SecuredTrust: A Dynamic Trust Computation Model for Secured Communication in Multi-Agent Systems

Anupam Das and M. Mahfuzul Islam, *Member, IEEE,*

Abstract—Security and privacy issues have become critically important with the fast expansion of multi-agent systems. Most network applications such as pervasive computing, grid computing and P2P networks can be viewed as multi-agent systems which are open, anonymous and dynamic in nature. Such characteristics of multi-agent systems introduce vulnerabilities and threats to providing secured communication. One feasible way to minimize the threats is to evaluate the trust and reputation of the interacting agents. Many trust/reputation models have done so, but they fail to properly evaluate trust when malicious agents start to behave in an unpredictable way. Moreover, these models are ineffective in providing quick response to a malicious agent's oscillating behavior. Another aspect of multi-agent systems which is becoming critical for sustaining good service quality, is the even distribution of workload among service providing agents. Most trust/reputation models have not yet addressed this issue. So, to cope with the strategically altering behavior of malicious agents and to distribute workload as evenly as possible among service providers; we present in this paper a dynamic trust computation model called 'SecuredTrust'. In this paper we first analyze the different factors related to evaluating the trust of an agent in a and then propose a comprehensive quantitative model for measuring such trust. We also propose a novel load balancing algorithm based on the different factors defined in our model. Simulation results indicate that our model compared to other existing models can effectively cope with strategic behavioral change of malicious agents and at the same time efficiently distribute workload among the service providing agents under stable condition.

Index Terms—Multi-agent system, trust management, reputation model, load balancing, malicious behavior.

1 Introduction

In a multi-agent system, agents interact with each other to achieve a definite goal that they cannot achieve alone [1] and such systems include P2P [2–5], grid computing [6], the semantic web [7], pervasive computing [8] and MANETs. Multi-agent Systems (MASs) are increasingly becoming popular in carrying valuable and secured data over the network. Nevertheless, the open and dynamic nature of MAS has made it a challenge for researchers to operate MAS in a secured environment for information transaction. Malicious agents are always seeking ways of exploiting any existing weakness in the network. This is where trust and reputation play a critical role in ensuring effective interactions among the participating agents [9, 10]. Researchers have long been utilizing trust theory from social network to construct trust models for effectively suppressing malicious behaviors of participating agents. Trust issues have become more and more popular since traditional network security approaches such as the use of fire-wall, access control and authorized certification cannot predict agent behavior from a 'trust' viewpoint.

A reputation based trust model [11–14] collects, distributes, and aggregates feedback about participants' past behavior. These models help agents decide whom to trust, encourage trustworthy behavior, and discourage participation by agents who are dishonest. Reputation based trust models are basically divided into two category based on the way information is aggregated from an evaluator's perspective [15,16]. They are "Direct/Local experience model" and "Indirect/Global reputation model" where direct experience is derived from direct encounters or observations (first hand experience) and indirect reputation is derived from inferences based on information gathered indirectly (second-hand evidence such as by word-of-mouth). So, in case of global reputation models [17–29] an agent aggregates feedback from all the agents who have ever interacted with the target agent i.e., an agent has a view of the network which is wider than its own experience, thus enabling it to quickly converge to a better decision. However, global reputation models are much more complex to manage than local experience models as malicious agents have the opportunity to provide false feedbacks.

Most of the existing global reputation models can successfully isolate malicious agents when the agents behave in a predictable way. However, these models suffer greatly when agents start to show dynamic personality i.e., when they start to behave in a way that benefits them. These models also fail to adapt to the abrupt change in agents' behavior and as a result suffer when agents alter their activities strategically. Moreover, some of the models show

Anupam Das is currently working as a PhD student in the Department of Computer Science at University of Illinois at Urbana Champaign, Urbana, IL 61801-2302, USA and M. Mahfuzul Islam is with the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh,
e-mail: anupamdas@cse.buet.ac.bd, mahfuzul.islam@ieee.org

little effect in dealing with more complex attacks such as dishonest or unfair rating and collusion. Another aspect which is slowly becoming critical for the proper maintenance of service quality is the appropriate distribution of workload among the trusted service providers. Without a proper load balancing scheme the load at highly reputable service providers will be immense which will eventually cause a bottleneck in the system's service quality. To the best of our knowledge none of the existing trust models consider load balancing among service providers.

With these research problems in mind, we propose a feedback based dynamic trust computation model named SecuredTrust which can effectively detect sudden strategic alteration in malicious behavior with the additional feature of balancing workload among service providers. SecuredTrust considers variety of factors in determining the trust of an agent such as satisfaction, similarity, feedback credibility, recent trust, historical trust, sudden deviation of trust and decay of trust. We have used a novel policy of utilizing exponential averaging function to reduce storage overhead in computing the trust of agents. We have also proposed a new load balancing algorithm based on approximate calculation of workload present at different service providers.

The remaining part of the paper is organized as follows. Section 2 reviews the most recent and well known related works. In section 3, we formally introduce our dynamic trust computation model. We present our novel load balancing algorithm in section 4. To show the effectiveness of our model, we present simulations and comparative studies in section 5. In section 6, we discuss how our model resists the common threats confronted in multi-agent systems and how other threats can be addressed. Finally we conclude the paper in section 7.

2 Related Work

In this section we look into some of the most recent and popular research works done on reputation model. Here we discuss the key ideas of the following models- Bayesian network based trust model [18], EigenTrust [19], trust model given by Dou *et al.* [30], ReGreT [31–33], PeerTrust [20,34], FCTrust [25], SFTrust [24], FIRE [35–37], recommendation model provided by Wang *et al.* [26], trust model provided by Li *et al.* [23] and trust model proposed by Wen *et al.* [27].

Bayesian network based trust model [18] believes that trust is multi-dimensional and agents need to evaluate trust from different aspects of an agent's capability. This model uses bayesian network and bayesian probability to calculate trust. This model's main flaw lies in the authors' assumption that all the agents have identical bayesian network architecture which is unrealistic because different agents have different requirements which leads to different network architecture. In case of aggregating recommendation from other agents, this model assumes that all the agents are truthful in providing their feedbacks. This assumption is also not realistic as malicious agents will often provide false feedback to other agents to disrupt the system.

EigenTrust [19] aggregates the local trust values of all agents to calculate the unique global trust value of a given agent. An agent depends on some pre-trusted agents for trust evaluation in absence of trustworthy recommenders. Even though EigenTrust may work well in social network infrastructure where pre-trusted neighbors (agents) are likely to be trustworthy, but in case of other multi-agent systems like P2P, EigenTrust poses a few problems. Firstly, In P2P network such pre-determined trustworthy agents are not readily available. Secondly, depending on these pre-trusted agents create a vulnerability in the sense that if some of these pre-trusted agents get compromised then it will be much easier to launch a large scale malicious attack. The trust model proposed by Dou *et al.* [30] is similar to EigenTrust, but it does not consider the use of pre-trusted agents in the calculation of trust. Dou's model reduces iteration cost and punishes malicious behavior, but does not consider the punishment of dishonest recommenders.

ReGreT [31–33] is a reputation model that computes reputation from three dimensions: *individual dimension*, *social dimension* and *ontological dimension*. The individual dimension reflects an agent's own observation whereas the social dimension considers the opinion of other agents in the community. Finally, the ontological dimension considers reputation as a multi-facet concept and computes reputation from different aspect. Other models such as Repage [38] and BDI+Repage [39] consider a role based dimension in computing reputation. While ReGreT considers different dimensions in computing reputation, it does not address the collusion problem associated with computing global reputation.

PeerTrust [20, 34], computes the trustworthiness of an agent as normalized feedback weighted against the credibility of feedback originators. In PeerTrust Xiong *et al.* defined five factors in computing the trustworthiness of agents among which three factors are basic trust parameters while the remaining two are adaptive factors. PeerTrust uses personalized similarity measure to compute the credibility of recommenders and it uses this credibility measure to weight each feedback submitted by the recommenders. PeerTrust's main drawback is that it has to retrieve all the transactions within the recent time window (which may contain a large number of transactions) to compute the trust of an agent. So the trust evaluation process is both computationally and spatially expensive. Furthermore, all the transactions in the retrieved window is given equal significance but recent transactions should be given higher weight than past transactions.

FCTrust [25] uses transaction density and similarity measure to define the credibility of any recommender providing feedback as opposed to [19,30] which use global trust to weigh the quality of feedbacks. In other words, FCTrust differentiates the role of providing feedbacks from that of providing services. However, FCTrust's main drawback is that in computing direct trust it retrieves all the transactions performed within a time frame. This imposes storage overhead. Moreover, the simple averaging function used to define local trust assigns equal weight to all the transactions but realistically recent transactions should be given more importance than historical transactions. Another drawback of FCTrust is that it assigns equal degree of reward and punishment in computing similarity but the degree of punishment should be greater than that of reward.

SFTrust [24] is a double trust metric model which maintains two trust metric, one for service trust and the other for feedback trust. It separates service trust from feedback trust to take full advantage of all the agents' service capabilities even in the presence of fluctuating feedbacks. In this model, recommendation is aggregated through local broadcasting (limited by the TTL field) which can be really time consuming and moreover, recommendation should come from agents who have first hand experience [40, 41] with the respective target agent. SFTrust computes service trust as a weighted average of local trust and recommendation trust, but the weight itself is static and as a result it cannot properly accommodate the experience gained by the evaluating agent over time.

