.

$$CR_n^{PSM}(u) = \frac{Sim(n,u.x)}{\sum_{u \in I(n)} Sim(n,u.x)} \text{ where} \quad (8)$$

$$Sim(n,x) = 1 - \sqrt{\frac{\sum_{r \in IJS(n,x)} \left(\frac{\sum_{v \in I(n,r)} Fn(v)}{|I(n,r)|} - \frac{\sum_{v \in I(x,r)} Fx(v)}{|I(x,r)|} \right)^2}{|IJS(n,x)|}}$$

This notion of personalized (local) credibility measure provides a great deal of flexibility and stronger predictive value as the feedback from similar raters are given more weight. It also acts as an effective defense against potential malicious cliques of nodes that only give good ratings within the clique and give bad rating outside the clique. Using personalized credibility to weight the feedbacks will result in a low credibility for dishonest feedbacks by malicious cliques. This is particularly true when measuring the feedback similarity between a node m in a clique and a node n outside the clique. Our experiments show that PSM outperforms TVM when the percentage of malicious nodes become large and when the malicious nodes collude with each other.

In this section, we report results from our simulation based experiments to evaluate TrustGuard's approach to build dependable reputation management. We implemented our simulator using a discrete event simulation [8] model. Our system comprises of about $N = 1024$ nodes; a random $p\%$ of them is chosen to behave maliciously. In the following portions of this section, we demonstrate the effectiveness of the three guards that we have proposed in this paper.

6. EVALUATION

6.1 Guarding from Strategic Node Behaviors

In this section we evaluate the ability of our trust model to handle dynamic node behaviors. We first study the behavior of our guard against strategic oscillations by comparing the optimistic and pessimistic summarization techniques. We demonstrate the significance of various parameters in our dependable trust metrics by varying the weights assigned to reports received in the recent time window (α), the history (β), and the derivative component (γ). Then, we show the impact of history size ($maxH$) on the effectiveness of our trust model and the advantages of storing past experiences using fading memories.

For all experiments reported in this section, we studied four different models of strategic malicious behaviors. In Model I shown in Figure 5, the malicious nodes oscillate from good to bad behavior at intervals of regular time periods. In model II, the malicious nodes oscillate between good and bad behaviors at exponentially distributed intervals. In model III, the malicious nodes choose a random level of goodness and stay that level for an exponentially distributed duration of time. In model IV the malicious node shows a sinusoidal change in its behavior that is the node steadily and continuously changes its behavior unlike models I, II and III which show sudden fluctuations.

6.1.1 Comparing Optimistic and Pessimistic Summarizations

We first compare the two types of weighted summarization techniques discussed in section 3.2. Figure 6 shows the values obtained on summarization given the node behavior model I shown in Figure 5 (using $\rho = 0.7$, $maxH = 10$ and time period of malicious behavior oscillation = 10). The result shows that *mean value (mean)* and *exponentially weighted sum (exp)* have similar effect and they both are more optimistic than the *inverse trust value weighted sum (invtv)*. Observe that the more pessimistic a summarization is, the harder it is for a node (both malicious and non-malicious) to attain a high trust value in a short span of time and the easier it is to drop its trust value very quickly (recall our *cost* function in Equation 1). Also observe that the exponentially weighted sum in comparison to the mean rises quite steeply making it unsuitable for summarization.

6.1.2 Trust Model Parameters

Figure 7 shows the results obtained from our trust model with various parameter settings under the malicious behavior shown in model I (*m1*). *alpha* shows the results obtained when α is the dominant parameter ($\alpha \gg \beta, \gamma$). With a dominant α the trust model almost follows the actual behavior of the node since it amounts to disregarding the history or the current fluctuations in the behavior of the node (see Equation 2). *beta-invtv* shows the results obtained with β as the dominant parameter using inverse trust value weighted sum. With more importance given to the behavior history of a node, the trust value of a node does not change very quickly. Instead it slowly and steadily adapts to its actual behavior. *gamma* shows the results obtained with γ being the dominant factor. With a large γ the trust value responds very swiftly to sudden changes in the behavior of the node. Observe the steep jumps in the trust value that correspond to the time instants when the node changes its behavior. These results match our intuition, namely, α, β and γ are indeed the weights attached to the current behavior, historical behavior and the fluctuations in a node's behavior. Finally, *non-adaptive* shows the trust value of a node in the absence of dependable schemes to handle dynamic node behaviors. From Figure 7 it is evident that the cost paid by a malicious node in a non-adaptive model is almost zero, while that in a dependable model is quite significant.

