M.Sc. Engg. Thesis

Feedback Based Dynamic Trust Model for Secured Communication in Multi-agent Systems

by
Anupam Das

Submitted to

Department of Computer Science and Engineering
in partial fulfilment of the requirement for the degree of
Master of Science in Computer Science and Engineering

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka 1000

June, 2010

The thesis titled '**Feedback Based Dynamic Trust Model for Secured Communication in Multi-agent Systems**', submitted by Anupam Das, Roll No. 040805015P, Session April 2008, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents. Examination held on June 05, 2010.

Board of Examiners

1. _____

Dr. M. Mahfuzul Islam — Chairman
Associate Professor — (Supervisor)
Department of Computer Science and Engineering
BUET, Dhaka 1000

2. _____

Dr. Md. Monirul Islam — Member
Professor & Head — (Ex-officio)
Department of Computer Science and Engineering
BUET, Dhaka 1000

3. _____

Dr. Md. Mostofa Akbar — Member
Professor
Department of Computer Science and Engineering
BUET, Dhaka 1000

4. _____

Dr. Mahmuda Naznin — Member
Associate Professor
Department of Computer Science and Engineering
BUET, Dhaka 1000

5. _____

Dr. Hasan Sarwar — Member
Professor & Head — (External)
Department of Computer Science and Engineering
United International University, Dhaka 1209

Candidate's Declaration

This is to certify that the work entitled 'Feedback Based Dynamic Trust Model for Secured Communication in Multi-agent Systems' is the outcome of the research carried out by me under the supervision of Dr. M. Mahfuzul Islam in the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka-1000. It is also declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

Anupam Das
Candidate

Acknowledgments

I would like to take this opportunity in expressing my heartiest and sincere gratitude to all those people who have spent their time and shared their knowledge in assisting me to complete my thesis.

First and foremost, I would like to thank my supervisor, Dr. M. Mahfuzul Islam, Associate Professor, Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET) for his complete supervision, continual inspiration and scholarly suggestions, without which this thesis would have not been possible. His mentorship and tireless scrutiny helped me to finish the work with the best possible results.

I also want to show my gratitude to the other members of my thesis examination board: Dr. Md. Monirul Islam, Dr. Md. Mostofa Akbar, Dr. Mahmuda Naznin and Dr. Hasan Sarwar for their patience and valuable suggestions.

I am grateful to all my friends and colleagues who have helped me from time to time in many ways, without whose encouragement it would have been impossible for me to finish this work in time. I would like to give special thanks to the Information Access Center (IAC), in the Department of Computer Science and Engineering, BUET for providing me with the technical resources that I needed to complete my work.

Lastly, and most importantly, I am forever indebted to my parents and my sister for their patience and mental support when it mattered most.

Abstract

Most network applications such as pervasive computing, grid computing, P2P etc. can be viewed as multi-agent systems which are open, anonymous and dynamic in nature. Due to such nature multi-agent systems present potential threats in ensuring secured communication. One of the key feasible ways to minimize threats is to evaluate the trust and reputation of the interacting agents. Many trust models have done so, but they fail to properly evaluate trust when malicious agents behave in an unpredictable way. Besides that they are also ineffective in providing quick response to a malicious agent's oscillating behavior, i.e., they are unable to assess the true nature of an opportunistic agent. To cope with the strategically altering behavior this thesis presents a dynamic trust model called SECTrust where we analyze the different factors related to the trust of an agent and then propose a comprehensive model for evaluating trust. In our trust model we have considered various factors in computing the trust of agents which includes recent trend, historical trend, sudden deviation of trust, confidence factor and decay of trust. We have also tuned our trust model for the different parameters that we have considered to get the best possible results. Simulations show that our model compared to other existing models can effectively cope with the strategic behavioral change of an agent and at the same time efficiently isolate the malicious ones. So, our trust model is more robust and effective in preventing attacks from opportunistic malicious agents and can thus ensure secured communication among the participating agents.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Security and privacy have become critically important to society with the fast expansion of information communication systems. The scope of associated challenges and applications are broadening accordingly, leading to new requirements and approaches. Challenges arise as information systems evolve into distributed systems that are open in the sense that they do not pre-identify a set of known participants and are dynamic in a way that the participants change regularly, not just due to occasional failures. Such systems include peer-to-peer networks, grid computing environments, ad hoc networks, web services, pervasive computing spaces and multi-agent systems. Now, every security system depends on trust in one form or another among the agents of the system. Trust issues influence not only the specification of security policies but also the techniques needed to manage and implement security policies for systems. This trust is often evaluated through a trust model and as a result researchers have long been devoting their efforts in devising threat resistance robust trust models.

1.1 Motivation

In a multi-agent system, agents interact with each other to achieve a definite goal that they cannot achieve alone [15] and such systems include P2P [29], grid computing [9], the semantic web [5], pervasive computing [26] and so on. 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 it 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 [8, 23]. Researchers have long been utilizing trust theory from social network to construct trust models for effectively suppressing the malicious behavior 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 the agent's behavior from the viewpoint of trust.

Now most of the existing global trust models can successfully isolate malicious agents when the agents behave in a predictable way. These models, however, suffer greatly when agents start to show dynamic personality, i.e., when they start to behave in an unpredictable way. These models also fail to adapt to quick change in agent's behavior and as a result suffer when agents alter their activities strategically. Moreover, some of the models show little effect in dealing with more complex attacks such as dishonest or unfair rating and collusion.

With this research problem in mind, we propose a dynamic trust model named SECTrust which can effectively detect sudden strategic alteration in malicious behavior. SECTrust considers variety of factors in determining the trust of an agent such as recent and historical trend, decay of trust and deviation in trust. We have also used a novel policy of utilizing exponential averaging function to reduce storage overhead in computing the trust of agents.

1.2 Preliminaries

1.2.1 Agents

Philosophically an agent means an entity which is capable of doing action. However, the term 'agent' is difficult to define as agents are described as entities with attributes that are considered useful in a particular domain. Different researchers have given different definitions about 'agents' and some of these definitions are given below-

- "Most often, when people use the term 'agent' they refer to an entity that functions continuously and autonomously in an environment in which other processes take place and other agents exist."-(Shoham, 1993).
- "An agent is an entity that senses its environment and acts upon it"-(Russell, 1997).
- "The term agent is used to represent two orthogonal entities. The first is the agent's ability for autonomous execution. The second is the agent's ability to perform domain oriented

reasoning."-(the MuBot Agent).

- "Intelligent agents are software entities that carry out some set of operations on behalf of a user or another program, with some degree of independence or autonomy, and in so doing, employ some knowledge or representation of the user's goals or desires."-(the IBM Agent).

- "An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, in pursuit of its own agenda and so as to effect what it senses in the future."-(Franklin, Gasser, 1997).

- "An agent is a hardware or (more usually) a software-based computer system that enjoys the following properties: autonomy - agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state; social ability - agents interact with other agents (and possibly humans) via some kind of agent-communication language; reactivity: agents perceive their environment and respond in a timely fashion to changes that occur in it; pro-activeness: agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by taking initiative."- Wooldridge and Jennings (1995)

So, in brief we can say that an agent refers to an intelligent, autonomous entity which observes and acts upon an environment and directs its activity towards achieving a defined goal. There are different types of agents like-Mobile Agents (e.g. Voyager, Grasshopper), Learning Agents, Autonomous Agents (e.g. robots), Planning Agents, Simulation agents, Distributed Agents and Software agents. However, in our research work we are referring to only software agents i.e., agents that constitute of computer programs or computer applications.

1.2.2 Multi-Agent Systems

A multi-agent system (MAS) is a loosely coupled network of software agents that interact to solve problems that are beyond the individual capacities or knowledge of each agent [19]. In other words, multi-agent systems are computational systems in which several agents interact or work together to perform some set of tasks or satisfy some set of goals.

The main characteristics of MASs are [30]-

◇ Each agent has incomplete information or capabilities for solving the problem and, thus, has a limited viewpoint.

◇ There is no global control system.

◇ Data are decentralized.

◇ Computation is asynchronous.

Typically multi-agent systems research refers to software agents. However, the agents in a multi-agent system could equally well be robots, humans or human teams. A multi-agent system may even contain combined human-agent teams. Multi-agent systems can manifest self-organization and complex behaviors even when the individual strategies of all their agents are simple.

The advantages [3] of using a multi-agent approach over a single agent or centralized approach are:

- An MAS distributes computational resources and capabilities across a network of interconnected agents, whereas a centralized system may be plagued by resource limitations, performance bottlenecks, or critical failures. An MAS is decentralized and thus does not suffer from the ”single point of failure” problem associated with centralized systems.
- An MAS allows for the interconnection and inter-operation of multiple existing legacy systems. By building an agent wrapper around such systems, they can be incorporated into an agent society.
- An MAS models problems in terms of autonomous interacting component-agents, which is proving to be a more natural way of representing task allocation, team planning, user preferences, open environments, and so on.
- An MAS efficiently retrieves, filters, and globally coordinates information from sources that are spatially distributed.
- An MAS enhances overall system performance, specifically along the dimensions of computational efficiency, reliability, extensibility, robustness, maintainability, responsiveness, flexibility, and reuse.

Multi-agent systems include a wide range of computer communication networks such as-

1. Peer-to-Peer Network
2. Grid computing environments

3. MANETs

4. Internet Web services

5. Pervasive computing spaces

Applications of multi-agent research cover a variety of domains, including aircraft maintenance, electronic book buying coalitions, military demining, wireless collaboration and communications, military logistics planning, supply-chain management, joint mission planning, financial portfolio management etc.

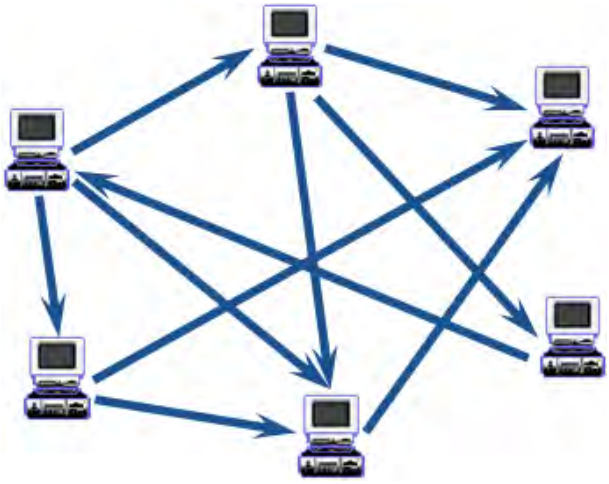


Figure 1.1: A simple illustration of multi-agent system.

1.2.3 Threats in Multi-Agent Systems

A multi-agent system is faced with numerous threats due to its open and dynamic nature. These threats can severely affect the performance of multi-agent system. The most common types of threats [6] encountered in an MAS are:

- **Whitewashing attack**: The attacker resets a poor reputation by rejoining the system with a new identity. Systems that allow for easy change of identity and easy use of new pseudonyms are vulnerable to this attacks.

- **Sybil attack**: The attacker creates multiple identities (sybils) and exploits them in order to manipulate a reputation score.

- **Impersonation and reputation theft**: One entity acquires the identity of another entity (masquerades) and consequently steals its reputation.
- **Denial-of-reputation**: Attack designed to damage an entity's reputation (e.g. in combination with a sybil attack or impersonation) and create an opportunity for blackmail in order to have the reputation cleaned.
- **Unfair or dishonest rating**: Reporting of false reputation score to give a false impression about a given agent.
- **Collusion**: Multiple users conspire (collude) to influence a given reputation.
- **Repudiation of data and repudiation of transaction**: An entity can deny that a transaction happened, or the existence of data for which he was responsible.
- **Recommender dishonesty**: The voter is not trustworthy in his scoring.
- **Dynamic personality**: Entities show oscillating personality like-milking reputation after building it. Entities alternate between building and milking reputation.

In this thesis work we are concentrating on developing techniques to address threats like-unfair/dishonest rating, collusion, recommender dishonesty and dynamic personality.

1.2.4 Trust and Reputation

Security of any system depends on trust issues and trust policies [6,35] i.e., the security of any system depends on trust in one form or another among the agents of the systems.

Trust is defined as firm belief in the competence of an agent to act dependably, securely and reliably within a specified context. It is the belief that an entity has about another entity from its past experience and knowledge about the entity's nature. The belief expresses an expectation of the entity's behavior in future and hence it implies uncertainty and risk. Moreover, trust is a personal and subjective phenomenon that is based on various factors or evidence i.e., trust is a cognitive process and as a result it is fuzzy and imprecise in nature.

Now, the concept of reputation is closely linked to that of trustworthiness but there is a subtle difference [17, 32]. Reputation is what is generally said or believed about a person's or thing's character or status. So, reputation can be considered as a collective measure of trustworthiness (in the

sense of reliability) based on the referrals or ratings from members in a community. Reputation based systems [24] help participants to evaluate trustworthiness of each other and predict future behaviors of participants based on the community feedback about the participants' past behavior. By harnessing the community knowledge in the form of feedback, these systems help agents decide who to trust, encourage trustworthy behavior, and deter dishonest participation. Reputation based systems minimize risks by collecting and aggregating feedback about participants' past behavior. These systems provide a way for participants to share their experiences and knowledge so that they can estimate the trustworthiness of other participants with whom they may not have any personal experiences and thus can avoid participation from malicious agents to reduce risk.