The FIRE model [35–37] believes that most of the trust information source can be categorized into four main sources- direct experience, witness information, role based rules and third party references. FIRE integrates all four sources of information and is able to provide trust metrics in a wide variety of situations. However, among the four sources two of them namely- witness information and third-party references are directly dependent on third-party agents and are therefore, susceptible to dishonesty and unfair rating. Since agents in an open MAS are self-interested, they may provide false rating to gain unwarranted trust from their partners and form collusive groups. As far as we understand FIRE thats does address this problem.

In computing the global trust of any agent, the recommendation model provided by Wang *et al.* [26] combines both local trust calculated from the agent's own experience and recommendation provided by other agents. This model uses average number of successful transaction as a metric of local trust which fails to assign time relative weight to the transactions. This model calculates global trust as a weighted combination of local trust and recommendation, but the weights assigned to these trust values are fixed and do not change over time. So, the model cannot accommodate the experience gained over time by the evaluating agent.

The trust model provided by Li *et al.* [23] address different aspects in determining the trust of an agent such as recent trust, historical trust, expected trust and confidence in its trust for other agents. However, in computing direct trust this model uses simple averaging function which fails to assign any time relative weight to the transactions. Another drawback of this model is that the formulation of both historical trust and credibility requires the storage of previous values upto a certain interval and this causes storage overhead.

Wen *et al.* [27] combines both direct trust and indirect trust in computing the trust value of an agent. In calculating indirect trust this model considers both direct and indirect recommenders. As a result two types of credibility have been defined namely direct credibility and referral credibility. In case of referral credibility it uses the transitive law of trust and as a result there exists the possibility of false error propagation. The model's main flaw lies in the assumption that all routes to target agent have equal weight which is not logical since longer the path, higher the chance that a malicious agent lies in that path. So longer paths should be given lesser weight than smaller paths.

The models defined in [18–20, 35, 30, 26, 25, 24, 23] do not address a critical aspect of trust theory which is decay of trust value with the elapse of time. Since, at present, the network is highly dynamic and unpredictable, trust values should decay as time elapse in absence of interaction. Some models [27, 42] address the above mentioned issue by incorporating their own decay function. However, these models fail to simulate real life decay function which has a small decay rate in the initial phase while displaying higher decay rate as more and more time elapse. We have incorporated such decay function in our trust model along with many other issues which have not been addressed by existing trust models.

Another aspect which is slowly becoming vital for the sustaining service quality, is the balanced distribution of workload among service providers. Almost all trust models have ignored this issue so far. In fact none of the models discussed so far address the aspect of balancing load among the trusted agents for proper maintenance of service quality. In a trust computing environment an agent with the highest trust is normally selected as service provider, so highly reputed agents handle bulk of the total service requests. This can degrade the overall service quality of the system if these highly reputed agents are assigned too much workload. So a load balancing algorithm which distributes service requests to all capable (i.e., a bit less reputable but trustworthy) agents is required to maintain a satisfactory level of service quality. We have proposed such a load balancing algorithm.

3 SecuredTrust

The main objective of this paper is to provide a dynamic trust computation model for effectively evaluating the trust of agents even in the presence of highly oscillating malicious behavior. Our model also provides an effective load balancing scheme for proper distribution of workload among the service providing agents. A number of parameters have been considered in our trust model for computing the trust of an agent. Now, some of these parameters have been previously discussed in [25,20,24,26,27,23] but none of these models can fully cope with the strategic adaptations made by malicious agents. The mathematical and logical definitions used for these parameters also cannot reflect the true scenarios faced in real life. Moreover, none of the models have considered the wide range of parameters that we have considered. In the following sections we redefine the mathematical expressions used for some of the parameters and at the same time define some new ones and then finally combine all of the parameters to present our new dynamic trust model. For the following sections we assume that agent ***p*** (called evaluator) needs to calculate the trustworthiness of agent ***q*** (called the target agent).

3.1 Satisfaction

Satisfaction function measures the degree of satisfaction an agent has about a given service provider. In other words, it keeps record of the satisfaction level of all the transactions an agent makes with another agent. However, instead of storing all of the transaction history we have defined an exponential averaging update function to store the value of satisfaction. This greatly reduces the storage overhead and at the same time assigns time relative weight to the transactions. Let, $Sat_n^t(p,q)$ represent the amount of satisfaction agent *p* has upon agent *q* based on its service up to *n* transactions in the *t-th* time interval . The satisfaction update function is defined as follows-

$$Sat_n^t(p,q) = \alpha \times Sat_{cur} + (1-\alpha) \times Sat_{n-1}^t(p,q) \quad (1)$$

$Sat_0^t(p,q) = Sat_{last}^{t-1}(p,q)$ i.e., the value of satisfaction at the start of *t*-th time interval is equal to the last computed satisfaction in the *(t-1)*-th time interval and the very initial value of satisfaction is $Sat_0^0(p,q) = 0$. Here, Sat_{cur} represents the satisfaction value for the most recent transaction and we have used a feedback based system where an agent rates other agent's service quality according to the following function [1]-

$$Sat_{cur} = \begin{cases} 0, & \text{if transaction is fully unsatisfactory} \\ 1, & \text{if transaction is fully satisfactory} \\ \in (0,1), & \text{otherwise} \end{cases} \quad (2)$$

The weight α changes based on the accumulated deviation $\xi_n^t(p,q)$.

$$\begin{aligned} \alpha &= threshold + c \times \frac{\delta_n^t(p,q)}{1+\xi_n^t(p,q)} && (3) \\ \delta_n^t(p,q) &= |Sat_{n-1}^t(p,q) - Sat_{cur}| && (4) \\ \xi_n^t(p,q) &= c \times \delta_n^t(p,q) + (1-c) \times \xi_{n-1}^t(p,q) && (5) \end{aligned}$$

$\xi_0^t(p,q) = \xi_{last}^{t-1}(p,q)$ and $\xi_0^0(p,q) = 0$. Here c is some user defined constant factor which controls to what extent we will react to the recent error ($\delta_n^t(p,q)$). So, if we increase the value of c then we give more significance to the recent deviation than the accumulated deviation and vice versa. Again we can see that as recent error ($\delta_n^t(p,q)$) increases so does α which means that recent satisfaction is given higher weight than accumulated satisfaction (i.e., higher significance is given to the recent service feedback). The $threshold$ represents a threshold which is used to prevent α from saturating to a fixed value. Initial value of α is set to 1 (we consider only the current transaction at the very beginning as we have no knowledge of any previous transaction) and $threshold$ is set to 0.25.

1. We could compute satisfaction by considers multiple attributes (a_i) of a given transaction. Then satisfaction Sat could be computed using the function: $Sat = \sum_{i=1}^{N} w_i.a_i$, where there are N attributes for evaluating a transaction and w_i represent their corresponding weight.

3.2 Similarity

Similarity metric defines to what extent two agents are alike. We have computed similarity by determining the personalized difference in satisfaction rating over the common set of interacted agents and have then used the computed difference rating to define the degree of similarity. Let, $IS(p)$ represent the set of agents with whom agent p has made interaction. Then $CIS(p,q) = IS(p) \bigcap IS(q)$ denotes the set of agents with whom both agent p and q have made interaction.

The personalized difference in satisfaction rating between agent p and q denoted as $Diff_n^t(p,q)$ is defined as follows-

$$Diff_n^t(p,q) = \sqrt{\frac{\sum_{x \in CIS(p,q)} (Sat_n^t(p,x) - Sat_n^t(q,x))^2}{|CIS(p,q)|}} \quad (6)$$

To measure the similarity between agent p and q ($Sim_n^t(p,q)$), agent p first compares $Diff_n^t(p,q)$ with the similarity deviation constant (τ) and then updates similarity according to the following function-

$$Sim_n^t(p,q) = \begin{cases} Sim_{n-1}^t(p,q) + \frac{1-Sim_{n-1}^t(p,q)}{\mu}, \\ \qquad \text{if } Diff_n^t(p,q) < \tau \\ Sim_{n-1}^t(p,q) - \frac{Sim_{n-1}^t(p,q)}{\psi}, \text{else} \end{cases} \quad (7)$$

where μ, ψ represent the reward and punishment factor respectively and both of them can be changed dynamically depending on the system. From equation (7) we see that as $Diff_n^t(p,q)$ increases beyond τ, similarity $Sim_n^t(p,q)$ decreases. Here $\mu, \psi \in \Re$ and we must make sure that $\mu > \psi$ because the degree of punishment should be greater than that of incentive. By doing so we incorporate the principle of "slow rise and quick decline". If however, $|CIS(p,q)| = 0$ then we set $Diff_n^t(p,q) = 0$ and $Sim_n^t(p,q) = 0.5$ (default value).