6.1.3 Varying History Size

In this section we show the effect of history size $maxH$ on the *cost* (see Equation 1) paid by malicious nodes. Figure 8 shows a scenario wherein the malicious nodes oscillate in their behavior every 10 time units. Note that in this experiment we used $\alpha = 0.2$, $\beta = 0.8$, $\gamma_1 = 0.05$ and $\gamma_2 = 0.2$. Based on our experiences with the dependable trust model one needs to choose α and β such that $\frac{\beta}{\alpha}$ is comparable to $maxH$ (intuitively, this weights the history component in proportion to its size ($maxH$)). Note that this experiment uses $maxH = 5$ which is less than the time period of oscillations by the malicious nodes. From Figure 8 it is clear that the dependable trust models (*TrustGuard-adaptive* in figure) performs better in terms of cost to be paid by the malicious nodes than the non-adaptive trust model (recall the cost model in Section 3.1). However, this does not entirely maximize the cost paid by malicious nodes. Figure 9 shows the trust values obtained when $\alpha = 0.1$, $\beta = 0.9$, $\gamma_1 = 0.05$, $\gamma_2 = 0.2$ and $maxH = 15$ (larger than the time period of oscillation by the malicious node). Clearly, having

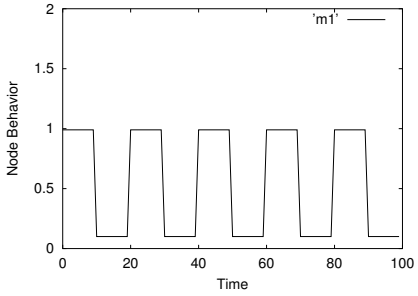

Figure 5: Model I

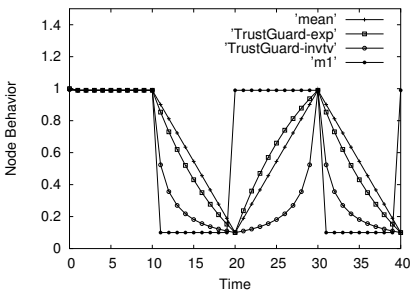

Figure 6: Optimistic Vs Pessimistic Summarization

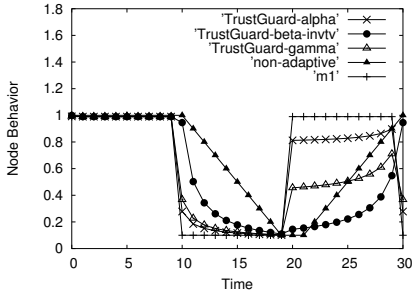

Figure 7: Effect of Varying Parameters in the Trust Model

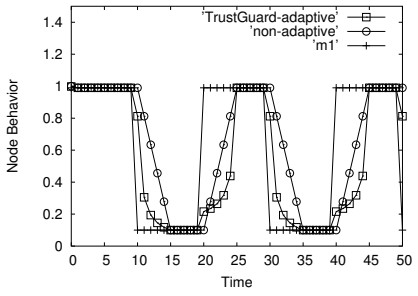

Figure 8: Trust Model with a Small History

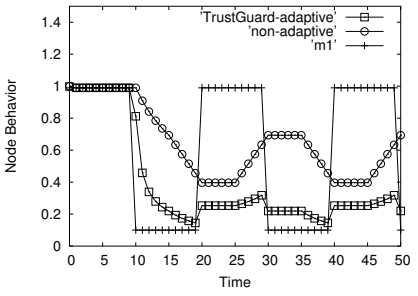

Figure 9: Trust Model with a Large History

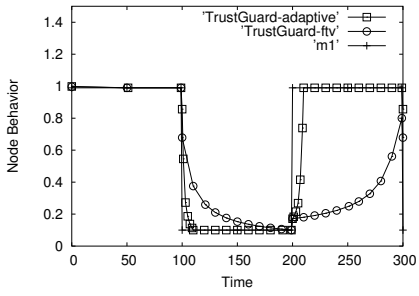

Figure 10: Trust Model with Fading Memories

a larger history ensures that one can maximize the cost paid by the malicious nodes. In fact, one observes that the cost paid by malicious nodes for $maxH$ equal to 5, 10 and 15 are in the ratio of $0.63 : 1 : 3.02$ respectively. This observation tells us that if a strategic malicious node knew that $maxH = 5$, then it would oscillate at a period equal to 5 time intervals since anyway the system does not remember its past performance beyond 5 time intervals. In short, by knowing the exact value of $maxH$, a strategic malicious node would start to oscillate with time period equal to $maxH$ so as to minimize its cost. It is interesting to note that, when the non-adaptive model is used, the cost paid by malicious nodes is close to zero for all values of time period of behavior oscillation and history size $maxH$.