So, in order to get a better view of the real nature of an agent both trust and reputation should be considered. In other words, both trust and reputation play a critical role in determining the degree of confidence an agent has upon other agents.

1.2.5 Trust Model

Trust models represent trust quantitatively through various trust metrics. A trust metric is a measurement of the degree to which members of a group is trusted by each other. So, trust models provide us with a means to determine the approximate trust value of an agent in the system. Trust models consider a numbers factors such as direct/local trust, indirect/referral trust, credibility and similarity in determining the value of trust.

Now, the network at present is highly dynamic and the agents in the network themselves show dynamic personality. As a result the trust model has to consider all these dynamism and adapt to the new requirements and challenges that arise. In other words, the agents in the system must have the abilities to monitor and respond to the current situation and act accordingly. Trust models that adapt to these dynamism are known as dynamic trust models.

1.3 Scope of Work

The main objective of this thesis work is to provide a dynamic trust model for effectively evaluating the trust of agents even in the presence of highly oscillating malicious agents. The work comprises the following studies-

- A thorough analysis of the different factors related to computing the trust of an agent.

- Detailed investigation about different types of malicious behavior conducted by agents.
- Designing a comprehensive trust model for effectively isolating malicious agents.
- Comparison study with the existing trust model.

1.4 Thesis Organization

The rest of the chapters are organized as follows-

Chapter 2 represents a review of the most recent works done on trust model. A detailed description along with the limitations of the trust models have also been discussed in this chapter.

Chapter 3, the main chapter of this thesis, illustrates our proposed trust model and the different factors considered in developing the trust model. A through analysis and description of the mathematical expressions used for the different factors have also been presented in this chapter.

Chapter 4 contains the simulation results of our proposed model and a comparative study against other existing trust models under different scenarios. The details of the simulation setup and the behavioral pattern of agents are also discussed in this chapter.

Chapter 5 concludes by describing the key contributions of this thesis followed by some suggestion for future research work related to this topic.

Chapter 2

Related Works

Dynamic trust model is relatively a new approach in the domain of network security research. The main objective of any trust model is to provide a threat resistance policy for selecting agents in case of providing services. The main challenges arise as agents in the system start to behave unpredictably i.e., when malicious agents alternate the activities strategically with the intent to rip benefits for themselves. Most complex malicious activities include providing false rating, forming collusive groups and strategically altering nature (e.g., becoming malicious after attaining high reputation) with the view to not getting noticed by others. In this chapter we will discuss some of the most recent and well known works done on trust model and review their approaches to solving the threats that exist in a multi-agent system.

2.1 Introduction

A reputation based trust model [24] 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 mainly divided into two categories: local trust models and global trust models. Local trust models [1, 7, 21] aggregate feedback from some predetermined neighbors through local broadcasting. In case of global trust models [18, 20, 25, 28, 35] an agent aggregates feedback from all the agents who ever interacted with the target agent i.e., in global trust models an agent has a view of the network which is wider than its own experience. So local trust models only have a partial view of the network and as a result cannot compute the trust of an agent accurately. On the other hand global trust models utilize the experience

gained by others in computing the trust of an agent and therefore can properly reflect the actual nature of an agent. So, we will discuss the key ideas of the following global trust models: Bayesian network based trust model [32], EigenTrust [18], FIRE [13], PeerTrust [35, 36], FCTrust [12], SFTrust [39], recommendation trust model provided by Wang et al. [31], trust model provided by Li et al. [20] and trust model proposed by Wen et al. [34].

2.2 Bayesian Network-Based Trust Model

The Bayesian network based trust model [32] believes that trust is multi-dimensional and agents need to evaluate trust from different aspects of an agent's capability. The agents needs are different in different situations and depending on the situation, an agent may need to consider its trust in a specific aspect of another agent's capability or in a combination of multiple aspects. Bayesian networks provide a flexible way to present differentiated trust and combine different aspects of trust. A Bayesian network is a relationship network that uses statistical methods to represent probability relationships between different agents.

In the Bayesian network-based trust model every agent develops a naive Bayesian network for each service provider that it has ever interacted with. Each Bayesian network has a root node *T*, which has two values, "satisfying" and "unsatisfying", denoted by 1 and 0, respectively. $p(T = 1)$ represents the value of agents overall trust in the service provider's competence in providing good services. It is the percentage of interactions that are satisfying and measured by the number of satisfying interactions m divided by the total number of interactions n. $p(T = 0)$ is the percentage of not satisfying interactions.

$$p(T = 1) = \frac{m}{n} \tag{2.1}$$

$$p(T = 1) + p(T = 0) = 1 \tag{2.2}$$

The leaf nodes under the root node represent the service providers capability in different aspects. Each leaf node is associated with a conditional probability table (CPT). This conditional probability is calculated by using the Bayes rule [11].

$$p(h|e) = \frac{p(e|h) * p(h)}{p(e)} \tag{2.3}$$

where $p(h)$ is the prior probability of hypothesis h, $p(e)$ is the prior probability of evidence e, $p(h|e)$ is the probability of h given e and $p(e|h)$ is the probability of e given h. Agents calculate conditional

probability for all the aspects of the service provider that it thinks is relevant. So, with the Bayesian networks, agents can infer trust in the various aspects that they need from the corresponding probabilities. After each interaction, agents update their corresponding Bayesian networks. If an interaction is satisfying, m and n are both increased by 1 in formula 2.1. If it is not satisfying, only n is increased by 1.

The main problem with this approach is that different agents have different requirements and as result the architecture of the Bayesian network may be different for different agents. Moreover, the probabilities are calculated based on the agent's own subjective judgement and this may vary from agent to agent i.e., the measures are not unique rather it is biased by the agent calculating it.

In this trust model agents also aggregate recommendation from other agents. When other agents receive recommendation requests, they will check their trust-representations, i.e. their Bayesian networks, to see if they can provide answers such requests. The requesting agent might receive several such recommendations at the same time, which may come from the trustworthy acquaintances, untrustworthy acquaintances, or strangers. The agent first discards recommendations from untrustworthy agents and then combines the recommendations from trustworthy references and unknown references to get the total recommendation for the service provider:

$$r_{ij} = w_t * \frac{\sum_{l=1}^{k} tr_{il} * t_{lj}}{\sum_{l=1}^{k} tr_{il}} + w_s * \frac{\sum_{z=1}^{g} t_{zj}}{g}, \text{where } w_t + w_s = 1 \tag{2.4}$$

r_{ij} is the total recommendation value for the j^{th} service provider that the i^{th} agent gets. k and g are the number of trustworthy references and the number of unknown references, respectively. tr_{il} is the trust that the i^{th} user has in the l^{th} trustworthy reference. t_{lj} is the trust that the l^{th} trustworthy reference has in j^{th} service provider. t_{zj} is the trust that the z^{th} unknown reference has in j^{th} service provider. w_t and w_s are the weights to indicate to what extent the user values the importance of the recommendation from trustworthy references and from unknown references. The agent should weight the recommendations from its trustworthy acquaintances higher than those recommendations from strangers as trustworthy acquaintances share similar preferences and viewpoints.

However, this model assumes that all the agents are truthful in providing their evaluations. This is not realistic as malicious agents will not provide true feedback to others rather it will provide high ratings for other malicious agents and low ratings for good agents.

After each interaction an agent updates the trust it has upon the recommenders according to the

following reinforcement learning formula-

$$tr_{ij}^{n} = \alpha * tr_{ij}^{o} + (1 - \alpha) * e_{\alpha} \tag{2.5}$$

where tr_{ij}^{n} denotes the new trust value that the i^{th} agent has in the j^{th} reference after the update and tr_{ij}^{o} denotes the old trust value. α is the learning rate which is a real number in the interval $[0, 1]$. e_{α} is the new evidence value and it is defined as shown below-

$$e_{\alpha} = \begin{cases} 1, \text{if recommendation value is greater than a given threshold } \theta \text{ and} \\ \quad \text{the interaction with the service provider afterwards is satisfying} \\ -1, \text{if recommendation value is greater than a given threshold } \theta \text{ and} \\ \quad \text{the interaction with the service provider afterwards is unsatisfying} \end{cases} \tag{2.6}$$

This trust model also provides a way for an agent to determine the trustworthiness of another agent. This is done by a direct comparison between the two agents Bayesian networks relevant to an identical service provider. However, in comparing the Bayesian networks the authors have assumed that all the agents have identical network structure but as discussed earlier agents tend to have different requirements and as a result the Bayesian network for one agent may differ from that of another agents.

Another aspect that this trust model forgot to handle is the decay of trust with elapse of time.

2.3 EigenTrust

EigenTrust [18] aggregates the local trust values of all peers/agents to calculate the unique global trust value of a given peer. The global reputation of each peer *i* is calculated by the local trust values assigned to peer *i* by other peers, weighted against the global reputations of the assigning peers.

Local trust value is calculated as the sum of the ratings of the individual transactions that peers perform. Local trust value is defined by the following equation-

$$s_{ij} = \sum_{all\_transactions} tr_{ij} \tag{2.7}$$

where s_{ij} denotes the local trust value that peer *i* has about peer *j* and tr_{ij} represents the transaction rating (which lies in the range $[-1, 1]$) the peer *i* assigns for the service provided by peer *j*. Equivalently, each peer *i* can store the number satisfactory transactions it has had with peer *j*, $sat(i, j)$ and

the number of unsatisfactory transactions it has had with peer *j*, $unsat(i, j)$. Then, s_{ij} can be define as

$$s_{ij} = sat(i, j) - unsat(i, j) \tag{2.8}$$

In order to aggregate local trust values, EigenTrust normalizes the local trust values. Local trust values are normalized according to the following function-

$$c_{ij} = \begin{cases} \frac{max(s_{ij},0)}{\sum_j max(s_{ij},0)}, & \text{if } \sum_j max(s_{ij}, 0) \neq 0 \\ p_j, & \text{else} \end{cases} \tag{2.9}$$

where c_{ij} represents the normalized trust value and p_j represents the probability of choosing a pre-trusted peer (peers that are fully trusted by all peers in the system) . In other words, if peer *i* does not know anybody or does not trust anybody (i.e., $\sum_j max(s_{ij}, 0) = 0$) then it will choose to trust the pre-trusted peers. For this purpose in Eigentrust the exists a set of pre-trusted peers (P) where the probability of choosing any of these pre-trusted peers is $p_j = 1/|P|$.

However, the manner in which local trust is normalized in equation 2.9 has some drawbacks. For one, the normalized trust values do not distinguish between a peer with whom peer *i* did not interact and a peer with whom peer *i* has had poor experience. Also, these c_{ij} values are relative, and there is no absolute interpretation i.e., if $c_{ij} = c_{ik}$, we know that peer *j* has the same reputation as peer *k* in the eyes of peer *i*, but we do not know if both of them are very reputable, or if both of them are mediocre. Another flaw of EigenTrust is its dependency on some pre-trusted agents for trust calculation because the selection of these pre-trusted agents is difficult in real life.

After normalizing local trust value, EigenTrust aggregates the normalized local trust values. In a distributed environment, one common way to do this is for a peer to ask its acquaintances (recommenders) about their opinions regarding some target peer. In taking opinions from others, their opinions are weighted according to the trust that the inquiring peer assigns to them. So, the global trust is calculated according to the following function-

$$t_{ik} = \sum_j c_{ij} c_{jk} \tag{2.10}$$

where t_{ik} represents the trust that peer *i* places in peer *k* based on asking his friends. From equation 2.10, we see that credibility of any recommender is a function of the normalized trust value of that peer i.e., feedback from trustworthy peers were considered more credible and, thus, weighted more than those from untrustworthy peers. So, here it has been assumed untrustworthy peers are more likely to submit false or misleading feedback in order to hide their malicious intent while trustworthy

peers are more likely to be honest on the feedback they provide. However, a honest peer might not necessarily provide honest feedback to its competitors.

The trust model proposed by Dou et al. [33] 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 doesn't consider the punishment of dishonest recommenders. Moreover neither [18, 33] address as critical aspect of trust which the decay of trust value in absence of interaction.

2.4 FIRE: An Integrated Trust and Reputation Model for Open Multi-Agent Systems

The FIRE model [13, 14, 16] believes that most of the trust information source can be categorized into the four main sources: direct experience, witness information, role based rules and third party references.

- Direct experience: The evaluator uses its previous experiences in interacting with the target agent to determine its trustworthiness. This type of trust is called *interaction trust* (IT).

- Witness information: Assuming that agents are willing to share their direct experiences, the evaluator can collect experiences of other agents that interacted with the target agent. Such information will be used to derive the trustworthiness of the target agent based on the views of its witnesses. Hence this type of trust is called *witness reputation* (WR).

- Role-based rules: Besides an agents past behaviors (which is used in the two previous types of trust), there are certain types of information that can be used to deduce trust. These can be the various relationships between the evaluator and the target agent. For example, an agent may be preset to trust any other agent that is owned, or certified, by its owner; or it may trust another agent if it is a member of a trustworthy group. Such settings or beliefs (which are mostly domain-specific) can be captured by rules based on the roles of the evaluator and the target agent to assign a predetermined trustworthiness to the target agent. Hence this type of trust is called *role-based trust* (RT).