3.3 Feedback Credibility

Feedback credibility is used to measure the degree of accuracy of the feedback information that the recommending agent provides to the evaluator. Normally it is assumed that good agents always provide true feedback and malicious agents provide false feedback. However, this is not always the real scenario as good agents might provide false feedbacks to their competitors and malicious agents might occasionally provide true feedbacks to hide their real nature. So feedback credibility is needed to determine the reliability of the feedback. During trust evaluation, feedbacks provided by agents with higher credibility are trustworthier, and are therefore weighted more than those from agents with lower credibility. Let, $FCre_n^t(p,q)$ present the feedback credibility of agent q from agent p's perspective.

$$FCre_n^t(p,q) = \begin{cases} 1 - \frac{\ln(Sim_n^t(p,q))}{\ln \theta}, & \text{if } Sim_n^t(p,q) > \theta \\ 0, & \text{else} \end{cases} \quad (8)$$

where $\theta = 0.01$ represents the lowest allowed value of similarity. As we can see from equation (8), credibility is a direct logarithmic function of similarity for its slow rise to the highest attainable value. This implies that agents with higher similarity with respect to the evaluating agent have higher feedback credibility.

3.4 Direct Trust

Direct trust also known as local trust represents the portion of trust that an agent computes from its own experience about the target agent. Let, $DT_n^t(p,q)$ represent the direct trust that agent *p* has upon agent *q* up to *n* transactions in the *t-th* time interval. We have used the satisfaction measure to define direct trust as follows-

$$DT_n^t(p,q) = Sat_n^t(p,q)) \tag{9}$$

Since satisfaction is computed by an agent's own experience, we are using it as local trust. So, if agent *q* provides good service then agent *p* will rate it with a high satisfaction value and as a result agent *q* will obtain a high local trust rating from agent *p*'s perspective.

3.5 Indirect Trust

Indirect trust also referred as recommendation is computed from the experience of other agents. An agent utilizes the experience gained by other agents in the system to make effective transaction decisions especially when it has no or very little experience with the given target agent. To do so, an agent requests other agents to provide recommendation about the target agent. The evaluating agent then aggregates recommendation from other agents along with the feedback credibility of the recommenders. Let, $IT_n^t(p,q)$ represent the indirect trust that agent *p* computes about agent *q*.

$$IT_n^t(p,q) = \begin{cases} \frac{\sum_{x \in W-\{p\}} FCre_n^t(p,x) \times DT_n^t(x,q)}{\sum_{x \in W-\{p\}} FCre_n^t(p,x)}, & \\ & \text{if } |W-\{p\}| > 0 \\ 0, & \text{if } |W-\{p\}| = 0 \end{cases} \tag{10}$$

Here $W = TS(q)$, represents the set of agents who have ever interacted with agent *q*. From equation (10) we see that indirect trust is computed as weighted average of recommendation from different recommenders where the weights represent the feedback credibility of the recommenders. The recommendation made by a recommender is its own experience i.e., its own direct trust (DT).

3.6 Recent Trust

Recent trust reflects only the recent behaviors. We have defined recent trust as a weighted combination of direct and indirect trust. Direct trust is given higher weight as the evaluating agent performs more and more interactions with the target agent, i.e., the evaluator becomes more confident about its own experience than taking recommendation from others. Let, $RT_n^t(p,q)$ represent the recent trust that agent *p* has upon agent *q*.

$$RT_n^t(p,q) = \beta \times DT_n^t(p,q) + (1-\beta) \times IT_n^t(p,q) \tag{11}$$

where β represents the weight of direct trust which can be calculated as follows-

$$\beta = \frac{I^t(p,q)}{I^t(p,q) + M^t(p,q)} \tag{12}$$

$$M^t(p,q) = \frac{\sum_{x \in W-\{p\}} FCre_n^t(p,x) \times I^t(x,q)}{|W-\{p\}|} \tag{13}$$

Here $W = TS(q)$, represents the set of agents who have ever interacted with agent *q* and $I^t(p,q)$ represents the number of interactions agent *p* has conducted with agent *q* in the *t-th* interval. So, $M^t(p,q)$ represents the mean number of interactions that other agents (agents other than *p*) have conducted with agent *q*. We have weighted the interaction count (I) with feedback credibility ($FCre$) of the recommenders in computing $M^t(p,q)$. We can see that as $I^t(p,q)$ increases (compared to $M^t(p,q)$) the value of β also increases signifying that as the evaluator becomes more experienced, it tends to rely more on its own judgement. If $|W-\{p\}| = 0$ then we set $M^t(p,q) = 0$ and if $I^t(p,q)+M^t(p,q) = 0$ then we set $\beta = 0.5$ (default value).

3.7 Historical Trust

Historical trust is built from past experience and it reflects long term behavioral pattern. With the elapse of time recent trend becomes historical trend, and as a result we have defined historical trust by using an exponential averaging update function. By using an exponential averaging update function we not only reduce the storage overhead associated with storing the previous recent trusts but also assign time relative weights to all the previous values. Let, $HT_n^t(p,q)$ represent the historical trust that agent *p* has about agent *q*.

$$HT_n^t(p,q) = \frac{\rho \times HT_{n-1}^t(p,q) + RT_{n-1}^t(p,q)}{2} \tag{14}$$

where $\rho (0 \leq \rho \leq 1)$ is the forgetting factor (discounting older experiences) and $HT_0^0(p,q) = 0$. With historical trust present malicious agents cannot suddenly forget their past and start behaving good. In other words, since we are keeping track of an agent's past behavior it cannot deceive other agents into believing that it is a good agent by just behaving cooperatively in the recent transactions. For an agent to be considered as good, it has to perform in a cooperative manner for a significantly large number of transactions so that its recent trend can replace most of its historical trend.

3.8 Expected Trust

Expected trust reflects expected performance of the target agent and it is deduced from both recent and historical trust. In other words, we are combining both recent trend and historical trend to get a prediction of the future trend. Let, $ET_n^t(p,q)$ represent the expected trust of agent *q* from agent *p*'s perspective. Expected trust is calculated by the following equation-

$$ET_n^t(p,q) = \begin{cases} 0, & \text{if neither RT nor HT is available} \\ \eta RT_n^t(p,q) + (1-\eta)HT_n^t(p,q), \\ \quad \text{if either RT and/or HT is available} \end{cases} \tag{15}$$

Initially η is set to 0.5, but η adjusts dynamically based on the difference of recent and historical trust (deviation factor ε). Here we are encouraging benevolent behavior by agents in the recent interactions by increasing η when recent trust exceeds historical trust by a given threshold (ε). This also provides agents with the opportunity to improve their rating after network failure/congestion (provides a second chance opportunity).

$$\eta = \begin{cases} \eta + 0.1, & \text{if } RT_n^t(p,q) - HT_n^t(p,q) > \varepsilon \\ \eta - 0.1, & \text{if } RT_n^t(p,q) - HT_n^t(p,q) < -\varepsilon \\ \eta, & \text{if } -\varepsilon < RT_n^t(p,q) - HT_n^t(p,q) < \varepsilon \end{cases} \tag{16}$$

As we see from equation (16) that when recent trust exceeds historical trust by ε the value of η increases by 0.1 (empirically tuned) and as a result the contribution of recent trust to expected trust increases i.e., the expected trust reflects more of the recent trust than the historical trust and vice versa. So, we are actually opting to take an optimistic approach where we are considering the best possible outcome for an agent. In other words, we are highlighting or emphasizing the best possible behavior expected from an agent. Now, by controlling the value of ε we can determine how quickly an agent can recover from its historical trend. However, the value of ε should not be set very small as malicious agents might then try to exploit this by quickly recovering from their past mischiefs.

3.9 Decay model

Due to the highly dynamic nature of agents, trust should attenuate with the elapse of time in absence of interaction. If an agent remains idle for a long time i.e., if it does not interact with the network for a long period, the evaluation of its trust should degrade gradually. Since in our model direct trust depends on satisfaction and indirect trust is calculated from the direct trust of recommenders we apply a decay function on satisfaction metric. The decay function is given as follows-

$$\widehat{Sat}_n^t(p,q) = Sat_n^t(p,q)e^{-\lambda \Delta t} \tag{17}$$

$$\Delta t = t_{current} - t_{previous} \tag{18}$$

where $\widehat{Sat}_n^t(p,q)$ represents the value of satisfaction after decay. Here, λ is the decay constant and it controls how quickly the value will diminish to zero. Δt represents the interval between the current interaction and the last interaction. So, as Δt increases, i.e., as the interval between successive interactions increases the trust value of the target agent decreases from the perspective of the evaluating agent. By including the time decay model we establish the principle "the more recent the transaction the more reliable it is".

3.10 Deviation Reliability

Deviation reliability is a measure of how much deviation we are willing to tolerate. Malicious agents sometimes strategically oscillate between raising and milking their reputation which seriously affects the performance of the network. So, some form of measurement is required to handle such scenario. Deviation reliability handles such trust fluctuation. To record the sudden misuse of trust by agents, we introduce the component *accumulated trust fluctuation* (denoted as $ATF_n^t(p,q)$).

$$ATF_n^t(p,q) = \begin{cases} ATF_{n-1}^t(p,q) + \frac{RT_n^t(p,q) - HT_n^t(p,q)}{\omega}, \\ \quad \text{if } RT_n^t(p,q) - HT_n^t(p,q) > \varphi \\ ATF_{n-1}^t(p,q) + HT_n^t(p,q) - RT_n^t(p,q), \\ \quad \text{if } RT_n^t(p,q) - HT_n^t(p,q) < -\varphi \\ ATF_{n-1}^t(p,q), \quad \text{otherwise} \end{cases} \tag{19}$$

where φ represents the tolerated margin of error in the evaluation of trust and ω ($\omega > 1$) represents the punishment factor for sudden rise in trust. From equation (19) we see that we are considering both sudden rise and fall of trust by agents. However, we are penalizing lesser for sudden rise (through controlling the value of ω) since we are encouraging agents to raise their trust through benevolent interactions. This is evident from the above equation because with $\omega > 1$ the contribution to accumulated trust fluctuation will be less than $RT_n^t(p,q) - HT_n^t(p,q)$ for sudden rise where it is exactly $HT_n^t(p,q) - RT_n^t(p,q)$ for sudden fall. Initial value of accumulated misused trust $ATF_0^0(p,q) = 0$.