6.1.4 Fading Memories

We now evaluate the effectiveness of the fading memories technique in efficiently storing the performance of a node over the last 2^{maxH} intervals using a logarithmically small number of values. Figure 10 shows the effect of using fading memories when a malicious node oscillates with time period equal to 100 time units. It compares a dependable trust model (*TrustGuard-adaptive* in figure) with $maxH = 10$ and a dependable trust model using fading memories (*TrustGuard-ftv* in figure) based technique with $m = 8$. From Figure 10 it is apparent that using a simple adaptive technique with $maxH = 10$ enables a bad node to recover from its bad behavior that stretched over 100 time units in just 10 additional time units, since the past performance of the node is simply forgotten after 10 time units. As we discussed in section 3, one of the design principles for dependable trust management is to prevent a bad node that has performed poorly over an extended period of time to attain a high trust value quickly. Clearly, the adaptive fading memories based technique can perform really well in this regard, since using just 8 values, it can record the performance of the node over its last 256 (2^8) time intervals. It is important to note that the solution based on fading memories has bounded effectiveness in the sense that by setting $m = 8$, any node could erase its malicious past over 256 time intervals. However, the key benefit of our fading memories based approach is its ability to increase the cost paid by malicious nodes, with minimal overhead on the system performance.

6.1.5 Other Strategic Oscillation Models

We also studied the cost of building reputation under different bad node behavior models discussed in the beginning of Section 6.1. From our experiments, we observed that the response of our trust model towards models II, III and IV are functionally identical to that obtained from model I (Figure 5). However, from an adversarial point of view, we observed that these strategies do not aid in minimizing the cost to be paid by malicious nodes to gain a good reputation when compared to model I. In fact, the cost paid by malicious nodes using models I, II, III and IV are in the ratio of $1 : 2.28 : 2.08 : 1.36$. In models II and III, the malicious nodes do not pursue their malicious activities the very moment they attain a high reputation. In model IV, the malicious nodes slowly degrade their behavior, which does not given them good benefits (see the extent of misuse $X_n(t)$ in Figure 2) when compared to a steep/sudden fall. Hence, a strategic malicious node that is aware of $maxH$ would oscillate with time period $maxH$ in order to minimize its cost (refer Equation 1). Nonetheless this emphasizes the goodness of our dependable trust model since it is capable of effectively handling even its worst vulnerability (model I with oscillation time period $maxH$).

T_{off}	0	0.05	0.1	0.2	0.3
$Cost$	1	0.9	0.85	0.78	0.66

Table 1: Relative Cost paid by Malicious Nodes Vs T_{off} (normalized by $maxH$)

6.2 Guarding from Fake Transactions

In this section we study the feasibility of using optimistic fair-exchange protocol for exchanging transaction proofs.

6.2.1 Trust Value based Protocol Vs Optimistic Protocol

Figure 11 shows the percentage of fair exchange of transaction proofs with progress in time for the two exchange protocols discussed in section 4, namely the trust value based fair exchange protocol and the optimistic fair exchange protocol. The experiment measures the percentage of fair transactions when 20% of the nodes are malicious. The optimistic fair-exchange protocol is agnostic to the number of transactions performed by the system, since they do not rely on any *evolving measures* like the trust values of nodes in the system. However, the trust value based exchange scheme pays for relying on node's trust value since a strategic malicious node may gain high trust value initially and then fake arbitrarily large number of transactions by unfairly exchanging transaction proofs without being *detected* by the system.

6.2.2 Trusted Server Offline Duration in Optimistic Protocol

As we have discussed in Section 4, it is important to decrease the amount of time the trusted third party server is online so as to make it less susceptible to attackers. However, doing so increases the amount of time it takes to resolve a conflict. We have shown in Section 4.2 that a malicious node can exploit the delay in conflict resolution to may ensure that none of its malicious activities be made visible to the trust management system for T_{off} units of time. In this experiment, we show how the transaction success rate varies with T_{off}, the time period for which the trusted third party server is offline.

Table 1 shows the normalized cost (see Equation 1) paid by malicious nodes when we introduce a delay in conflict resolution. We have normalized T_{off} with the history size ($maxH$) maintained by TrustGuard's adaptive trust model (see Section 3). Figure 1 shows that in order to keep the cost from dropping more that 10%, T_{off} should be no more than 5% of $maxH$. Note that this is another scenario where fading memories (see Section 3) helps the system. Fading memories essentially allow the history size ($maxH$) to be very large and hence the duration of the time for which a trusted third party server is offline could be sufficiently large without significantly decreasing the cost paid by malicious nodes.