- Third-party references provided by the target agents: In the previous cases, the evaluator needs to collect the required information itself. However, the target agent can also actively seek

the trust of the evaluator by presenting arguments about its trustworthiness. In this approach, such arguments are references produced by the agents that have interacted with the target agents certifying its behaviors. However, in contrast to witness information which needs to be collected by the evaluator, the target agent stores and provides such certified references on request to gain the trust of the evaluator. Those references can be obtained by the target agent (assuming the cooperation of its partners) from only a few interactions, thus, they are usually readily available. This type of trust is called *certified reputation* (CR).

FIRE integrates all four sources of information and is able to provide trust metrics in a wide variety of situations. The reliability value bases on the rating reliability and deviation reliability to counteract the uncertainty due to instability of agents.

However, WR and CR components depend on third-party information (witness experiences and references) and, therefore, they are susceptible to dishonesty and unfair rating. Since agents in an open MAS are self-interested, they may provide false ratings to gain unwarranted trust for their partners. In FIRE authors have not considered the problem of lying and dishonesty. They have assumed that all agents are honest in exchanging information which is completely unrealistic. Moreover, for trust computation each agent stores a maximum number of the latest ratings given to another agent and this causes storage overhead. Last of all this model did not consider the attenuation of trust with the elapse of time.

2.5 PeerTrust

In PeerTrust [35,36], a peer's/agent's trustworthiness is defined by the evaluation that the peer receives in providing service to other peers in the past. Such reputations reflect the degree of trust that other peers in the community have on the given peer based on their past experiences. In PeerTrust Xiong et al. defined five factors in computing the trustworthiness of peers. The factors are-

1. the feedback a peer obtains from other peers.
2. the feedback scope, such as the total number of transactions that a peer has with other peers.
3. the credibility factor for the feedback source.
4. the transaction context factor for discriminating mission-critical transactions from less or non-critical ones.

5. the community context factor for addressing community-related characteristics and vulnerabilities.

As we can see the first three of these factors are basic trust parameters while the remaining two are adaptive factors.

Then Xiong et al. combined these parameters to present a general trust metric as shown below-

$$T(u) = \alpha * \sum_{i=1}^{I(u)} S(u,i) * Cr(p(u,i)) * TF(u,i) + \beta * CF(u) \tag{2.11}$$

where $I(u)$ denotes the total number of transactions performed by peer *u* with all other peers, $p(u,i)$ denotes the other participating peer in peer *u*'s *i*th transaction, $S(u,i)$ denotes the normalized amount of satisfaction peer *u* receives from $p(u,i)$ in its *i*th transaction, $Cr(v)$ denote the credibility of the feedback submitted by *v*, $TF(u,i)$ denote the adaptive transaction context factor for peer *u*'s *i*th transaction, and $CF(u)$ denote the adaptive community context factor for peer *u*. Here, α and β denote the normalized weight factors for the collective evaluation and the community context factor.

As we can see from equation 2.11 that the metric consists of two parts. The first part is a weighted average of amount of satisfaction a peer receives for each transaction. The weight takes into account the credibility of feedback source to counter dishonest feedback, and transaction context to capture the transaction-dependent characteristics. The second part of the metric adjusts the first part by an increase or decrease of the trust value based on community-specific characteristics and situations. The α and β parameters can be used to assign different weights to the feedback-based evaluation and community context according to different situations.

Now, considering only the basic trust parameters: the feedback, the number of transactions, and the credibility of feedback source i.e., if we turn off the transaction context factor ($TF(u,i) = 1$) and the community context factor ($\alpha = 1$ and $\beta = 0$) then equation 2.11 reduces to

$$T(u) = \sum_{i=1}^{I(u)} S(u,i) * Cr(p(u,i)) \tag{2.12}$$

In PeerTrust Xiong et al. defined two credibility measures. In the first one they used a function of the trust value of a peer as its credibility factor i.e., feedback from trustworthy peers were considered more credible and, thus, weighted more than those from untrustworthy peers. Credibility function used is defined below-

$$Cr_u(v) = \frac{T(v)}{\sum_{j=1}^{I(u)} T(p(u,j))} \tag{2.13}$$

where $Cr_u(v)$ denotes the credibility measure of peer *v* from peer *u*'s perspective. However, this solution is based on two assumptions. Firstly, untrustworthy peers are more likely to submit false or misleading feedback in order to hide their own malicious behavior. Secondly, trustworthy peers 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 at all time. For example, it is possible (though not common) that a peer may maintain a good reputation by performing high quality services, but send malicious feedback to its competitors. In this extreme case, using a function of trust to approximate the credibility of feedback will generate errors.

The second measure that Xiong et al. defined uses a personalized similarity measure to rate the credibility of another peer. This definition uses a similarity measure which is calculated according to the following function-

$$Sim(u,v) = 1 - \sqrt{\frac{\sum_{x \in IJS(u,v)} \left(\frac{\sum_{i=1}^{I(x,u)} S(x,i)}{I(x,u)} - \frac{\sum_{i=1}^{I(x,v)} S(x,i)}{I(x,v)} \right)^2}{|IJS(u,v)|}} \tag{2.14}$$

where $IJS(u,v)$ denotes the common set of peers that have interacted with both peer *u* and peer *v*, and $I(u,v)$ denotes the total number of transactions performed by peer *u* with *v*. The second form of credibility measure defined by Xiong et al. is shown below-

$$Cr_u(v,w) = \frac{Sim(v,w)}{\sum_{j=1}^{I(u)} Sim(p(u,j),w)} \tag{2.15}$$

Here, $Cr_u(v,w)$ denotes the credibility measure of peer *v* from peer *w*'s perspective where peer *w* is computing the trust value of peer *u* and peer *v* is one of the recommenders providing recommendation to peer *w*. Using personalized credibility measure provides great deal of flexibility and stronger predictive value as the feedback from similar raters are given higher weight than those that are less similar. It also act as an effective defense against potential malicious collusion.

The main limitations of PeerTrust is that it has to retrieve all the transactions within the recent time window ($I(u)$, which may contain a large number of transactions) to compute the trust of an agent. So the process is expensive from both computational and spatial point of view. Moreover, all the transactions in the window is given equal significance but recent transactions should be given higher weight than past transactions.

PeerTrust also does not consider sudden fluctuation of trust, but malicious peers tend to strategically fluctuate their between building and milking reputation. Another aspect that PeerTrust does not address is the decay of trust with time in absence of interaction.

2.6 P2P Recommendation Trust Model

In computing the global trust of any agent, the recommendation model provided by Wang et al. [31] combines both local trust calculated from the agent's own experience and recommendation provided by other agents in the system.

This model calculates local trust by taking the ratio of successful transaction over the total number of transactions as shown below-

$$P_{ij} = \begin{cases} \frac{Suc_{ij}}{Total_{ij}}, & \text{if } Total_{ij} > 0 \\ 0, & \text{else} \end{cases} \tag{2.16}$$

where P_{ij} represents the local trust calculated by agent *i* regarding agent *j*, Suc_{ij} denotes all the successful transactions between agent *i* and *j* while $Total_{ij}$ represents the total number of transactions between *i* and *j*. However, this formulation of local trust fails to reflect the trend and relative time significance of the transactions.

Recommendation is obtained by considering both recommendation value provided by recommenders and their credibility. It is calculated according to the following equation-

$$R_{ij} = \frac{\sum_{k=1}^{|W|} C_{ik} * (T_{kj} + \lambda)}{|W|} \tag{2.17}$$

$$T_{ij} = \begin{cases} \frac{2*Suc_{ij} - Total_{ij}}{\sum_{k=1}^{|W|} Suc_{kj}}, & \text{if } 2 * Suc_{ij} - Total_{ij} > 0 \\ 0, & \text{else} \end{cases} \tag{2.18}$$

where R_{ij} denotes the recommendation value received by agent *i* about agent *j*, C_{ij} represents the credibility of *j* from *i*'s respective and T_{ij} represents the recommendation value (shown in eqn 2.18) that *i* has about *j*. W corresponds to the recommenders set and λ is a user defined constant which signifies the default recommendation value in the initial stage of the network when recommenders have not interacted with the respective target agent. The value of λ becomes zero as the network enters into a stable state i.e., when a significant number of transactions have occurred. However, no specification is given as to who are included in the recommendation membership set.

The credibility (reputation) of any agent is updated according to a reward and punishment policy. Credibility is updated using the following function-

$$C_{ij} = \begin{cases} C_{ij} + 0.1, & \text{if own evaluation and feedback are nearly the same} \\ \frac{C_{ij}}{2}, & \text{else} \end{cases} \tag{2.19}$$

However, as we can see from the above expression there is no specification of the threshold of difference between own evaluation and feedback provided by recommenders.

Global trust is calculated from the combination of local trust and recommendation according to the following equation-

$$G_{ij} = \alpha * P_{ij} + (1 - \alpha) * R_{ij} \tag{2.20}$$

where G_{ij} represents the value of global trust and α represents the contribution of local trust. However, the value of α is fixed and does not change dynamically. So the model cannot accommodate the experience gained over time by the agent or recommenders.

Another limitation of this model is that it has not considered any time decay model i.e., trust value does not change over time in absent of transaction. Nonetheless, trust values should attenuate with time to signify the dynamic nature of agents behavior.

2.7 FCTRust: Feedback Credibility Based Trust Model

FCTrust [12] uses transaction density and similarity measure to define the credibility of any recommender as opposed to [18, 33] which use the global trust as the weight of the quality of feedbacks. In other words, FCTrust has pointed out the fact that the role of any agent in providing feedbacks in different from that of providing services to others.

FCTrust defines local trust as the normalized value of local satisfactory transactions. Local trust value is calculated according to the following function-

$$D_{ij} = \begin{cases} \frac{\sum_{k=1}^{m} f(i,j)}{m}, & \text{if } m \neq 0 \\ 0, & \text{else} \end{cases} \tag{2.21}$$

where D_{ij} denoted the direct trust calculated by agent *i* about agent *j*, $f(i, j)$ represents satisfaction degree evaluation function between agent *i* and *j*, and it is defined as follows-

$$f(i,j) = \begin{cases} 1, & \text{if transaction totally satisfactory} \\ 0, & \text{if transaction totally unsatisfactory} \\ e \in (0,1), & \text{else} \end{cases} \tag{2.22}$$

The main problem with the formulation of direct/local trust is that FCTrust stores all the transaction performed within a time frame which might be a significantly long period and this results in storage overhead. Moreover, the simple averaging function defined in equation 2.21 fails to reflect the trend and relative time significance of the transactions.

FCTrust defines credibility through two factors: transaction density and similarity. Transaction density signifies the transaction frequency where higher frequency means higher credibility. Transaction density is defined as follows-

$$TNum_{ij} = \frac{m}{n} * \beta^{1/m}, \beta \in (0,1) \bigcap m \neq 0 \tag{2.23}$$

where m denotes the transaction number between agent *i* and *j*, and n denotes the total transaction number of agent *i* with others. If $m = 0$ then $TNum_{ij} = 0$. However, according to equation 2.23 if $n >> m$ then the value of $TNum_{ij}$ will be very small even if agent *j* makes a significant number (m) of cooperative transactions and since $TNum_{ij}$ is directly used in the calculation of credibility, it will result in a low value of credibility for *j* irrespective of it's cooperative behavior.

FCTrust defined similarity according to the following equation-

$$TSim_{ij} = \begin{cases} TSim_{ij} + \frac{1-TSim_{ij}}{2} * (1 - \frac{TDif_{ij}}{\theta}), & \text{if } TDif_{ij} < \theta \\ TSim_{ij} - \frac{TSim_{ij}}{2} * (1 - \frac{\theta}{TDif_{ij}}), & \text{else} \end{cases} \tag{2.24}$$

where $TDif_{ij}$ represents the difference in rating between agent *i* and *j* over the common interaction set of agents. θ denotes the maximum deviation that agent *i* allows for agent *j*. From equation 2.24 we can see that FCTrust assigns equal degree of reward and punishment in computing similarity but the degree of punishment should be greater than that of incentive.

Credibility is then calculated from transaction density and similarity as follows

$$Cr_{ij} = TNum_{ij} * TSim_{ij}; \tag{2.25}$$

Again, FCTrust does not address the issue of decay of trust with the elapse of time in absence of interaction.

2.8 SFTrust: A Double Metric Based Trust Model

SFTrust [39] is a double trust metric model. It separates service trust from feedback trust with the view to take full advantage of all the agents' service abilities even in the presence of fluctuating feedbacks. In SFTrust each agent maintains two lists namely transaction list and neighbor list. For each neighbor it maintains a service record and a credibility record.

SFTrust defined direct/local trust by an iterative update function which stores only one value in the system and as a result reduce storage overhead. The iterative update function for direct trust (*DT*)

is defined as follows-

$$DT_{ij}^{n+1} = \frac{DT_{ij}^{n}(1-\lambda^{n}) + \lambda^{n+1}(1-\lambda)S_{ij}^{n+1}}{(1-\lambda^{n+1})} \tag{2.26}$$

where DT_{ij}^{n} represents the direct trust of the past n transactions, S_{ij}^{k} represents the satisfaction degree of the $k\ th$ service/transaction and λ represents the fading factor ($0 < \lambda < 1$) that can be changed dynamically depending on the system.