Deviation reliability uses the accumulated trust fluctuation metric to measure the deviation in agent behavior. Deviation reliability (denoted $DR_n^t(p,q)$) is defined by the following equation-

$$DR_n^t(p,q) = \begin{cases} 0, & \text{if } ATF_n^t(p,q) > maxAT \\ \cos(\frac{\Pi}{2} \times \frac{ATF_n^t(p,q)}{maxATF}), & \text{otherwise} \end{cases} \tag{20}$$

where $maxATF$ represents the maximum tolerable trust fluctuation. So from equation (20) we can see that the deviation reliability follows the property of a cosine curve when the accumulated trust fluctuation is less than the given threshold $maxATF$ and it becomes zero when the accumulated trust fluctuation exceeds $maxATF$. We have used a cosine function for defining deviation reliability since the cosine function has a low degradation rate in the initial stage and as more and more trust fluctuation occurs the degradation rate increases. Here, the initial value of deviation reliability is $DR_n^t(p,q) = 1$.

3.11 Overall Trust Metric

This is the actual trust value used in prioritizing all agents. It is computed from expected trust and deviation reliability. Let, $Trust_n^t(p,q)$ represent the final trust value agent p places upon agent q.

$$Trust_n^t(p,q) = ET_n^t(p,q) \times DR_n^t(p,q) \tag{21}$$

From the equation it is evident that agents with high expected trust values but with low deviation reliability will eventually have low overall trust value. Otherwise stated, agents that strategically oscillate between building and milking trust will have low trust value due to low deviation reliability. For an agent to attain a high overall trust value it must behave cooperatively and at the same time must not show major trust fluctuation. Therefore, an agent will use equation (21) to select the target agent with the highest trust value as this metric combines all the factors we have discussed so far.

4 Load Balancing Among Agents

In this section we propose an algorithm for balancing loads among the trusted agents. For selective scenario, we first compute the trust of agents who respond to a transaction request and then we select the agent with the highest trust value. However, in this scenario the agent with the highest trust value will have immense workload while other capable agents with slightly lower reputation will have considerably less workload. The problem that will arise from this disproportionate allocation of workload is that the quality of service will fall greatly due to the heavy workload present at the highly trusted agents. So a load balancing algorithm is required for the sustainability of good service quality.

In our load balancing algorithm we either calculate a heuristic value of workload and choose the agent with the smallest load or make a probabilistic choice based on the computed trust value of agents. We start our load balancing algorithm by first classifying the responders (agents that respond to a transaction request) into two groups namely- good service providers (G) and unknown service providers (U) based on a threshold value of trust (γ). We then first seek to choose an agent from G by computing an approximate value (heuristic value) of load present at each responders in G. Sorting the responders in increasing order of load we take the responder with the smallest workload. In case of no responders being present in the class G we select an agent from U either probabilistically based on its trust value or randomly.

We compute the approximate load present at the good service providers using the following equation-

$$N^t(p,q) = I^t(p,q) + \sum_{x \in W-\{p\}} FCre^t_n(p,x) \times I^t(x,q) \tag{22}$$

Here $W = TS(q)$, represents the set of agents who have ever interacted with agent q and $I^t(p,q)$ represents the number of interactions agent p has conducted with agent q. So, $N^t(p,q)$ represents the total number of direct (from own experience) and indirect (from recommenders) interactions considered during the computation of trust from agent p's perspective. We finally sort the good service providers in increasing order of load and select the one with the smallest workload as the corresponding service provider to a transaction request.

However, all the service providing agents might be classified as unknown service providers in the initial stage of the system as their trust values might not have reached a stable state due to the lack of transactions. In such case, we choose an agent based on the following probability measure-

$$Prob(p,q) = \begin{cases} \frac{Trust^t_n(p,q)}{\sum_{x\in U} Trust^t_n(p,x)}, \\ \quad \text{if } \sum_{x\in U} Trust^t_n(p,x) \neq 0 \\ \text{randomly select any agent,} \\ \quad \text{else} \end{cases} \tag{23}$$

In case of choosing an agent from U, we see from equation (23) that higher the trust value the more chance an agent has in being selected as a service provider. In other words, the probability of being selected is directly proportional to an agent's trust value which is logical as trust values are an indication of agents' service capabilities. However, in case the trust values of all agents in U are zero we have no other option but to select an agent randomly as there are no useful metrics to help us make any predictions. In such case, we cannot guarantee effective selection of responders.

Pseudo-code of the load balancing algorithm is given in Algorithm 1.

Algorithm 1 Selection of service providing agent(p, S)

```
Input: Evaluating agent p and the set of agents responding to a service request S
Output: Service providing agent q
for each $x \in S$ do
    compute $Trust(p,x)$
    if $Trust(p,x) > \gamma$ then
        $G \leftarrow G \cup \{x\}$
    else
        $U \leftarrow U \cup \{x\}$
    end if
end for
if $G \neq \emptyset$ then
    for each $x \in G$ do
        compute load $N(p,x)$
    end for
    sort $G$ in increasing order of load $N$
    return agent q with the smallest load $N$
else
    $Total\_trust \leftarrow 0$
    for each $x \in U$ do
        $Total\_trust \leftarrow Total\_trust + Trust(p,x)$
    end for
    if $Total\_trust > 0$ then
        for each $x \in U$ do
            compute $Prob(p,x)$
        end for
        return agent q with probability $Prob(p,q)$
    else
        return any agent q randomly
    end if
end if
```

5 Experimental Evaluation

This section evaluates SecuredTrust's performance and shows its effectiveness under different adversarial strategy. We have carried out our experiment to achieve four main objectives. Firstly, we evaluate its accuracy in terms of trust computation in the presence of malicious agents under two settings. The second experiment shows how quickly it adapts to strategically oscillating behavior. In the third set of experiments we demonstrates the robustness of SecuredTrust compared to other existing trust models under different scenarios. Lastly, we show its effectiveness under the load balancing scheme.

5.1 Simulation Setup

We have developed our simulation in Java using JBuilder [43] and the discrete event simulation toolkit SimJava [44, 45]. From the hardware's point of view we used Pentium-4(P4) 3.00 Ghz processor with 2GB RAM. This section describes the general simulation setup including the environment setting, agent's behavioral pattern, transaction setting and performance evaluation index.

5.1.1 Environment Setting

Our simulated environment contains N agents. N is set to 100 in almost all the experiment. However, in one experiment we have varied the value of N to show the scalability of the trust model. It was evident from the experiment that the variation in N did not affect the performance of the trust model and as such N was set to 100 for most of the experiments. The agents are of mainly two types- good and malicious. Good agents cooperate in providing both good service and honest feedback. In contrast, malicious agents are opportunistic in the sense that they cheat whenever it is advantageous for them. Malicious agents provide both ineffective service and false feedback. The percentage of malicious agents in the environment is denoted by the parameter *malicious_per* which is varied in different experiments.

5.1.2 Agent's Behavioral Pattern

The behavioral pattern of good agents is quite easy to simulate as they provide good service and honest feedback. However, it is challenging to simulate an agent's malicious behavior realistically. We mainly study three behavioral patterns namely- noncollusive, collusive and strategically altering. In noncollusive setting malicious agents cheat during transaction and give false feedback to other agents i.e., they rate good agents poorly while rating malicious agents highly. The collusive setting is similar to the noncollusive setting with one additional feature that malicious agents form a collusive group and deterministically help each other by performing numerous fake transactions to boost their own rating while disparaging other good agents. We have used the parameter *collusion* to denote the percentage of malicious agents forming a collusive group. In the strategically altering setting a malicious agent may occasionally decide to cooperate in order to confuse the system. We use the parameter *malicious_res* to model the rate of dishonest feedback by a malicious agent. In this case, other agents are commonly fooled into thinking that the malicious agent is actually a good agent.

5.1.3 Transaction Setting

Three types of transaction setting are simulated namely, random setting, trust prioritized setting and load balanced setting. In the random setting, agents randomly interact with each other. In the trust prioritized setting an agent first initiates a transaction request. Against each request certain percentage of agents respond. The response percentage is controlled by *response_rate* parameter. The initiating agent then sorts the responders based on their trust value and selects the agent with the highest trust value to perform the desired transaction. Finally, in the load balancing scheme a service provider with least amount of workload is selected.

Table 1, summarizes the different parameters related to the environmental setting and trust computation. The table also lists the default values of the different parameters used. These default values have been empirically tuned.

5.1.4 Performance evaluation index

To compare the performance of SecuredTrust with other existing trust models we use a evaluation index named, successful transaction rate (STR). STR is described as the ratio of the number of successful transactions to the total number of transactions. Since computed trust values may range differently for different trust models, other form of evaluation index such as trust computation error is not suitable for comparison. It really does not matter what range of trust value we assign to an agent, what matters is how efficiently the model can filter out malicious agents based on the calculated trust value. In other words, the relative ranking of agents based on their trust values is comparable and thats why we only compute STR for comparison with other models. We determine STR against the variation of *malicious_per*, *malicious_res* and *collusion*. All experimental results are averaged over 30 runs.