6.3 Guarding from Dishonest Feedbacks

In this section we present an evaluation of our algorithm to filter dishonest feedbacks (section 5). Recall that the fake transaction guard does not prevent fake transactions between two malicious nodes. So, we simulated two settings, namely, non-collusive and collusive setting. In the collusive setting, a group of collusive malicious nodes may attempt to deterministically boost their ratings by providing highly positive feedbacks on each other through innumerable fake transactions.

Figures 12 and 13 show the error in trust computation as a function of the fraction of malicious nodes in the system in a non-collusive and a collusive setting respectively. Observe that the naive technique (an average of the feedback without credibility factor) for computing trust drops almost linearly with fraction of malicious nodes. Also, the naive technique and the TVM approach are extremely sensitive to collusive attempts even when the number of malicious nodes is very small. On the other hand, the PSM approach remains effective even with both large fraction of malicious nodes and collusion. Recall that PSM computes a *personalized* trust value and hence, the trust value of a node may be different from the perspective of various other nodes in the system. For example, the trust value of a node m from the perspective of other nodes within its clique may be very high, and yet, the trust value of node m as seen by other nodes in the system might be very low. Therefore, the similarity metric used by PSM is very effective even in an overwhelming presence of collusive malicious nodes.

7. RELATED WORK

Dellorocas [5] provides a working survey for research in game theory and economics on reputation. The game theory based research lays the foundation for online reputation systems research and provides interesting insights into the complex behavioral dynamics. Most of the game theoretic models assume that stage game outcomes are publicly observable. Online feedback mechanisms, in contrast, rely on private (pair-wise) and subjective ratings, thereby raising concerns on the incentive for providing feedbacks or the truthfulness of the feedback.

In the P2P domain reputation management systems like P2Prep [3], Xrep [4] and EigenTrust [10] have recently emerged. P2PRep provides a protocol on top of Gnutella to estimate trustworthiness of a node. It does not discuss trust metrics in detail and does not have evaluations. XRep extends P2PRep by assigning a reputation value for both peers and resources. EigenTrust assumes that trust is transitive and addresses the weakness of the assumption and the collusion problem by assuming there are pre-trusted nodes in the system. We argue that pre-trusted nodes may not be available in all cases. More importantly, neither of these reputation management systems addresses the temporal dimension of this problem (strategic behavior by malicious nodes) and the problem of fake transactions.

Dellorocas [6] has shown that storing feedback information on the most recent time interval is enough; and that summarizing feedback information for more than one window of time interval does not improve the reputation system. However, this result subsumes that there are no errors in the feedbacks and that all nodes behave rationally. In the presence of dishonest feedbacks there are bound to be errors in identifying a honest feedback from a dishonest one. Further, our experiments show that the history component helps in stabilizing the system by avoiding transient fluctuations due to transient errors or dishonest feedbacks.

B. Yu and M. P. Singh [17] suggest refining personal opinions differently for cooperation and defection. They rely on a gossip protocol for propogating trust values. TrustGuard builds reputation based on transaction feedbacks as against using word-of-the-mouth testimonies. TrustGuard achieves this by securely tying a feedback to a transaction using transaction proofs. Further, our dependable trust model

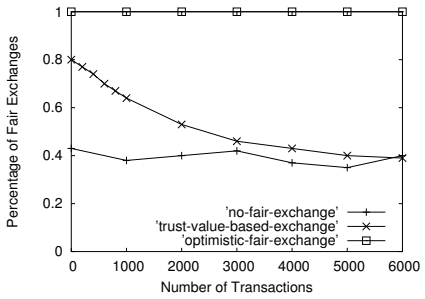

Figure 11: Fair Exchange of Transaction Proofs: Optimistic Vs Trust-Value Based Exchange Protocol

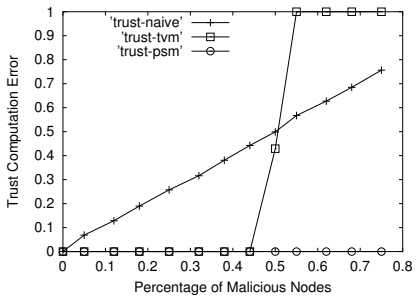

Figure 12: Robustness in Non-Collusive Setting

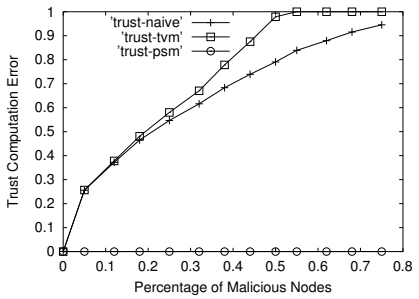

Figure 13: Robustness in Collusive Setting

is based upon the PID controller popularly used in control theory [12], as against adhoc techniques suggested in their paper.