SFTrust also considers taking recommendation from neighbors. Recommending trust (*RT*) is defined as follows-

$$RT_{ij} = \frac{\sum_{k=1}^{m} FT_{ik} * f_{kj}}{\sum_{k=1}^{m} FT_{ik}} \tag{2.27}$$

where f_{ij} denotes the feedback that agent *i* provides about agent *j* according to the service trust of *j*. Now, in SFTrust an agent blindly forwards request to all it neighbors ($k = 1, 2, ..., m$) for recommendation. The agents receiving the request can then forward it to their neighbors and so on. TTL field is used to limit the broadcast. However, aggregating information through this kind of local broadcasting can be time consuming. Moreover, recommendation should come from agents that have first hand experience [4, 22, 27] with the respective target agent.

Service trust (*ST*) is calculated by combining direct trust and recommending trust as shown below-

$$ST_{ij} = \delta * DT_{ij} + (1-\delta)RT_{ij} \tag{2.28}$$

where δ ($0 \leq \delta \leq 1$) represents the contribution of direct trust in computing service trust. However, the value of δ is fixed and does not change dynamically. So the model cannot accommodate the experience gained over time by the agent or recommenders.

SFTrust uses similarity measure (*Sim*) in defining feedback trust (credibility measure). SFTrust uses a cosine similarity measure as defined below-

$$Sim(i,j) = \frac{\sum_{k \in CS(i,j)} f_{ik} * f_{jk}}{\sqrt{\sum_{k \in CS(i,j)} f_{ik}^{2}} * \sqrt{\sum_{k \in CS(i,j)} f_{jk}^{2}}} \tag{2.29}$$

where $CS(i,j)$ denotes the common set of agents that have interacted with both agent *i* and agent *j*. SFTrust updates feedback trust (*FT*) by comparing the computed similarity measure (eqn 2.29) with the similarity threshold (Sim_{θ}) according to the following function-

$$FT_{ij} = \begin{cases} FT_{ij} + \eta * (1 - FT_{ij}), & \text{if } Sim_{i,j} > Sim_{\theta} \\ FT_{ij}, & \text{if } Sim_{i,j} = Sim_{\theta} \\ FT_{ij} - \theta * (1 - FT_{ij}), & \text{if } Sim_{i,j} < Sim_{\theta} \end{cases} \tag{2.30}$$

where $\eta, \theta \in [0, 1]$ represent the degree of reward and punishment respectively. However, the punishment function is wrongly defined because it fails to limit FT_{ij} within the interval $[0, 1]$ as claimed.

Once again, SFTrust does not address decay of trust in absence of interaction but trust values should decay with elapse of time.

2.9 Dynamic Trust Model for Multi-agent Systems

The trust model provided by Li et al. [20] addresses different aspects in determining the trust of an agent such as recent trust, historical trust, expected trust and confidence in trust for other agents. They have also given a method to filter out inaccurate reports submitted by recommenders. Our main focus is on this trust model.

This model defines direct trust/local trust (*DT*) by averaging the satisfaction level of all the transactions as shown below-

$$DT_i(a, o) = \begin{cases} \frac{\sum_{i=1}^{m} S_i(a,o)}{m}, & \text{if } m \neq 0 \\ 0, & \text{else} \end{cases} \tag{2.31}$$

where m denotes the total number of interaction between agent *a* and agent *o*, and $S_i(a, o)$ represents the degree of satisfaction for the *i-th* transaction between agent *a* and agent *o*, and its defined as-

$$S_i(a, o) = \begin{cases} 1, & \text{if transaction totally satisfactory} \\ 0, & \text{if transaction totally unsatisfactory} \\ e \in (0, 1), & \text{else} \end{cases} \tag{2.32}$$

However, this simple averaging function defined in equation 2.31 fails to reflect the trend and relative time significance of the transactions.

Indirect trust or reputation is built from other agents' experience and Li et al. have defined reputation according to the following function-

$$Rep_i(a, o) = \frac{\sum_{w \in W} Cre_i(a, w) * DT_i(w, o)}{\sum_{w \in W} Cre_i(a, w)} \tag{2.33}$$

where W denotes the set of witnesses that have interacted with the target agent o and $Cre_i(a, w)$ represents the credibility of witness $w(w \in W)$. As we can see from the above equation that in taking recommendation from witnesses the credibility of the witness is also considered.

Li et al. then combined direct trust and reputation to define recent trust (*RT*) which reflects the

recent trend in an agent's behavior. Recent trust is defined as follows-

$$RT_i(a,o) = \begin{cases} \eta * DT_i(a,o) + (1-\eta)Rep_i(a,o), & \text{if } n < N \\ DT_i(a,o), & \text{else} \end{cases} \tag{2.34}$$

where η determines the weight of direct trust and it depends on the number of own interactions. Here, n denotes the number of own interaction in the *i-th* interval and N represents the threshold of interaction number. So, when $n < N$ the value of η equals n/N and when $n \geq N$, η equals 1. However, agent should never rely totally on own experience as a malicious agent can intentionally be honest to a given agent while it may be acting maliciously with other agents in the system. By taking recommendation from other agents that have interacted with the given target agent we can identify its actual nature.

Li et al. also defined historical trust (*HT*) which is built from pervious experiences and it reflects the long term behavioral pattern of an agent. They have defined historical trust by storing the recent trust value of the target agent up to a certain number of intervals ($maxH$). Historical trust is defined as follows-

$$HT_i(a,o) = \frac{\sum_{k=1} maxH\rho^{k-1} * RT_{i-k}(a,o)}{\sum_{k=1} maxH\rho^{k-1}} \tag{2.35}$$

where ρ ($0 \leq \rho \leq 1$) represents the forgetting factor i.e., with $0 \leq \rho \leq 1$, $HT_i(a,o)$ gives higher weight to recent values and lower weight to older values. However, the main problem with this formulation of historical trust is that it requires the storage of previous values and this results in storage overhead.

Li et al. combined recent trust with historical trust to define expected trust (*T*) which reflects the expected performance of the target agent. They defined expected trust by the following function-

$$T_i(a,o) = \begin{cases} default\_value, & \text{if neither } RT \text{ nor } HT \text{ available} \\ RT_i(a,o), & \text{if only } RT \text{ available} \\ HT_i(a,o), & \text{if only } HT \text{ available} \\ \min(RT_i(a,o), HT_i(a,o)), & \text{if both } RT \text{ and } HT \text{ available} \end{cases} \tag{2.36}$$

However, in case of both RT and HT being available taking min of these two values does not allow an agent to improve its trust value after any set back. In other words, suppose an agent suffered from some programming malfunction (e.g., programming bug or virus) and as a result could not provide effective service which ultimately reduced its reputation to others. Now, in the recent interval it has solved it problem and is behaving in a cooperative manner, so its recent trust has increased

significantly. However, if we consider the min of recent and historical trust then we will not be able to give any credit to its recent honesty but we should encourage agents to behave honestly and cooperatively.

In calculating credibility Li et al. first determined similarity and then used the similarity measure to define credibility. In order to determine the similarity with a given witness ($w \in W$), the evaluating agent (a) compares its direct trust for the target agent (o) with the direct trust experienced by the witness. Similarity (V) is defined as follows-

$$V_i(a,w) = \begin{cases} 1 - abs(DT_i(a,o) - DT_i(w,o)), & \text{if } abs(DT_i(a,o) - DT_i(w,o)) < \tau \\ 0, & \text{else} \end{cases} \tag{2.37}$$

where $\tau (0 < \tau < 1)$ represents the inaccurate tolerance threshold. So we see that the smaller the difference the higher the similarity. This similarity rating calculated at each interval is then stored and used to calculate credibility. Credibility (*Cre*) is defined according to the following function-

$$Cre_i(a,w) = \frac{\sum_{k=1}^{maxH} \rho^{k-1} * V_{i-k}(a,w)}{\sum_{k=1}^{maxH} \rho^{k-1}} \tag{2.38}$$

where ρ $(0 \leq \rho \leq 1)$ represents the forgetting factor. So, we can see that in order to calculate credibility we have to store the similarity measure for the last $maxH$ intervals and this causes storage overhead.

Unlike some of the trust models the trust model given by Li et al. also considered calculating a confidence factor which reflects the reliability of the trust value computed. For this purpose they defined three reliability factors: reliability in recent trust (*Rrt*), reliability in historical trust (*Rht*) and deviation reliability (*DR*). The main theme behind confidence factor is that as the number of interactions increase and as lower deviations are observed the more confident we become about the calculated trust value.

Reliability in recent trust in based on the number of interactions that were considered in the calculation of recent trust (*RT*). As this number of interaction grows the reliability in recent trust also increases until it reaches the highest attainable value of 1 (when the number of interaction exceeds a user defined threshold $maxN$). Reliability in recent trust is defined as follows-

$$Rrt_i(a,o) = \begin{cases} \sin(\frac{\Pi}{2} * (\frac{N_i(a,o) + \alpha \sum_{w \in W} Cre_i(a,w) * N_i(w,o)}{maxN})), \\ \quad \text{if } N_i(a,o) + \alpha \sum_{w \in W} Cre_i(a,w) * N_i(w,o) < maxN \\ 1, \quad \text{otherwise} \end{cases} \tag{2.39}$$

where $N_i(x, y)$ represents the number of interactions between agent *x* and agent *y* within the *i-th* interval, and $\alpha(0 \leq \alpha \leq 1)$ is used to discount the weight of the witness' interaction experience in calculating reliability in recent trust.

Reliability in historical trust is similarly based on the number of interactions that were considered in the calculation of historical trust (*HT*). Li et al. computed reliability in historical trust by storing *Rrt* values for the last $maxH$ intervals and then used the following averaging function to compute *Rht* value.

$$Rht_i(a, w) = \frac{\sum_{k=1}^{maxH} Rrt_{i-k}(a, o)}{maxH} \tag{2.40}$$

However, the main problem with this approach is not only does this function requires us to store previous values which adds to storage requirement but also it assigns equal weight to all the values. Logically recent values should receive higher weight than historical values.

Deviation reliability is used to signify the degree of deviation that is tolerated i.e., lower deviation means higher reliability. Deviation reliability is calculated from another metric named accumulated misused trust (*AT*) which keeps track of trust fluctuation and it is defined as follows-

$$AT_i(a, o) = \begin{cases} AT_{i-1}(a, o) + HT_i(a, o) - RT_i(a, o), & \text{if } RT_i(a, o) - HT_i(a, o) < -\varepsilon \\ AT_{i-1}(a, o), & \text{otherwise} \end{cases} \tag{2.41}$$

where ε represents the tolerated margin of error in the evaluation of trust. From equation 2.41 we see that accumulated misused trust (*AT*) only considers sudden fall of trust but sudden rise should also be considered as malicious agents tends to strategically oscillate between building and milking reputation. Deviation reliability is defined as follows-

$$DR_i(a, o) = \begin{cases} 0, & \text{if } AT_i(a, o) > maxAT \\ 1 - \sin(\frac{\Pi}{2} * \frac{AT_i(a,o)}{maxAT}), & \text{otherwise} \end{cases} \tag{2.42}$$

So from the above equation we can see that deviation reliability becomes zero when accumulated misused trust (*AT*) exceeds a given threshold $maxAT$. The main problem with this function is that the declination rate is higher in the initial stage and lower in the latter stage but realistically it should be the reverse i.e., in the initial stage there is small amount of deviation so the declination rate should be smaller and as more and more deviations occur the declination rate should increase significantly.

Finally Li et al. combined the reliability factors to define the confidence factors as shown below-

$$C_i(a, o) = (Rrt_i(a, o) + Rht_i(a, o)) * DR_i(a, o) \tag{2.43}$$

The final trust value is compute by combining expected trust with confidence factor as shown below-

$$Trust_i(a, o) = T_i(a, o) * C_i(a, o) \tag{2.44}$$

An agent uses this value to select the target agent with the highest value.

Like the previous models this trust model also does not address decay of trust in absence of interaction but trust values should decay with elapse of time in absence of interaction.

2.10 Users' Behavior Dependent Trust Model

Wen et al. [34] proposed a trust model which considers both direct and referral credibility. In case of referral credibility it uses the transitive law of trust. This model computes global trust by combining direct trust and indirect trust.

In case of direct trust they considered a set of attributes (taking into account the user's behavior or subjective aspect) for determining the satisfaction of a transaction. They defined direct trust by the following equation-

$$\overline{T_u} = \sum w_{a_i} * R_i \tag{2.45}$$

where T_u denotes the direct trust computed about agent *u*, $a_1, a_2, ..., a_n$ represents the set of attributes signifying the quality of a transaction and R_i denotes the rating of attribute a_i where $R_i \in (0, 1)$. The relative importance assigned to each attribute is modeled as a weight w_{a_i} where $\sum w_{a_i} = 1$.