5.2 Trust Computation Accuracy

The objective of this set of experiments is to show the effectiveness of SecuredTrust against different malicious behavior. The experiment starts as agents randomly start interacting with each other. After each agent performs on average 500 transactions, a good agent is randomly selected to compute the trust value of all the other agents. The experiments are performed under both noncollusive and collusive setting as described in the previous section. The trust computation error is computed by taking the root-mean-square (RMS) of the computed trust value for all the agents against their actual likelihood of performing a satisfactory transaction, which is 1 for good agents and 0 for malicious agents. So a low RMS value indicates better performance.

In the first experiment we vary the percentage of malicious agents (*malicious_per*) in the system while keeping

TABLE 1
Simulation Parameter settings

	Parameter	Description	Default value
Environment Setting	N	# of agents in the system	100
	malicious_per	% of agents malicious in the system	40%
	malicious_res	% of time a malicious agents gives false feedback	100%
	response_rate	% of agents who respond to a transaction request	5%
	collusion	% of malicious agents forming a collusive group	0%
Trust Computation Setting	α	contribution factor for recent satisfaction	1
	threshold	minimum threshold of α	0.25
	c	user defined constant	0.9
	β	weight of direct trust in computing recent trust	0.5
	ρ	forgetting factor	0.9
	η	weight of recent trust in computing expected trust	0.5
	ε	deviation factor for expected trust	0.3
	μ	reward factor for similarity	20
	ψ	punishment factor for similarity	4
	τ	similarity deviation	0.25
	θ	lowest allowed value of similarity	0.01
	λ	decay constant or decay rate	0.05
	φ	tolerated margin of error	0.25
	ω	punishment factor for sudden rise in trust	2
	$maxAT$	threshold for accumulated misused trust	10
	γ	threshold for credibility/trust value	0.8

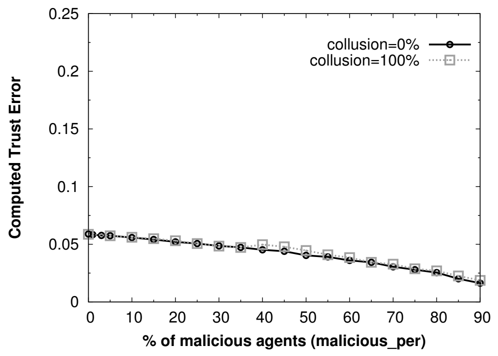

Fig. 1. Trust computation error (RMS) with respect to percentage of malicious agents (*malicious_per*).

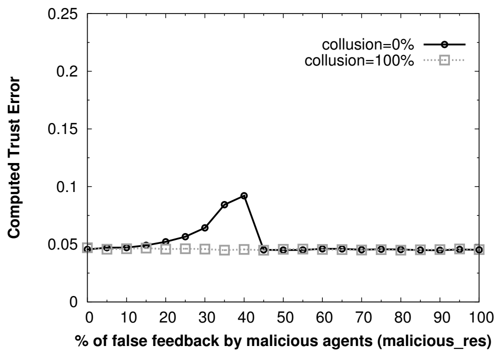

Fig. 2. Trust computation error (RMS) with respect to percentage of false response by malicious agents (*malicious_res*).

malicious_res to 100% (i.e., malicious agents give false feedback in every interaction). Fig. 1 represents the trust computation error with respect to *malicious_per* under two settings. For noncollusive (*collusion* set to 0%) setting in Fig. 1, we see that SecuredTrust remains more effective in the presence of large percentage of malicious agents. The reason behind this is that feedback credibility metric effectively filters out dishonest feedbacks submitted by malicious agents. For the collusive setting in Fig. 1, we set *collusion* to 100% and again we observe that our model efficiently discards the dishonest feedbacks submitted by the collusive group. This certifies that our similarity measure appropriately computes feedback credibility of recommenders providing recommendation.

In the second experiment we vary *malicious_res* while setting *malicious_per* to 40%. Fig. 2 represents the trust computation error in both noncollusive and collusive setting. In the noncollusive setting where *collusion* is set to 0%, we see that the error is slightly high when *malicious_res* varies from 30% to 45% signifying that malicious agents can confuse the system a little when they oscillate between good and malicious nature alternatively. For collusive setting where *collusion* is set to 100%, we see a better result in the presence of collusion. This signifies that SecuredTrust can successfully discard false rating provided by collusive groups.

In the third experiment we vary *collusion* while setting *malicious_per* to 40% and *malicious_res* to 100%. Again from Fig. 3 we see that our model effectively discards the impact of collusion by leveraging our sensitive feedback credibility metric. Here, in fact false feedback ratings come from agents (namely malicious agents) with very low credibility and as a result they cannot influence the overall trust value.

5.3 Handling Dynamic Personality of Agents

So far, we have considered more or less fixed personality of agents. The objective of this experiment is to show how SecuredTrust handles dynamic change in agent behavior. We have already showed SecuredTrust's effectiveness against filtering out dishonest feedback submitted by malicious

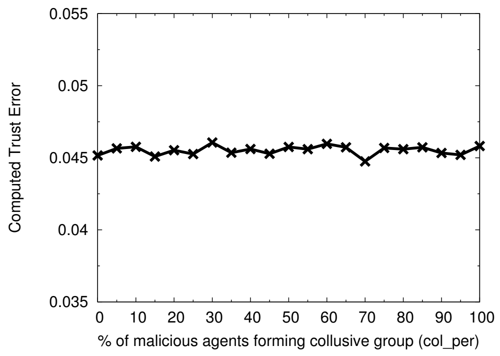

Fig. 3. Trust computation error with respect to percentage of malicious agents forming collusive group (*collusion*).

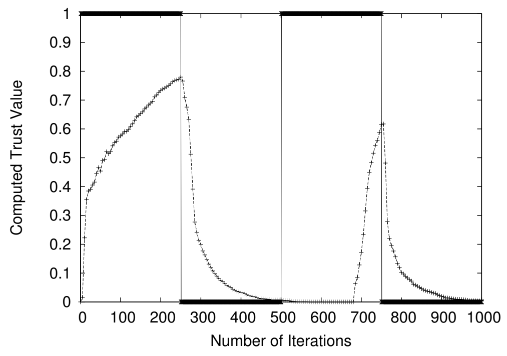

Fig. 4. Effectiveness against dynamic personality.

agents, so in this experiment we concentrate on alternating behavior of malicious agents. Here we simulate the pattern where a malicious agent first builds up its reputation and then milks the built reputation and finally tries to build its reputation back again i.e., the agent oscillates between building and milking reputation. For testing such scenario we simulate an environment which contains all good agents except for only one malicious agent with dynamic personality. The experiment proceeds as agents randomly perform transactions with each other and a good agent is selected to determine the trust value of the malicious agent periodically. In this experiment the malicious agent performs a total of 1000 interactions which are equally divided into four consecutive slots . The malicious agent then oscillates between good and malicious nature from one slot to the next starting with good nature.

Fig. 4 shows the computed trust value of the malicious agent under altering behavioral pattern. We see that SecuredTrust quickly responds to the sudden fall of performance by malicious agent and thus prevents it from utilizing its built reputation. The sharp fall in the curve signifies this. Once the trust value diminishes to zero it requires a significant number of consecutive good services for its trust value to rise again, i.e., it must give proof of its cooperative nature. From Fig. 4 we see that in spite of the good nature of the malicious agent in the third slot its trust value rises very late and even in that case it does not rise to the previous value. So, the cost of rebuilding reputation is actually higher than that of milking it. That is the model successfully incorporates the principle “quick decline and slow rise of trust value”.

5.4 Comparison with other Trust Models

In this set of experiments we will demonstrate the efficiency of SecuredTrust against other existing trust models. In these experiments an agent first computes and compares the trust values of the responding agents (i.e., agents who respond to a transaction request) and chooses the agent with the highest trust value for interaction. A transaction is successful if the participating agent is cooperative i.e., if it is a good agent. In all the experiments, we compute STR as the evaluation criterion under different scenarios. The experiment proceeds in iterations where in each iteration each agent in the system initiates one transaction. We have discarded the transactions initiated by malicious agents from the calculation of STR. We execute a total of 100 iterations in one experiment and compute the average STR. Since the responders to a transaction request is generated at random, we take the mean (along with the 95% confidence interval) of 30 experiments for each scenario. We compare our model with SFTrust [24], FCTrust [25], P2P recommendation trust model (for short we will use Reco-Trust) [26], trust model of users’ behavior (for short we will use User-Trust) [27], dynamic trust model for multi-agent systems (for short we will use MAS-Trust) [23] and PeerTrust [20].

First, we calculate STR against the variation of percentage of malicious agents, *malicious_per* while keeping *malicious_res* to 100% and *collusion* to 0%. As from Fig. 5 we see that both SecuredTrust and PeerTrust show superiority over the remaining trust models as the amount of malicious agents in the network increase beyond 40%. Due to the ease of accessibility, networks today are home to a significantly large number of malicious agents, especially the internet holds great threats as it teems with malicious agents (in the form of botnets [46–48]). In other words, threats and risks are implicitly increasing as network applications are widening. So, in such networks SecuredTrust would be the best option.