S. K. Lam and J. Riedl [11] experimentally studied several types of shilling attacks on recommender systems. Our experiments show that TrustGuard is resistant to random shilling attacks. As a part of our future work, we hope to model and analyze different types of shilling attacks on reputation systems and enhance our algorithms to further counter them.

Fair exchange protocols [14, 2, 9] have been the prime focus of researchers working in the field of electronic commerce. Ray [14] provides a survey on fair exchange of digital products between transacting parties. They compare various algorithms including trusted third parties, true/weak fair exchanges, gradual exchanges and optimistic exchanges. In this paper, we used an optimistic fair-exchange protocol proposed by Micali [2] for fair-contract signing.

8. CONCLUSION

We have presented TrustGuard − a framework for building distributed dependable reputation management systems with the countermeasures against three detrimental vulnerabilities, namely, (i) strategic oscillation guard, (ii) fake transaction guard, and (iii) dishonest feedback guard. In TrustGuard we promote a modular design such that one could add more safeguard components, or replace the techniques for one module without having to worry about the rest of the system. The main contribution of this paper is three fold. First, we proposed to measure the trustworthiness of peers based on current reputation, reputation history and reputation fluctuation and develop formal techniques to counter strategic oscillation of malicious nodes. Second, we presented electronic fair-exchange protocol based techniques to rule out the possibility of faking transactions in the system. Third, we developed algorithms to filter out dishonest feedbacks in the presence of collusive malicious nodes. We have demonstrated the effectiveness of these techniques through an extensive set of simulation based experiments. We believe that the TrustGuard approach can efficiently and effectively guard a large-scale distributed reputation system, making it more dependable than other existing reputation-based trust systems.

9. REFERENCES

[1] K. Aberer and Z. Despotovic. Managing trust in a peer-2-peer information system. In *Proceedings of the 10th International Conference of Information and Knowledge Management*, 2001.

[2] A. Bahreman and J. D. Tygar. Certified electronic mail. In *The Internet Society Symposium on Network and Distributed Systems Security*, 1994.

[3] F. Cornelli, E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati. Choosing reputable servents in a p2p network. In *Proceedings of the 11th World Wide Web Conference*, 2002.

[4] E. Damiani, S. Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *CCS*, 2002.

[5] C. Dellarocas. The digitization of word-of-mouth: Promises and challenges of online reputation mechanism. In *Management Science*, 2003.

[6] C. Dellarocas. Sanctioning reputation mechanisms in online trading environments with moral hazard. In *MIT Sloan Working Paper No. 4297-03*, 2004.

[7] J. Douceur. The sybil attack. In *2nd Annual IPTPS Workshop*, 2002.

[8] G. S. Fishman. Discrete-event simulation. Springer Series in Operations Research.

[9] F. C. G. Holger Vogt, Henning Pagnia. Modular fair exchange protocols for electronic commerce. In *Annual Computer Security Applications Conference*, 1999.

[10] S. Kamvar, M. Schlosser, and H. Garcia-Molina. Eigentrust: Reputation management in p2p networks. In *Proceedings of the 12th WWW Conference*, 2003.

[11] S. K. Lam and J. Riedl. Shilling recommender systems for fun and profit. In *Proceedings of the 13th World Wide Web Conference*, 2004.

[12] H. Ozbay. Introduction to feedback control theory. CRC Press Inc.

[13] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, 1998.

[14] I. Ray and I. Ray. Fair exchange in e-commerce. In *ACM SIGEcomm Exchange*, 2001.

[15] M. Srivatsa and L. Liu. Vulnerabilities and security issues in structured overlay networks: A quantitative analysis. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2004.

[16] L. Xiong and L. Liu. A reputation-based trust model for peer-to-peer ecommerce communities. In *IEEE Conference on E-Commerce (CEC'03)*, 2003.

[17] B. Yu and M. P. Singh. A social mechanism of reputation management in electronic communities. In *Proceedings of the 4th International Workshop on Cooperative Information Agents*, 2000.