Now, the attribute rating R_i is dynamically changed after each interaction according to the following equation-

$$R_i = \begin{cases} R_i^{old} + (R_i^{new} - R_i^{old}) * \beta^{(R_i^{new} - \mu + 1/n)}, & \text{if } R_i^{new} > \mu \\ R_i^{old} + (R_i^{new} - R_i^{old}) * \beta^{(R_i^{new} - \mu)}, & \text{if } R_i^{new} < \mu \ \& \ R_i^{new} < R_i^{old} \\ R_i^{old} + (R_i^{new} - R_i^{old}) * \beta^{(R_i^{new} - R_i^{old})}, & \text{if } R_i^{old} < R_i^{new} < \mu \end{cases} \tag{2.46}$$

where R_i^{old} and R_i^{new} denote the original and new values respectively. β $(0 < \beta < 1)$ is the subjective factor which controls the how quickly or slowly direct trust rises and falls. μ denotes the critical value of the trust i.e., trust values larger than μ are trustworthy and the smaller ones are distrustful. n denotes the number of the interactions.

Unlike the previous models discussed this trust model considers decay of trust value with the elapse of time in absence of interactions. The decay function that they incorporated is defined as

follows-

$$\widehat{T_u} = \frac{\overline{T_u}}{\Delta t^{1/k}} \tag{2.47}$$

where $\widehat{T_u}$ represents the value of trust after decay and Δt means the interval between the current interaction and the last interaction. k is a controlling factor and has different values in different networks. So we can see that, with Δt increasing, a trustworthy user may become an untrustworthy one in the trustworthy network. However, the main problem with this function is that the decay rate is initially high and it tends to saturate as time elapses but realistically the decay rate should be small at the initial stage and should increase as time elapses.

In case of 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. They have defined direct credibility for recommenders that have directly interacted with the desired target agent and it is defined by the following function-

$$c_i^{new} = c_i^{old} + \theta * (1 - d_i^{1/s} - c_i^{old}) \tag{2.48}$$

where, c_i^{new} and c_i^{old} are the new and original evaluations of the recommender *i* respectively. s is a strictness factor which is used to control the curve. θ is an impact factor, and its defined as-

$$\theta = \frac{e^{|1-d_i^{1/s}-c_i^{old}|} - 1}{e + 1} \tag{2.49}$$

where d_i is the deviation between the true trust value of the user and trust value given by recommender *i*, and it reflects the accuracy of the credibility evaluation. However, equation 2.48 fails to limit credibility within the stated range of $[0, 1]$ in some special cases. For example, in the experimental setup of [34] the strictness factor has been set to 1 and the initial value of credibility for any recommender has been set to 0.4 and since d_i denotes the deviation between the true trust value of the user and trust value given by recommender *i*, it can take any value between $[-1, 1]$. For worst case scenario we assume that $d_i = -1$ (i.e., the recommender totally disagrees with the user about a given target agent) then $|1 - d_i^{1/s} - c_i^{old}| = 1.6$ and $\theta = 1.063$ which causes c_i^{new} to be set to $1.7 > 1$.

For referral credibility this model has used the transitive law of trust i.e., if 'A' trusts 'B' and 'B' trusts 'C' then 'A' will also trust 'C'. In order to illustrate the evaluation of the referral credibility , an example is shown in Fig. 2.1, where user *i* can interact with the trustworthy network and user *j* is the recommender of the evaluated user and the other nodes are the intermediate users for the referral of credibility. As we can see from the figure there are *k* paths from *i* to *j*. If c_k denotes the credibility

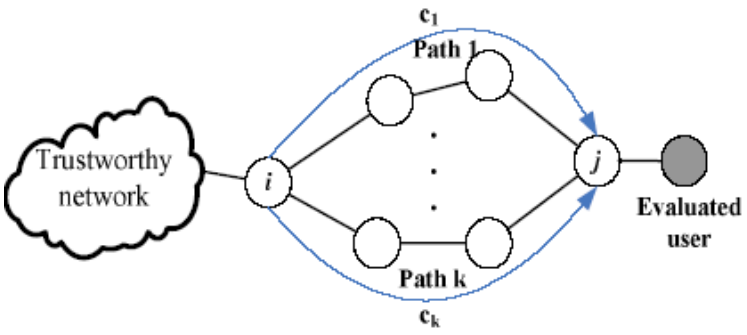


Figure 2.1: Referral credibility.

computed along path *k* then c_k is computed using the transitive law of trust according to the following function-

$$c_k = c_{i1} * c_{12} * \ldots * c_{mj} \tag{2.50}$$

where c_{xy} is the credibility of agent *y* from the viewpoint of agent *x*. So the total credibility of the recommender *j* can be obtained as follows:

$$c_j^{new} = \frac{1}{k} \sum_{i=1}^{k} c_k \tag{2.51}$$

However, in case of the referral credibility there exists the possibility of erroneous/false credibility propagation as we are multiplying the credibility of the intermediate agents and any one of the intermediate agents can act maliciously. Moreover, net credibility is computed by assigning equal weight to all the possible routes and then averaging them but longer the path the more possibility of that a malicious agent lies in it. So longer paths should not be given same weight as smaller paths.

Indirect trust is then calculated using the following equation-

$$\widetilde{T} = \frac{1}{m} \sum_{i=1}^{m} c_i^{new} * T_{i \to a} \tag{2.52}$$

where $T_{i \to a}$ represents the direct trust computed about agent *a* from agent *i*'s perspective and *m* is the number of recommenders.

Wen et al. combined direct trust and indirect trust to define global trust according to the following function-

$$T = w * \overline{T} + (1 - w) * \widetilde{T} \tag{2.53}$$

where w represents the weight of direct trust and it changes dynamically based on the agent's own interaction count. In other words, as the number of own interaction increases the weight of direct trust

also increases. w is calculated as follows-

$$w = \alpha^{k/n}, 0 < \alpha < 1 \text{ and } k \in \{1, 2, 3, ...\} \tag{2.54}$$

where n is the number of interactions that agent *i* has performed and α is the parameter that controls the curve. So from the above equation we see that as n increases so does w. However, the main problem with this formulation is that instead of considering the total number of interactions with the given target agent, Wen et al. have considered the total number of interactions the evaluating agent performs. Now, suppose agent *i* performs 0 transaction with agent *x* while it performs 100 interactions with agent *y* and then $n = 100$ which causes the value of w to be high (near 1). So, even though agent *i* has no experience about agent *x* still it will assign high weight to direct trust (own experience) according to equation 2.54 and this is a major drawback.

2.11 Decay Model

As discussed before the models defined in [12, 18, 20, 31–33, 35, 39] do not address a critical aspect of trust which is the decay of trust value with time. Since the network present today is highly dynamic and unpredictable, trust values should decay with the elapse of time. Some models [34, 38] have their own decay functions. However, their decay functions fail to properly simulate the decay process i.e., their decay functions have a high declination rate in the initial stage and a low declination rate at the latter stage. Realistically it should be the reverse that is the declination rate should be smaller in the earlier stage while it should be higher in the latter stage or it could even have a uniform decay rate and we have incorporated such a decay function.

2.12 Summary

The main concept behind developing a reputation based trust model is to combine both own experience and experience gained by others in the network. In considering experience from others their credibility must also be considered otherwise malicious agents will flood the network with false and inaccurate feedback. Another aspect that is important in assessing the nature of agents is their historical trend. Many models keep record of all the transactions that an agent performs but this approach is costly from spatial point of view. Moreover these models assign equal significance to all the previous

transactions which is not logical as recent transactions should be given higher significance than historical transactions. Finally, most of the trust models ignore the decay of trust with the elapse of time in absence of interactions but this is a critical aspect as agents nowadays show dynamic personality and we cannot just rely on their present nature.

Chapter 3

SECTrust:Our Trust Model

In this chapter we formally introduce our trust model SECTrust. A detailed description and analysis of the different parameters considered in computing the trust of an agent are discussed. The logical definition and mathematical expression used for each parameter/component are also presented in this chapter. We have also proposed an algorithm for filtering out dishonest recommenders from the recommendation membership set.

3.1 Introduction

The main objective of this thesis work is to provide a dynamic trust model for effectively evaluating the trust of agents even in the presence of highly oscillating malicious agents. A number of parameters have been considered in our trust model for computing the trust of an agent. Now, many of these parameters have been previously discussed in [12, 20, 31, 34, 35, 39] but none of these models can fully cope with the strategic adaptations made by the malicious agents. These strategic adaptations include- behaving maliciously after attaining high reputation, altering between cooperative and malicious nature with the intent of not getting noticed by other agents and so on. 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 all the parameters that we have considered. In the following sections we will 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 trust model called SECTrust. For the following sections we assume that agent $\boldsymbol{a}$ (called evaluator) needs to calculate the trustworthiness of agent $\boldsymbol{o}$ (called the target agent).

3.2 Schematic Flow Diagram of SECTrust

In this section we will show and briefly discuss the schematic flow of trust calculated from various components. As we can see from Fig. 3.1 that in computing trust an agent first computes direct trust from its own experience and then computes indirect trust from the recommendation of other agents. In computing direct trust it considers the credibility of the target agent (the agent being evaluated) and similarly in taking recommendation from other agents the evaluating agent also takes in account the recommending agent's credibility. Combining direct and indirect trust the agent calculates recent trust and this recent trust is used to compute historical trust as transactions proceed. Finally recent trust and historical trust is combined to define expected trust i.e., the trust that an agent places upon another agent to behave properly (or even maliciously).

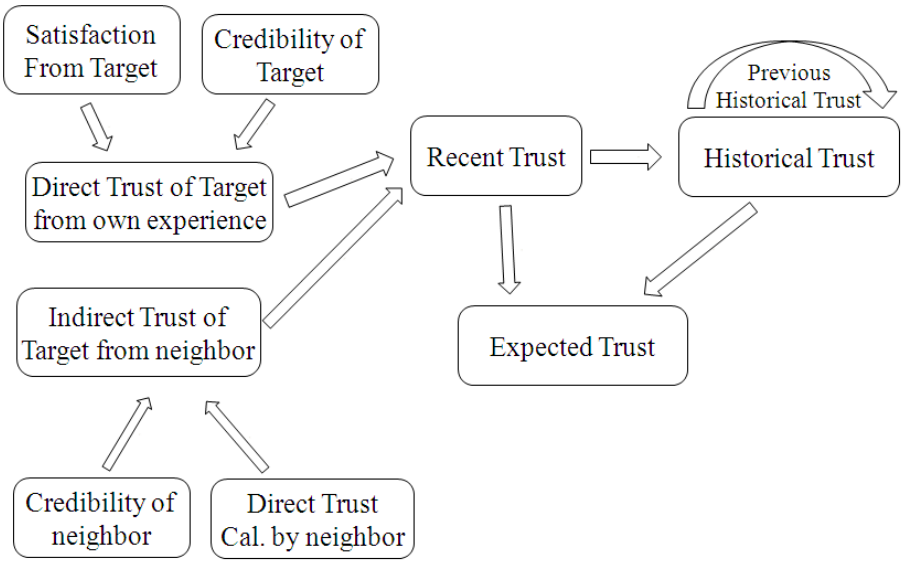


Figure 3.1: Schematic diagram of trust calculation.

After each transaction the evaluating agent updates the credibility of each recommender based on the satisfaction it receives from the transaction. Credibility is calculated from similarity which is computed from difference in satisfaction rating over the common set of interacting agents. This is shown in Fig 3.2.

Last of all we compute the level of confidence an agent has on the calculated trust value for another agent. The confidence factor is calculated from three components namely- reliability in recent trust, reliability in historical trust and deviation reliability. The philosophy behind computing reliability in recent trust and historical trust is that as more and more interactions occur an agent becomes more confident about its measured value. So, interaction count and credibility are used to define reli-

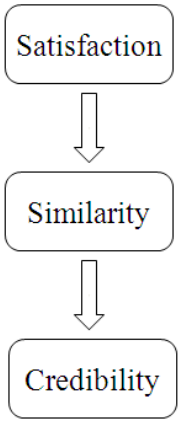


Figure 3.2: Schematic diagram of credibility calculation.

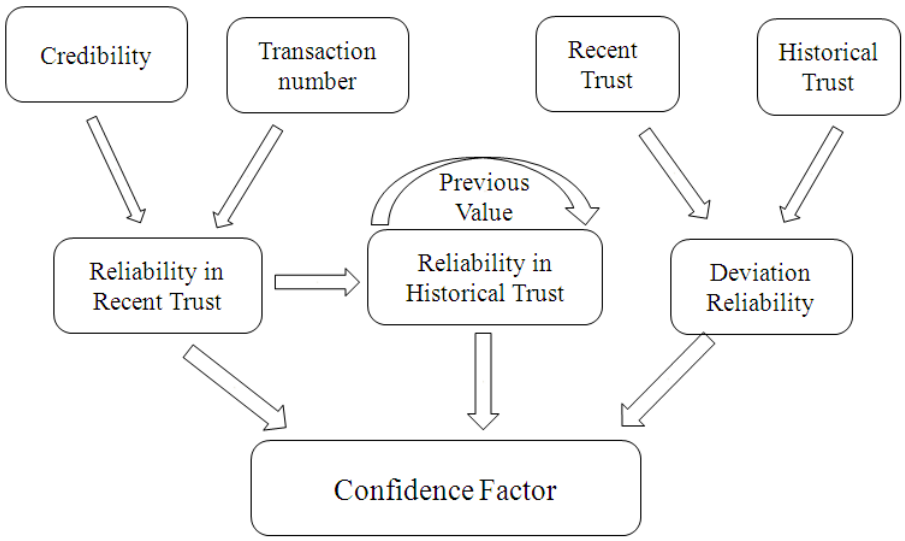


Figure 3.3: Schematic diagram of confidence factor calculation.

ability in recent trust and this values are used to calculate reliability in historical trust as interactions continue. Deviation reliability is used to measure trust fluctuation to prevent malicious agents from oscillating strategically. Here difference in recent trust and historical trust is used to calculate deviation reliability. Details of each of the components discussed are elaborately explained in the following sections.