In the next experiment we want to observe the impact of collusion on STR. So, for this experiment we set *malicious_per* to 60% because as the number of malicious agents increase their collusive impact becomes greater. We also set *malicious_res* to 100%. Fig. 6 represents the computed STR against *collusion*. Due to the experimental randomness, the gradient of the curves may vary from experiment to experiment. In Fig. 6 we see that SFTrust, MAS-Trust and User-Trust have negative gradient so in their case STR is actually decreasing as collusive group size is increasing. The remaining four trust models remain unaffected by collusion but we see that again, SecuredTrust and PeerTrust show superiority over others. The main reason behind this is the feedback credibility measure which filters out false feedbacks. Here false high ratings come

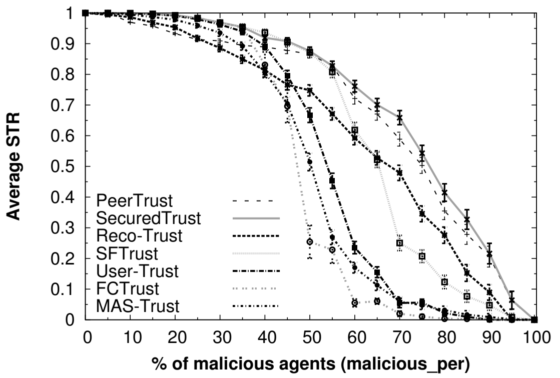

Fig. 5. Comparing SecuredTrust with other models in terms of average STR (with 95% confidence interval) against *malicious_per*.

from agents with low feedback credibility as a result they have no impact on STR. The low credibility itself results from the personalized similarity measure. In order to attain a high credibility malicious agents would have to provide honest feedback which goes against their true nature.

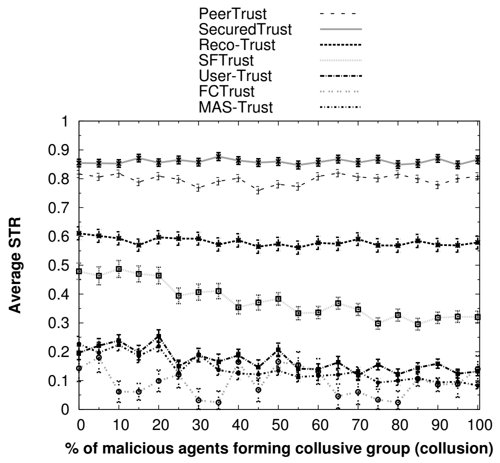

Fig. 6. Comparing SecuredTrust with other existing trust models in terms of average STR (with 95% confidence interval) against *collusion*.

In the third experiment we analyze the impact of *malicious_res* on STR. As we saw in Fig. 2 that the malicious agents tend to fool other agents by oscillating between good and malicious nature. In this experiment we test two scenarios with *malicious_per* set to 40% and 60% respectively while *collusion* is set to 0% in both the cases. Fig. 7 represents the computed STR against *malicious_res*. From the figures we see that SecuredTrust out performs all other trust models significantly and in these cases PeerTrust suffers the most. This is because our model keeps track of sudden rise and fall of trust by agents and penalizes any agent showing frequent trust fluctuations. While other models fail to identify the strategic alternations made by malicious agents, our model quickly distinguishes such alternations through our deviation reliability metric (see equation (20)). Thus, SecuredTrust can successfully restrain strategically altering behavior of malicious agents.

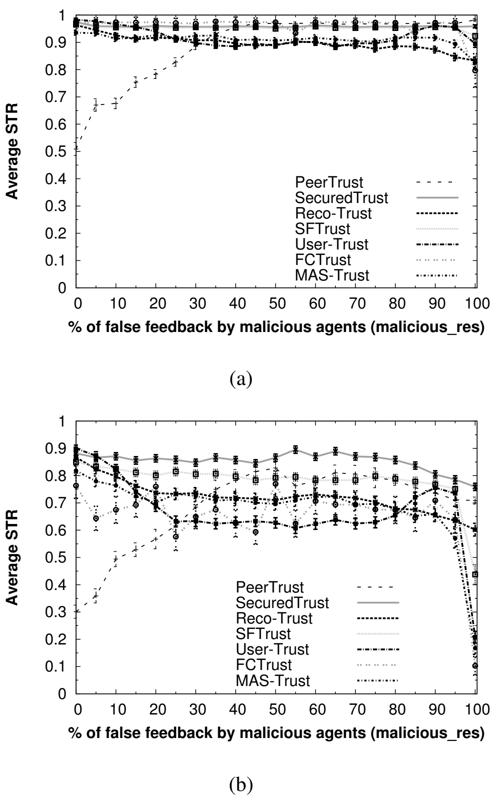

Fig. 7. Comparing SecuredTrust with other trust models in terms of average STR (with 95% confidence interval) against *malicious_res* (a) 40% malicious agents (b) 60% malicious agents.

In the next experiment we determine the number of times malicious agents are selected as service providers in the presence of oscillating malicious behavior. Here we run the experiment for a total of 500 iterations with *malicious_res* set to 50%, *malicious_per* set to 40% and *collusion* set to 0%. However, we divide the 500 iterations into four equal slots, so each slot contains 125 iterations. Malicious agents oscillate between good and malicious nature from one slot to the next starting with good nature. Then we compute the number of times malicious agents are selected as service providers to transactions initiated by only good agents. From Fig. 8 we see that in the initial slot malicious agents are selected numerous times. This is understandable because in the first slot they start off by behaving good so there is no reason to reject them, but in the following slots this number should decline as we now know their true nature. We see that our trust model performs best in isolating the malicious agents and thus reducing unauthentic transactions compared to other models. The reason behind our model's superiority is that we keep track of sudden rise and fall of trust with the intent to heavily punish any agents showing such trust fluctuations.

Finally, we compare the computation time required by the different trust models. For this purpose we compute the

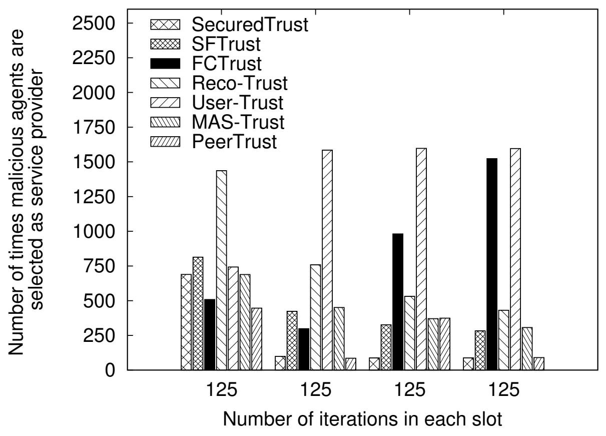

Fig. 8. Comparing SecuredTrust with other trust models in terms of the number of times malicious agents are selected as service providers.

amount of times it takes for the trust models to execute 200 iterations with *malicious_per* set to 50%, *collusion* set to 0% and *malicious_res* set to 100%. We take the average of 30 runs. From Fig. 9 we see that PeerTrust requires the largest amount of time while FCTrust requires the lowest. Our trust model requires on average 1.2 seconds to execute 200 iterations which is slightly higher than some of the remaining trust models. This is understandable as we have considered more components compared to the other trust models. For example we have considered sudden rise and fall of trust as well as historical trend of agent behavior all of which are not considered by other models. As a result these trust models fail to effectively filter out malicious agents when they start to show oscillating behaviors. So, we are sacrificing a very small amount of computational overhead for the sack of better resilience.

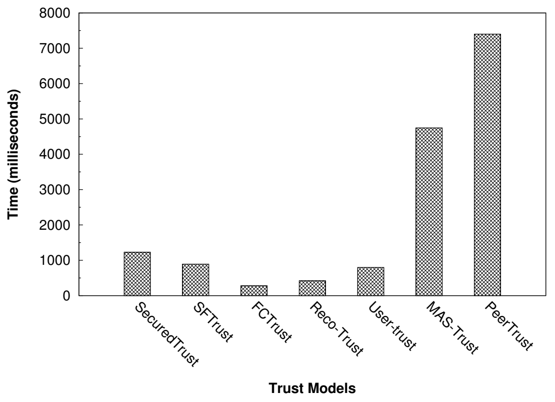

Fig. 9. Comparing SecuredTrust with other models in terms of the computational time required to execute 200 iterations.

5.5 Impact of Load Balancing

In this section we analyze the impact of our load balancing algorithm on the service providers. For this experiment we set *malicious_per* to 50%, *collusion* to 0% and *malicious_res* to 100%. We assume that fixed 10% of the agents act as service providers to different transaction requests and among these service providers 60% of them are good agents while the remaining 40% are malicious. Now, we set *response_rate* to 50% i.e., 50% of the service providers respond to a transaction request. So we can see that at least one good service provider responds to a transaction request. In this experiment we perform 500 iterations where in each iteration each agent performs one transaction. In Fig. 10 we present the workloads present at good service providers under both load-balancing and non load-balancing scheme. It is evident from Fig. 10 that under load balancing scheme the total workload is evenly distributed among the service providers. Fig. 11 presents the average STR under both schemes and it is observable that the performance remains same under both schemes. Since at least one good service provider responds to a transaction request, the STR in both cases are very high which is expected. So, we can see that our load balancing algorithm can successfully distribute workload among the agents without compromising performance.

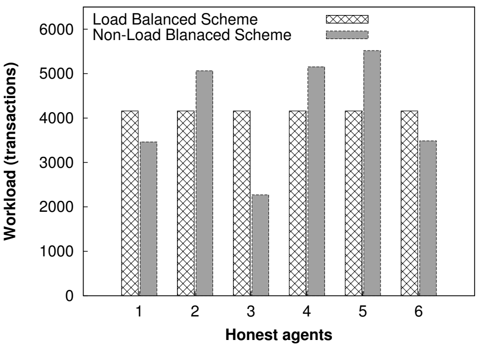

Fig. 10. Comparison between load-balancing and no-load-balancing scheme in terms of workload.

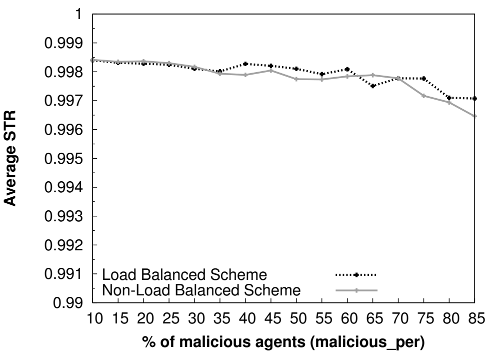

Fig. 11. Comparison between load-balancing and no-load-balancing scheme in terms of average STR.

5.6 Analyzing Scalability

The objective of this section is to show that our trust model remains unaffected in terms of performance as the number of agents in the network increases. For this purpose we computed average STR against the number of agents in the network. We set *malicious_per* to 40%, *malicious_res* to 100% and *collusion* to 0%. Fig. 12 shows that the computed average STR remains same as the number of agents in the network is varied. So, our trust model is scalable with the increase of agents in the system.

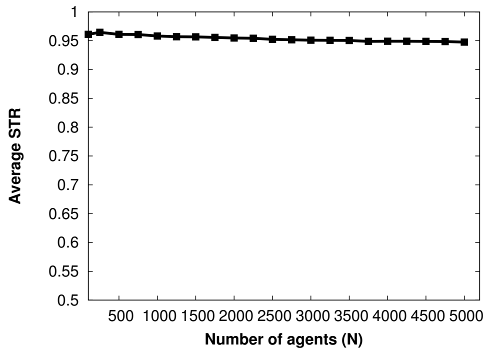

Fig. 12. Scalability of SecuredTrust.

5.7 Decay of trust with lapse of time

This experiment emphasizes the importance of attenuation of trust with elapse of time in absence of interaction. Since now a days agents are highly dynamic in nature, it is realistic that trust should decay with time. For example suppose agent *u* is malicious and it belongs to a community where the most important transactions happen on Friday before the weekends and comparatively less important transactions occur from Monday to Tuesday. Now, agent *u* can intentionally act good from Monday to Tuesday and builds its reputation with the long term plan of utilizing its reputation during the transactions on Friday to cheat other agents. With no decay model present such scenario is possible. To prevent such a case from occurring we incorporate a decay model as described in equation (18).

Fig. 13 shows the different decay schemes possible for our decay model. From the figure it is observable that the attenuation function defined in [27], initially has a higher degradation rate than later one. Realistically it should be the reverse i.e., initially the degradation rate should be smaller and as more and more time elapse without interaction the degradation rate should increase. Our decay function incorporates such philosophy. By including a time decay model we are establishing the principle-“the more recent the transaction the more reliable it is”.

6 Discussion

In this paper we have described a generic trust computation model for multi-agent systems. Our model can be tuned to the meet the requirements of a specific application. For example our model can be used in electronic markets and e-commerce environments (like Amazon Auctions, eBay, OnSale Exchange) where buyers rate sellers regarding their purchase after they make a transaction. Sellers might

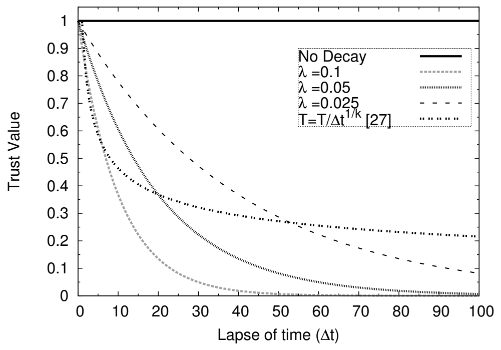

Fig. 13. Decay of trust value with elapse of time.

attempt to raise their trust value (hence reputation) by creating fake buyers and fake transactions. So, in such scenarios our model can be applied to filter out potential bad sellers.

Now we will briefly discuss how our proposed trust model resists some threats, commonly confronted in a multi-agent system. We then propose some potential solutions for restraining some of the other existing threats.

Like any other reputation model our model assists agents to choose reputed agents while avoiding untrustworthy ones. However, reputation-based trust mechanism also introduces vulnerabilities such as shilling attacks where adversaries attack the system by submitting false ratings to confuse the system. Shilling attack is often followed by collusion attack where malicious agents collaborate to raise each other’s rating by making fake transactions. SecuredTrust prevents such threats by assigning feedback credibility to each feedback provider. By doing so, SecuredTrust discards feedbacks submitted by malicious agents and thereby avoid collusion attack. Another challenging threat that most trust models fail to handle is the dynamic personality of malicious agents. By cleverly alternating between good and malicious nature they try to remain undetected while causing damage. SecuredTrust keeps track of sudden rise and fall of trust and thereby can easily penalize such oscillating behavior.

Now, we will propose some techniques which can be incorporated with our trust model to provide solutions to some of the other existing threats present in a multi-agent system. Threats such as tampering of distributed trust information and man in the middle attack can be resolved by combining public key cryptography algorithms on top of our trust model. By doing so we can ensure secured trust data transmission. The free riding problem [49] in case of P2P where an agent only consumes service but does not provide any service or feedback can be handled as suggested by Xiong and Liu [20]. Xiong and Liu have considered a community context factor for providing incentives to agents who give feedbacks. We can also incorporate such a factor in our trust model where the incentive factor can be equal to the ratio of total feedbacks over transactions. Several other remedies have been suggested for the incentive problem of reputation based systems in

[50]. Another common threat is the sybil attack [51] where an agent creates multiple identities and switches from one identity to another. If a malicious agent can easily switch its identity then the trust system may suffer as malicious agents can easily dispatch their bad history. The defense against such attacks should not depend on the trust model but rather on the authentication and access control system. Friedman and Resnick [52] discuss two approaches to this issue: either make it difficult to change online identities or structure the community in such a way that exit and entry with a new identity becomes unprofitable.

7 Conclusion

We have presented a novel trust computation model called SecuredTrust for evaluating agents in multi-agent environments. SecuredTrust can ensure secured communication among agents by effectively detecting strategic behaviors of malicious agents. In this paper we have given a comprehensive mathematical definition of the different factors related to computing trust. We also provide a model for combining all these factors to evaluate trust and, finally we propose a heuristic load balancing algorithm for distributing workload among service providers. Simulation results indicate, compared to other existing trust models SecuredTrust is more robust and effective against attacks from opportunistic malicious agents while being capable of balancing load among service providers.

Acknowledgment

We would like to thank our anonymous reviewers for their thoughtful reviews and advice. We would also like to thank BUET (Bangladesh University of Engineering and Technology) for its cooperation in providing the required computing resources.

References

[1] N. R. Jennings, “An agent-based approach for building complex software systems,” *Communications of the ACM*, vol. 44, no. 4, pp. 35–41, 2001.

[2] R. Steinmetz and K. Wehrle, *Peer-to-Peer Systems and Applications*. Springer-Verlag New York, Inc., 2005.

[3] (2000) Gnutella. [Online]. Available: http://www.gnutella.com

[4] Kazaa. [Online]. Available: http://www.kazaa.com/

[5] (2000) edonkey2000. [Online]. Available: http://www.emule-project.net/

[6] I. Foster, C. Kesselman, and S. Tuecke, “The anatomy of the grid: enabling scalable virtual organizations,” *International Journal of High Performance Computing Applications*, vol. 15, no. 3, pp. 200–222, 2001.

[7] T. Berners-Lee, J. Hendler, and O. Lassila, “The semantic web,” *Scientific American*, pp. 35–43, May 2001.

[8] D. Saha and A. Mukherjee, “Pervasive computing: A paradigm for the 21st century,” *Computer*, vol. 36, no. 3, pp. 25–31, 2003.

[9] S. D. Ramchurn, D. Huynh, and N. R. Jennings, “Trust in multi-agent systems,” *The Knowledge Engineering Review*, vol. 19, no. 1, pp. 1–25, 2004.

[10] P. Dasgupta, “Trust as a commodity,” *Trust: Making and Breaking Cooperative Relations*, pp. 49–72, 2000.

[11] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman, “Reputation systems,” *Communications of the ACM*, vol. 43, no. 12, pp. 45–48, 2000.

[12] A. A. Selcuk, E. Uzun, and M. R. Pariente, “A reputation-based trust management system for P2P networks,” in *Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, 2004, pp. 251–258.

[13] M. Gupta, P. Judge, and M. Ammar, “A reputation system for peer-to-peer networks,” in *Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video (NOSSDAV)*. ACM, 2003, pp. 144–152.