3.3 Satisfaction

Satisfaction function 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 significance to the transactions. Let, $S_t^i(a,o)$ represent the amount of satisfaction agent *a* has upon agent *o* based on its service up to *i* transactions in the *t-th* time interval. The satisfaction update function is defined as follows-

$$S_t^i(a,o) = \alpha S_{cur} + (1-\alpha) S_t^{i-1}(a,o) \ . \tag{3.1}$$

$S_t^0(a,o) = S_{t-1}^{last}(a,o)$ and $S_0^0(a,o) = 0$. Here, S_{cur} represents the satisfaction value for the most recent transaction and we have used a binary feedback system where an agent rates other agents with either 1 or 0 based on whether the transaction is satisfactory or not.

$$S_{cur} = \begin{cases} 0, & \text{if recent transaction is not satisfactory} \\ 1, & \text{if recent transaction is satisfactory} \end{cases} \tag{3.2}$$

The weight α changes based on the accumulated deviation $\xi_t^i(a,o)$.

$$\alpha = threshold + c\frac{\delta_t^i(a,o)}{1+\xi_t^i(a,o)} \tag{3.3}$$

$$\delta_t^i(a,o) = |S_t^{i-1}(a,o) - S_{cur}| \tag{3.4}$$

$$\xi_t^i(a,o) = c\delta_t^i(a,o) + (1-c)\xi_t^{i-1}(a,o) \tag{3.5}$$

$\xi_t^0(a,o) = \xi_{t-1}^{last}(a,o)$ and $\xi_0^0(a,o) = 0$. Here c is a user define constant which controls to what extent we will react to the recent error ($\delta_t^i(a,o)$). So, if we increase the value of c then we give more significance to the recent deviation than the accumulated deviation and vice versa. The $threshold$ is used to prevent α from saturating to a fixed value. For example if α ever becomes 0 then it will totally ignore the satisfaction level of the current transaction (S_{cur}) and as a result $S_t^i(a,o)$ (from eqn 3.1) will always remain constant irrespective of whatever kind of transactions follow. Initial value of α is set to 1 and $threshold$ is set to 0.25.

3.4 Similarity

To compute similarity we have first determined personalized difference rating $D_t^i(a,o)$ between agent *a* and agent *o*. Then we use $D_t^i(a,o)$ to define the dynamic similarity update function. Let, $PS(a)$ represent the set of agents with whom agent *a* has made interaction. Then $CIS(a,o) = PS(a) \bigcap PS(o)$ denotes the set of agents with whom both agent *a* and agent *o* have interacted.

$$D_t^i(a,o) = \sqrt{\frac{\sum_{x \in CIS(a,o)} (S_t^i(a,x) - S_t^i(o,x))^2}{|CIS(a,o)|}} \tag{3.6}$$

To measure the similarity between agent *a* and agent *o* ($Sim_t^i(a,o)$), agent *a* compares $D_t^i(a,o)$ with the similarity deviation constant (τ) and then updates similarity according to the following function-

$$Sim_t^i(a,o) = \begin{cases} Sim_t^{i-1}(a,o) + \frac{1-Sim_t^{i-1}(a,o)}{\mu}, & \text{if } D_t^i(a,o) < \tau \\ Sim_t^{i-1}(a,o) - \frac{Sim_t^{i-1}(a,o)}{\psi}, & \text{else} \end{cases} \tag{3.7}$$

where μ represents rewarding factor and ψ represents punishment factor and both of them can be changed dynamically depending on the system. In equation (3.7) we see that when the difference rating is less than τ similarity increases and otherwise it decreases. Here $\mu, \psi \in \Re$ and we must make sure that 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, $|CIS(a,o)| = 0$ then $D_t^i(a,o) = 0$ and $Sim_t^i(a,o) = 0.5$ (default value).

3.5 Credibility

Credibility is used to measure the degree of accuracy of the feedback information that the feedback/recommending agent provides to the evaluator. During trust evaluation, the feedbacks provided by agents with higher credibility are trustworthier, and are therefore weighted more than those from agents with lower credibility. Credibility is a critical measure to filter out inaccurate report from malicious agents since malicious agents very often provide dishonest or unfair feedback. Now, the nature of credibility is to gradually build and move towards the highest attainable value. So, it emulates a logarithmic curve and for this purpose we have used a logarithmic curve to define credibility. Let, $Cre_t^i(a,o)$ present the credibility of agent *o* from agent *a*'s perspective.

$$Cre_t^i(a,o) = \begin{cases} 1 - \frac{\ln(Sim_t^i(a,o))}{\ln\theta}, & \text{if } Sim_t^i(a,o) > \theta \\ 0, & \text{else} \end{cases} \tag{3.8}$$

where $\theta = 0.01$ represents the lowest allowed value of similarity. As we can see from equation (3.8) that credibility is a direct logarithmic function of similarity which means that the credibility of agents will slowly rise towards it highest attainable value as similarity increases. So, the "*slow rise and quick decline*" principle is also incorporated in the credibility function.

3.6 Direct Trust

Direct trust also known as local trust represents the portion of the trust that an agent computes from its own experience with the target agent. Let, $DT_t^i(a, o)$ represent the direct trust that agent *a* has upon agent *o* up to *i* transactions in the *t-th* interval. We have used the satisfaction measure along with the credibility of the target agent to define direct trust.

$$DT_t^i(a, o) = S_t^i(a, o) * Cre_t^i(a, o) \tag{3.9}$$

where $Cre_t^i(a, o)$ represents the credibility of agent *o* from agent *a*'s perspective. Here credibility is being considered as malicious agents may intentionally try to be honest towards certain agents while it may be acting maliciously with other agents in the system. Now, since credibility is calculated based on similarity and similarity is computed from the difference in satisfaction rating over the common set of interacted agents, malicious agents will not be able to fool the evaluating agent if its credibility is considered. Thats why we are considering credibility in computing direct trust.

3.7 Indirect Trust

Indirect trust is computed from the experience of other agents in the system. When there is no or even little interaction with the target agent, an agent has to depend on the experience gained by other agents to make a fruitful decision. Indirect trust is computed by considering recommendation from other agents along with the credibility of the recommender. Let, $IT_t^i(a, o)$ represent the indirect trust that agent *a* computes about agent *o*.

$$IT_t^i(a, o) = \frac{\sum_{x \in W - \{a\}} Cre_t^i(a, x) * DT_t^i(x, o)}{\sum_{x \in W - \{a\}} Cre_t^i(a, x)} \tag{3.10}$$

Here $W = IS(o)$, represents the set of agents who have ever interacted with agent *o*. If $|W - \{a\}| = 0$ then $IT_t^i(a, o) = 0$.

3.8 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 more and more interactions occur with the target agent, that is, the evaluator becomes more confident about its own experience than

taking recommendation from others. Let, $RT_t^i(a,o)$ represent the recent trust that agent *a* has upon agent *o*.

$$RT_t^i(a,o) = \beta DT_t^i(a,o) + (1-\beta)IT_t^i(a,o) \tag{3.11}$$

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

$$\begin{aligned} \beta &= \frac{I_t(a,o)}{I_t(a,o) + M_t(a,o)} \\ M_t(a,o) &= \frac{\sum_{x \in W-\{a\}} I_t(x,o)}{|W-\{a\}|} \end{aligned}$$

Here $W = IS(o)$, represents the set of agents who have ever interacted with agent *o* and $I_t(a,o)$ represents the number of interactions agent *a* has conducted with agent *o* in the *t-th* interval. So, $M_t(a,o)$ represents the mean number of interactions that other agents (agents other than *a*) have conducted with agent *o*. As we can see from the above equation that as $I_t(a,o)$ increases compared to $M_t(a,o)$ the contribution of direct trust to recent trust also increases i.e., as the evaluating agent's own experience exceeds the mean experience gained by the recommenders, the evaluating agent relies more on its own experience in calculating trust. However, if $|W-\{a\}| = 0$ then $M_t(a,o) = 0$ and if $I_t(a,o) + M_t(a,o) = 0$ then $\beta = 0.5$ (default value).

3.9 Historical Trust

Historical trust is built from past experience and it reflects long term behavioral pattern. We have defined historical using exponential averaging function. Let, $HT_t^i(a,o)$ represent the historical trust that agent *a* has about agent *o*.

$$HT_t^i(a,o) = \frac{\rho * HT_t^{i-1}(a,o) + RT_t^{i-1}(a,o)}{2} \tag{3.12}$$

where $\rho(0 \leq \rho \leq 1)$ is the forgetting factor (discounting older experiences) and $HT_0^0(a,o) = 0$. As we can see from the above expression recent value ($RT_t^{i-1}(a,o)$) is given higher weight than the past value ($HT_t^{i-1}(a,o)$). With historical trust present malicious agents cannot suddenly forget their past and start behaving good. In other words, as we are storing an agent's historical trend it cannot just fool other agents by suddenly behaving in a cooperative manner. For a malicious agent to be considered as a good agent it must behave in a cooperate manner for a significantly large number of transactions so that it's recent trend can replace most of it's historical trend.

3.10 Expected Trust

Expected trust reflects expected performance of the target agent and it is deduced from both recent and historical trust. Let, $ET_t^i(a, o)$ represent the expected trust of agent o from agent a's perspective. Expected trust is calculated for the following cases-

Case 1 : If agent a has not gained either recent or historical trust regarding agent o then expected trust is defined as follows:

$ET_t^i(a, o) = 0.$

Case 2 : If agent a can gain only recent trust about agent o then expected trust is defined as follows:

$ET_t^i(a, o) = RT_t^i(a, o).$

Case 3 : If agent a can gain only historical trust about agent o then expected trust is defined as follows:

$ET_t^i(a, o) = HT_t^i(a, o).$

Case 4 : If agent a can gain both recent and historical trust about agent o then expected trust is defined as follows:

$ET_t^i(a, o) = \eta_o RT_t^i(a, o) + (1 - \eta_o) HT_t^i(a, o).$

where η_o represents the contribution of recent trust.

Initially η_o is set to 0.5 but η_o adjusts dynamically based on the difference of recent and historical trust (deviation factor ε). Here we are allowing an agent with the scope of improving its expected trust through cooperative interaction i.e., we are rewarding the benevolent behavior of agents in the recent interactions by increasing η_o when recent trust exceeds historical trust by a given threshold (ε).

$$\eta_o = \begin{cases} \eta_o + 0.1, & \text{if } RT_t^i(a, o) - HT_t^i(a, o) > \varepsilon \\ \eta_o - 0.1, & \text{if } RT_t^i(a, o) - HT_t^i(a, o) < -\varepsilon \\ \eta_o, & \text{if } -\varepsilon < RT_t^i(a, o) - HT_t^i(a, o) < \varepsilon \end{cases} \tag{3.13}$$

As we see from the above mathematical expression that when recent trust exceeds historical trust by ε the value of η increases by 0.1 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 past trust. 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.

3.11 Decay Model

Due to easy access and abundant resource agents today are highly unpredictable and as a result their trust value should decline in absence of interaction. In other words, if an agent does not make a transaction with the network for a long period, its trust value will gradually degrade with the lapse of time. 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-

$$\widehat{S}_t^i(a,o) = S_t^i(a,o)e^{-\lambda\Delta t} \tag{3.14}$$

$$\Delta t = t_{current} - t_{previous} \tag{3.15}$$

where $\widehat{S}_t^i(a,o)$ represents the value of satisfaction after decay. Here, λ is the decay constant and its controls how quickly the value will diminish to zero. $\Delta t(\Delta t \geq 0)$ represents the interval between the current interaction and the last interaction. So as Δt increases a trustworthy agent becomes untrustworthy in the network. By including the time decay model we establish the principle "*the more recent the transaction the more reliable it is*".

3.12 Confidence Factors

Confidence is the metric that reflects the reliability of the trust calculated by an agent. Li et al. [20] have define some confidence factors. We address some of the shortcomings that their metrics have and at the same time redefine them to overcome the shortcomings. Confidence factor has three component- reliability in recent trust, reliability in historical trust and deviation reliability.