[14] K. Aberer and Z. Despotovic, “Managing trust in a peer-2-peer information system,” in *Proceedings of the tenth international conference on Information and knowledge management (CIKM)*. ACM, 2001, pp. 310–317.

[15] L. Mui, M. Mohtashemi, and A. Halberstadt, “A computational model of trust and reputation for e-businesses,” in *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS’02)*, 2002, pp. 2431 – 2439.

[16] L. Mui, “Computational models of trust and reputation: agents, evolutionary games, and social networks,” Ph.D. Thesis, Massachusetts Institute of Technology(MIT), 2002. [Online]. Available: http://groups.csail.mit.edu/medg/medg/people/lmui/docs/

[17] F. Cornelli, E. Damiani, S. D. Capitani, S. Paraboschi, and P. Samarati, “Choosing reputable servents in a P2P network,” in *Proceedings of the 11th ACM World Wide Web Conference (WWW)*, May 2002, pp. 376–386.

[18] Y. Wang and J. Vassileva, “Bayesian network-based trust model,” in *Proceedings of IEEE/WIC International Conference on Web Intelligence (WI)*, Halifax, Canada, October 2003, pp. 372–378.

[19] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, “The EigenTrust algorithm for reputation management in P2P networks,” in *Proceedings of the 12th ACM international World Wide Web conference (WWW)*, 2003, pp. 640–651.

[20] L. Xiong and L. Li, “Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, pp. 843–857, 2004.

[21] M. Srivatsa, L. Xiong, and L. Liu, “TrustGuard: Countering vulnerabilities in reputation management for decentralized overlay networks,” in *Proceedings of the 14th ACM international conference on World Wide Web (WWW)*, 2005, pp. 422–431.

[22] Z. Runfang and H. Kai, “Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 4, pp. 460–476, 2007.

[23] B. Li, M. Xing, J. Zhu, and T. Che, “A dynamic trust model for the multi-agent systems,” in *Proceedings of IEEE International Symposiums on Information Processing (ISIP)*, 2008, pp. 500–504.

[24] Y. Zhang, S. Chen, and G. Yang, “SFTrust: A double trust metric based trust model in unstructured P2P systems,” in *Proceedings of IEEE International Symposium on Parallel and Distributed Processing (ISPDP)*, 2009, pp. 1–7.

[25] J. Hu, Q. Wu, and B. Zhou, “FCTrust: A robust and efficient feedback credibility-based distributed P2P trust model,” in *Proceedings of IEEE 9th International Conference for Young Computer Scientists (ICYCS)*, 2008, pp. 1963–1968.

[26] X. Wang and L. Wang, “P2P recommendation trust model,” in *Proceedings of IEEE 8th International Conference on Intelligent Systems Design and Applications (ISDA)*, 2008, pp. 591–595.

[27] L. Wen, P. Lingdi, L. Kuijin, and C. Xiaoping, “Trust model of users’ behavior in trustworthy internet,” in *Proceedings of IEEE WASE International Conference on Information Engineering (ICIE)*, 2009, pp. 403–406.

[28] R. Aringhieri, E. Damiani, S. D. Capitani, S. Paraboschi, and P. Samarati, “Fuzzy techniques for trust and reputation management in anonymous peer-to-peer systems: Special topic section on soft approaches to information retrieval and information access on the web,” *Journal of the American Society for Information Science and Technology*, vol. 57, pp. 528–537, 2006.

[29] E. Damiani, S. D. Capitani, S. Paraboschi, and P. Samarati, “Managing and sharing servents’ reputations in p2p systems,” *IEEE Transaction on Knowledge and Data Engineering*, vol. 15, pp. 840–854, 2003.

[30] D. Wen, W. Huaimin, J. Yan, and Z. Peng, “A recommendation-based peer-to-peer trust model,” *Journal of Software*, vol. 15, no. 4, pp. 571–583, 2004.

[31] J. Sabater and C. Sierra, “Regret: A reputation model for gregarious societies,” in *Proceedings of the Fourth Workshop on Deception, Fraud and Trust in Agent Societies*, 2001, pp. 61–69.

[32] Jordi Sabater and Carles Sierra, "Social regret, a reputation model based on social relations," *ACM SIGecom Exchanges - Chains of commitment*, vol. 3, pp. 44–56, December 2001.
[33] J. Sabater and C. Sierra, "Reputation and social network analysis in multi-agent systems," in *Proceedings of the first international joint conference on Autonomous Agents and Multi-Agent Systems*, ser. AAMAS '02. ACM, 2002, pp. 475–482.
[34] L. Xiong and L. Liu, "A reputation-based trust model for peer-to-peer ecommerce communities [extended abstract]," in *Proceedings of the 4th ACM conference on Electronic commerce(EC)*, 2003, pp. 228–229.
[35] T. D. Huynh, N. R. Jennings, and N. R. Shadbolt, "An integrated trust and reputation model for open multi-agent systems," *Autonomous Agents and Multi-Agent Systems*, vol. 13, no. 2, pp. 119–154, 2006.
[36] N. R. Jennings, T. D. Huynh, and N. R. Shadbolt, "FIRE: An integrated trust and reputation model for open multi-agent systems," in *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI)*, 2004, pp. 18–22.
[37] T. D. Huynh, N. R. Shadbolt, and N. R. Jennings, "Developing an integrated trust and reputation model for open multi-agent systems," in *Proceedings of the 7th International Workshop on Trust in Agent Societies*, 2004, pp. 65–74.
[38] J. Sabater, M. Paolucci, and R. Conte, "Repage: REPutation and ImAGE Among Limited Autonomous Partners," *Journal of Artificial Societies and Social Simulation*, vol. 9, no. 2, 2006.
[39] I. Pinyol and J. Sabater-Mir, "Pragmatic-strategic reputation-based decisions in bdi agents," in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, ser. AAMAS '09, 2009, pp. 1001–1008.
[40] R. Andersen, C. Borgs, J. Chayes, U. Feige, A. Flaxman, A. Kalai, V. Mirrokni, and M. Tennenholtz, "Trust-based recommendation systems: An axiomatic approach," in *Proceeding of the 17th ACM international conference on World Wide Web (WWW)*, Beijing, China, 2008, pp. 199–208.
[41] J. J. Qi and Z. Z. Li, "Managing trust for secure active networks," in *Central and Eastern European Conference on Multi-Agent Systems(CEEMAS)*. Springer-Verlag, 2005, pp. 628–631.
[42] Y. Zhang, K. Wang, K. Li, W. Qu, and Y. Xiang, "A time-decay based P2P trust model," in *Proceedings of the 2009 International Conference on Networks Security, Wireless Communications and Trusted Computing*, vol. 2, 2009, pp. 235–238.
[43] Jbuilder. [Online]. Available: http://www.embarcadero.com/products/jbuilder
[44] R. McNab and F. Howell, "Using java for discrete event simulation," in *Proceedings of the Twelfth UK Computer and Telecommunications Performance Engineering Workshop*, 1996, pp. 219–228. [Online]. Available: http://www.dcs.ed.ac.uk/home/hase/simjava/UKPEWpaper.ps
[45] F. Howell. (1999) The simjava home page. [Online]. Available: http://www.dcs.ed.ac.uk/home/hase/simjava/
[46] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon, "Peer-to-peer botnets: overview and case study," in *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*. USENIX Association, 2007, pp. 1–8.
[47] *Expert: Botnets No. 1 Emerging Internet Threat*, CNN Technology.
[48] *Attack of the Zombie Computers Is a Growing Threat*, The New York Times, January 2007.
[49] E. Adar and B. A. Huberman, "Free riding on gnutella," *First Monday*, vol. 5, no. 10, 2000.
[50] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman, "Reputation systems," *Communications of the ACM*, vol. 43, no. 12, pp. 45–48, 2000.
[51] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "Sybilguard: Defending against sybil attacks via social networks," in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications (ACM SIGCOMM)*, 2006, pp. 267–278.
[52] E. J. Friedman and P. Resnick, "The social cost of cheap pseudonyms," *Journal of Economics and Management Strategy*, vol. 10, pp. 173–199, 1998.

Anupam Das received his B. Sc. Engg. and M. Sc. Engg. degree from the Department of Computer Science and Engineering (CSE) of Bangladesh University of Engineering and Technology (BUET) in 2008 and 2010 respectively.

In 2008, he joined the Department of Computer Science and Engineering (CSE) of BUET, first as a lecturer and in 2010 he became an Assistant Professor of the same department. In 2010 he received the prestigious Fulbright International Science and Technology award to pursue higher studies in USA. He is currently pursuing Ph.D. degree in the department of Computer Science at the University of Illinois at Urbana-Champaign (UIUC), USA. His research interests include information security and privacy, network measurement and distributed systems.

M. Mahfuzul Islam (S'03-M'11) received the B.Sc.Engg degree and the M. Sc. Engg. degree from the Department of Computer Science and Engineering (CSE) of Bangladesh University of Engineering and Technology (BUET) in 1997 and 2000, respectively. He obtained the Ph.D. degree from Gippsland School of Information Technology, Monash University, Australia in 2006.

He is an Associate Professor in the same Department of BUET. Dr. Islam has published more than 40 papers in peer-reviewed reputed international journals and conferences. His research interests include network security, vertical handover, wireless resource management and wireless sensor networks.