3.12.1 Reliability in Recent Trust

Reliability increases as more and more interactions occur and agents' behavior becomes more and more evident. So reliability in recent trust is based on the number of interactions which have been considered during the computation of $RT_t^i(a,o)$. While considering the interaction count of any recommender its credibility is also taken into account i.e., recommenders with higher credibility are given more weight than lower credible recommenders. Let, $N_t(a,o)$ represent the total number of

interactions (both direct and indirect) considered during the computation of $RT_t^i(a,o)$.

$$N_t(a,o) = I_t(a,o) + \sum_{x \in W - \{a\}} Cre_t^i(a,o) * I_t(x,o) \tag{3.16}$$

Reliability in recent trust increases as $N_t(a,o)$ increases until it reaches a user defined threshold ($maxN$), where it saturates to the maximum value 1. Let, $RRT_t^i(a,o)$ represent the reliability in recent trust computed at agent *a* about agent *o*.

$$RRT_t^i(a,o) = \begin{cases} \sin(\frac{\Pi}{2} * (\frac{N_t(a,o)}{maxN})), & \text{if } N_t(a,o) < maxN \\ 1, & \text{otherwise} \end{cases} \tag{3.17}$$

The value of the parameter $maxN$ is application dependent and $RRT_0^0(a,o) = 0$.

3.12.2 Reliability in Historical Trust

Reliability in historical trust is based on the number of interactions considered during the computation of $HT_t^i(a,o)$. Since $HT_t^i(a,o)$ is computed from past recent trust $RT_t^k(a,o)$ $(0 \leq k < i)$, reliability in historical trust is also computed from previous reliability in recent trust $RRT_t^k(a,o)$ $(0 \leq k < i)$. Previously, Li et al. [20] stored a number of past recent reliability to compute reliability in historical trust, but as mentioned earlier this causes storage overhead. Moreover they assigned equal weight to all the values, but recent values should be given higher weight than past values. To minimize storage overhead we use an exponential averaging function for this purpose. Let, $RHT_t^i(a,o)$ represent the reliability in historical trust that agent *a* has about agent *o*.

$$RHT_t^i(a,o) = \frac{\rho * RHT_t^{i-1}(a,o) + RRT_t^{i-1}(a,o)}{2} \tag{3.18}$$

where $\rho (0 \leq \rho \leq 1)$ is the forgetting factor (discounting older experiences) and $RHT_0^0(a,o) = 0$. From the above equation we can see that previous value are given lower weight than recent values.

3.12.3 Deviation Factor

Deviation reliability is a measure of how much deviation we are willing to tolerate. Malicious agents sometimes strategically oscillate between raising their trust and milking the reputation which seriously affects the performance of the network. So, some kind 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 misused trust* (denoted as $AT_t^i(a,o)$).

$$AT_t^i(a,o) = \begin{cases} AT_t^{i-1}(a,o) + \frac{RT_t^i(a,o) - HT_t^i(a,o)}{\omega}, & \text{if } RT_t^i(a,o) - HT_t^i(a,o) > \varphi \\ AT_t^{i-1}(a,o) + HT_t^i(a,o) - RT_t^i(a,o), & \text{if } RT_t^i(a,o) - HT_t^i(a,o) < -\varphi \\ AT_t^{i-1}(a,o), & \text{otherwise} \end{cases} \tag{3.19}$$

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 (3.19) we see that we are considering both sudden rise and fall of trust by agents whereas Li et al. [20] considered only sudden fall of trust. However, we are penalizing lesser for sudden rise 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 misused trust will be less than $RT_t^i(a,o) - HT_t^i(a,o)$ for sudden rise where it is exactly $HT_t^i(a,o) - RT_t^i(a,o)$ for sudden fall. Initial value of accumulated misused trust is $AT_0^0(a,o) = 0$.

Deviation reliability uses the accumulated misused trust metric to measure the deviation in agent behavior. Deviation reliability (denoted $DR_t^i(a,o)$) is defined by the following equation-

$$DR_t^i(a,o) = \begin{cases} 0, & \text{if } AT_t^i(a,o) > maxAT \\ \cos(\frac{\Pi}{2} * \frac{AT_t^i(a,o)}{maxAT}), & \text{otherwise} \end{cases} \tag{3.20}$$

where $maxAT$ represents the maximum tolerable misused trust. So from equation 3.20 we can see that when the accumulated misused trust exceeds a given threshold the deviation reliability becomes zero and when there is no misused trust the deviation reliability equals 1. 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 occur the degradation rate increases. Here, the initial value of deviation reliability $DR_t^i(a,o) = 1$.

The overall confidence is defined by using the above three reliability factor as shown below-

$$C_t^i(a,o) = (RRT_t^i(a,o) + RHT_t^i(a,o)) * DR_t^i(a,o) \tag{3.21}$$

Here we are combining recent and historical reliability and then multiplying it by the deviation reliability. So, if an agent fluctuates between building and milking reputation its deviation reliability will degrade and eventually its confidence factor will fall.

3.13 Trust Metric

The trust metric is the actual value used in prioritizing all agents. It is computed from expected trust and confidence. Let, $T_t^i(a, o)$ represent the final trust value agent *a* places upon agent *o*.

$$T_t^i(a, o) = ET_t^i(a, o) * C_t^i(a, o) \tag{3.22}$$

So we can see that for an agent to attain a high trust value it must not only have a high trust value but also it's confidence factor must be high. An agent will use equation (3.22) to select the target agent with the highest T value.

3.14 Filtering Out Dishonest Recommenders

In this section we discuss a methodology to filter out dishonest recommenders from the recommendation membership set (shown in Algorithm 1). In computing indirect trust (IT) about a certain target agent we consult other agents that have interacted with the target agent, but malicious agents often provide false feedback to obscure the true nature of the target agent. This type of false feedback can lead to wrong interpretation about an agent's capability and hence hinder effective communication. In equation 3.10 we have taken feedback from every agent who has ever interacted with the target agent. However, the evaluating agent might not have interacted with many of these recommenders as a result, the evaluating agent has no way of knowing how much credible they are and thus has to take the risk of relying on unknown recommenders. So we propose the following algorithm for filtering out dishonest recommenders.

Let, W represent the set of recommenders giving feedback to the evaluating agent *u* about the target agent *v*. Firstly, agent *u* will compute $W = PS(u) \bigcap IS(v)$ i.e., the set of agents who have interacted with agent *v* and with whom agent *u* has made transactions. If $|W| = 0$ then agent *u* will set $W = IS(v)$ i.e., the set of agents who have interacted with agent *v* and thus has to take the risk of relying on some unknown recommenders. Then agent *u* will discard recommenders with credibility less than γ. The pseudo-code of the algorithm is given as follows.

Algorithm 1 compute_recommender_set(u, v)

Input: $Evaluating\ agent$ **u** $and\ target\ agent$ **v**

Output: $Set\ of\ Recommenders$ **W**

$set\ W = PS(u) \bigcap IS(v)$

if $W = \emptyset$ **then**

$set\ W = IS(v) - \{u\}$

end if

if $W \neq \emptyset$ **then**

for $each\ x \in W$ **do**

if $Cre(u, x) < \gamma$ **then**

$W = W - \{x\}$

end if

end for

end if

return W

3.15 Summary

To summarize, our trust model considers a wide variety of factors in calculating the trust of an agent. All these factors are accumulated to define a single value which represents the trust that an agent has upon another agent. In our trust model we have introduced some new parameters in computing trust like- satisfaction and deviation of trust while at the same time we have refined most of the other parameters to address their former shortcomings. We have also applied a novel policy of using exponential averaging function to reduce storage overhead in computing trust. Moreover, we have described an algorithm to prevent dishonest recommenders from providing false recommendation.

Chapter 4

Experimental Evaluation

This chapter evaluates SECTrust's performance and shows its robustness and effectiveness in suppressing malicious activities. For this purpose we have conducted several experiments from different point of view like- parameter tuning, error calculation, comparative study and statistical analysis. All these experiments are elaborately explained in following sections.

4.1 Introduction

Simulation has been used extensively for evaluation of real-world systems including those used in communications and networking. Any practical research work needs to be verified under real world scenarios before it can be established. However, having access to ready-to-use test bed infrastructures is not feasible in most cases and this is where simulation programs or applications paly a vital role. Simulation provide us with a means to determine how the proposed system or scheme will act in real-life scenarios without actually deploying the system. In other words, the performance tests of the proposed schemes through simulations normally create the foundation and confidence for applying them in real-world environments. Therefore, simulations have become a integral part of most research work.

We have carried out our simulations to achieve three main objectives. Firstly, we evaluate our trust model's accuracy in terms of trust computation in the presence of malicious agents. Secondly, we assess how quickly it adapts to strategically oscillating behavior. Lastly, we demonstrates the effectiveness of SECTrust compared to other existing trust models under different scenarios.

4.2 Simulation Setup

We have developed our simulation in java language using JBuilder [2] and from the hardware point of view we used a 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.

4.2.1 Environment Setting

Our simulated environment contains N agents. In one experiment we have varied the value of N to show the scalability of the trust model. The experiment showed 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 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 *mal* which is varied in different experiments.

4.2.2 Agent's Behavioral Pattern

The behavioral pattern of good agents is quite easy to simulate as the provide good service and honest feedback. However, it is a challenging task to simulate agent's malicious behavior realistically. We mainly study three behavioral pattern namely- noncollusive, collusive and strategically altering setting. In noncollusive setting malicious agents cheat during transaction and give false feedback to other agents i.e., they rate good agents poorly while giving high ratings to malicious agents. 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. We use the parameter *cm_rate* to control 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 *mres* 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.

4.2.3 Transaction Setting

Two types of transaction setting are simulated, namely, random setting and trust prioritized 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 the parameter *res_rate*. The initiating agent then sort the responders based on their trust value and select the agent with the highest trust value to performing the desired transaction.

Table 4.1 summarizes the the different parameters related to the environment setting and trust computing. The table also lists the default value of the different parameters used. These default values have been empirically tuned.

Table 4.1: Simulation parameters settings.

	Parameter	Description	Default value
Environment Setting	N	# of agents in the system	100
	mal	% of agents malicious in the system	40%
	mres	% of time a malicious agents gives false feedback	100%
	res_rate	% of agents who respond to a transaction request	5%
	cm_rate	% 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	0.2
	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	0.1
	$maxN$	threshold for designated number of interaction	10
	φ	tolerated margin of error	0.25
	ω	punishment factor for sudden rise in trust	2
	$maxAT$	threshold for accumulated misused trust	10
	γ	credibility threshold for filtering out inaccurate recommenders	0.8

4.2.4 Performance Evaluation Index

We are mainly using two types of evaluation performance index namely: Trust Computation Error and Successful Transaction Rate (STR). Trust computation error is calculated by taking the root-mean-square (RMS) of the computed trust value of all the agents against their binary rating which is 1 for good agents and 0 for malicious agents. To compare the performance of SECTrust with other existing trust models we use successful transaction rate (STR). STR is described as the ratio of the number of successful transactions to the total number of transactions. Since the computed trust values are different for the different trust models, other evaluation index such as trust computation error is not suitable for comparison because it really doesn't 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 value is comparable and thats why we only compute STR for comparison with other models. We determine STR against the variation of *mal*, *mres* and *cm_rate*. All the experimental results are averaged over 30 runs of the experiment.

4.3 Tuning Different Parameters of the Trust Model

In this section we will tune the different parameters used in our trust model to obtain the best configuration. For this purpose we are computing the trust computation error against the variation of percentage of malicious agents (*mal*) under different values of the corresponding parameter. The following set of figures (Figs. 4.1-4.5) show the impact of the variation of the parameter values. We select the parameter value for which the trust computation error is lowest.

4.4 Trust Computation Accuracy

The objective of this set of experiments is to show the effectiveness and robustness of SECTrust against various malicious behavior. The experiments starts as agents randomly start interacting with each other. After each agent performs an average of 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 of all the agents against

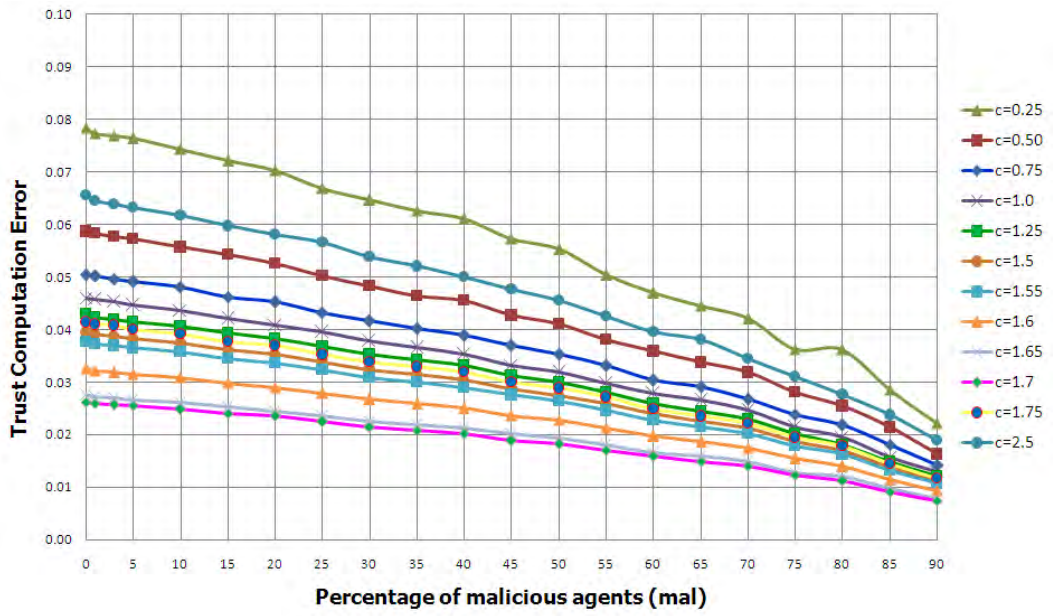


Figure 4.1: Trust computation error with respect to percentage of malicious agents(*mal*) for different values of *c*.

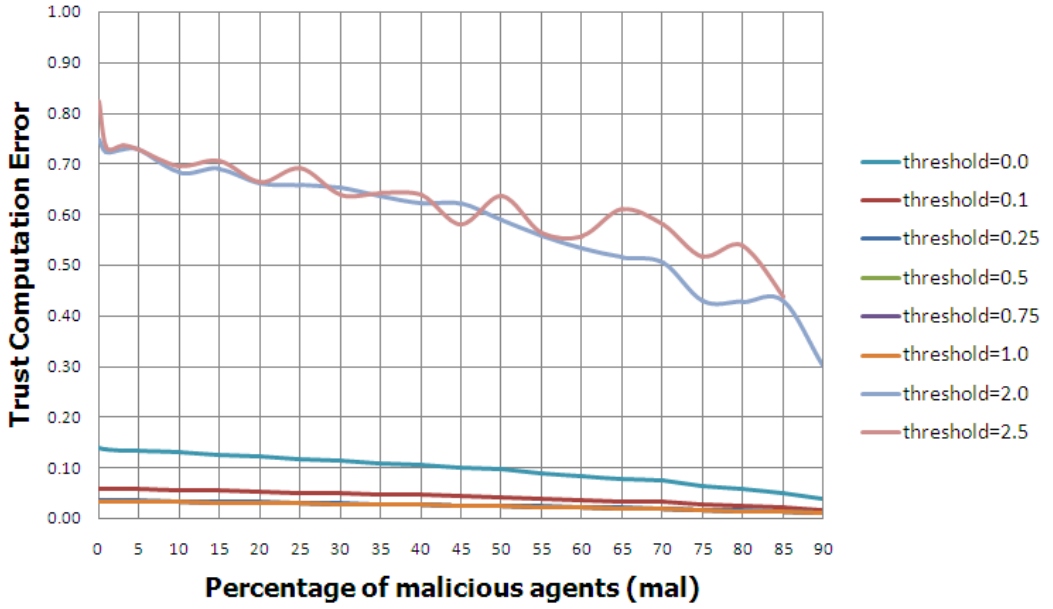


Figure 4.2: Trust computation error with respect to percentage of malicious agents(*mal*) for different values of *threshold*.

the actual likelihood of the agents performing a satisfactory transaction, which is 1 for good agents and 0 for malicious agents. The more lower the error the more accurate the model is.

In the first experiment we vary the percentage of malicious agents (*mal*) in the system while setting *mres* to 100% (i.e., malicious agents give false feedback in every interaction). Fig. 4.6 represents the

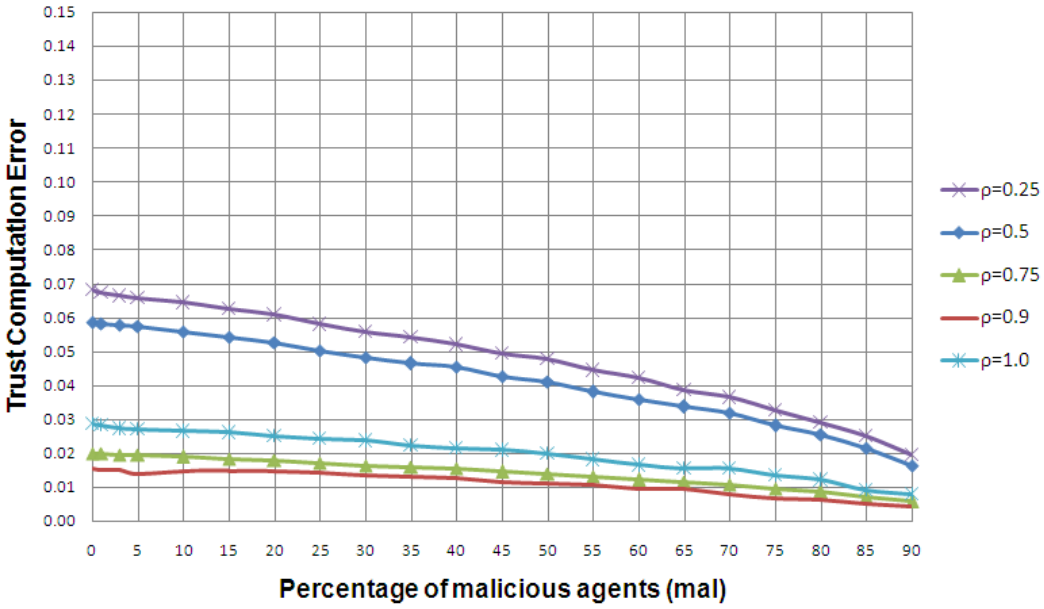


Figure 4.3: Trust computation error with respect to percentage of malicious agents(*mal*) for different values of forgetting factor ρ.

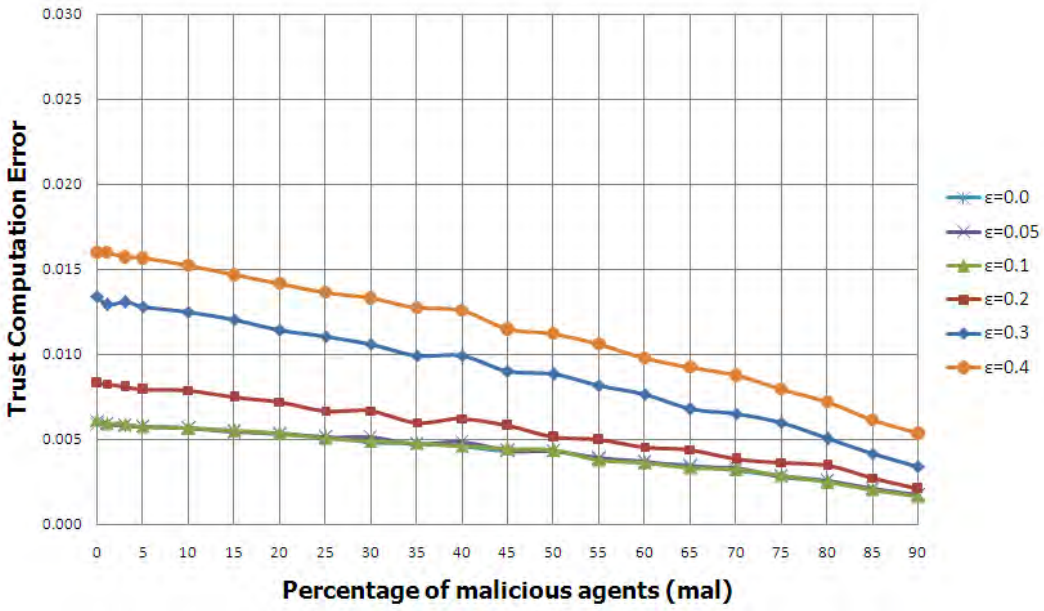


Figure 4.4: Trust computation error with respect to percentage of malicious agents(*mal*) for different values of deviation factor ε.

trust computation error with respect to *mal* in the two settings. In the noncollusive (*cm_rate* set to 0%) setting in Fig. 4.6(a) we see that SECTrust remains more effective in the presence of large percentage of malicious agents. The reason behind this is that the credibility measure effectively filters out the dishonest feedback submitted by malicious agents. In the collusive setting in Fig. 4.6(b)

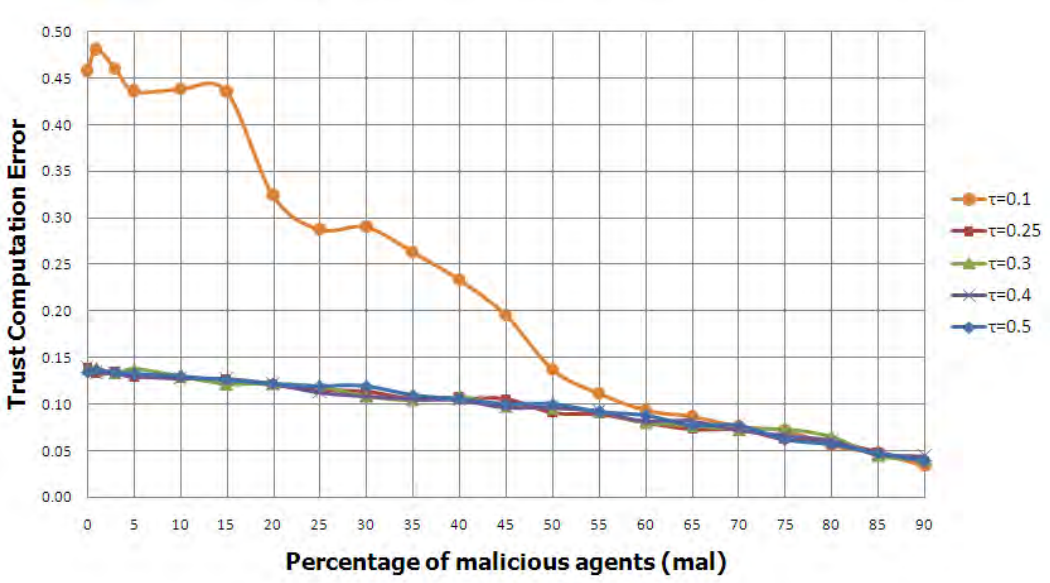


Figure 4.5: Trust computation error with respect to percentage of malicious agents(*mal*) for different values of similarity deviation factor τ.

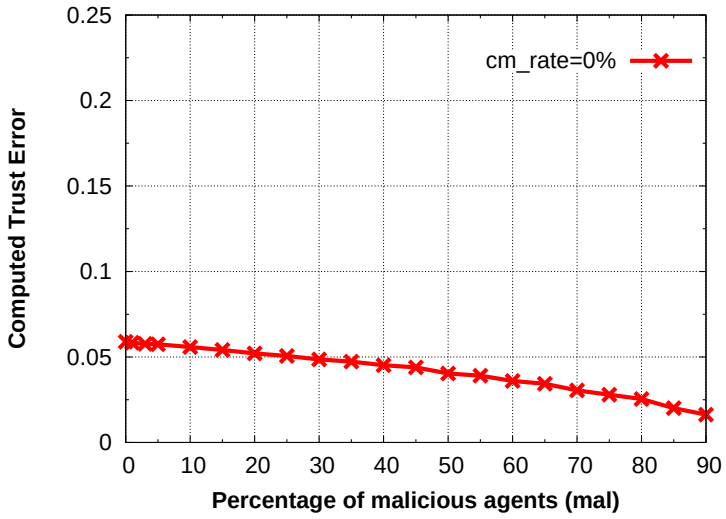
we set *cm_rate* to 100% and vary *mal*. Again we see that our model efficiently discards the dishonest feedback provided by the collusive group. This proofs that our similarity measure appropriately computes the credibility of the recommenders providing recommendation.

In the second experiment we vary *mres* while setting *mal* to 40%. Fig. 4.7 represents the trust computation error in both noncollusive and collusive setting. In the noncollusive setting (Fig. 4.7(a)) we see that the error is slightly high when *mres* varies from 35% to 50% signifying that malicious agents can confuse the system a little when they oscillate between good and malicious nature equally. For collusive setting (Fig. 4.7(b)) where *cm_rate* is set to 100%, we see even a better result in the presence of collusion. This signifies that SECTrust can successfully discard false rating provided by collusive groups.

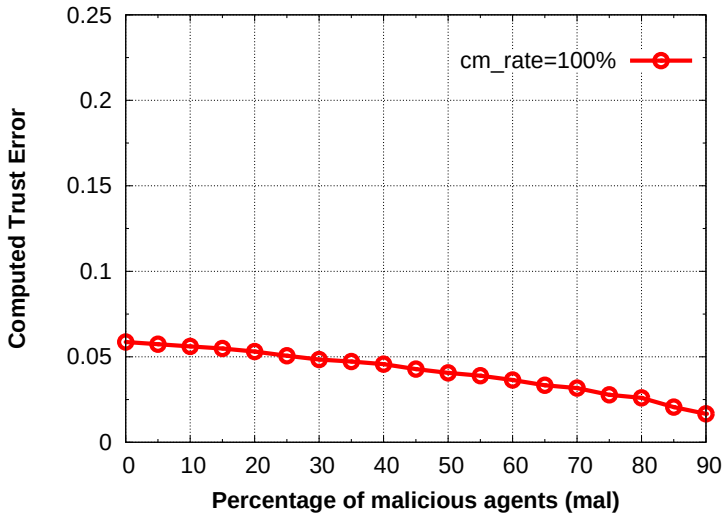
In the third experiment we vary *cm_rate* while setting *mal* to 40% and *mres* to 100%. Again from Fig. 4.8 we see that or trust model can effectively discards the impact of the collusive interaction due to the sensitive credibility measurement.

4.5 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 SECTrust handles dynamic change in agent behavior. We have already showed SEC-



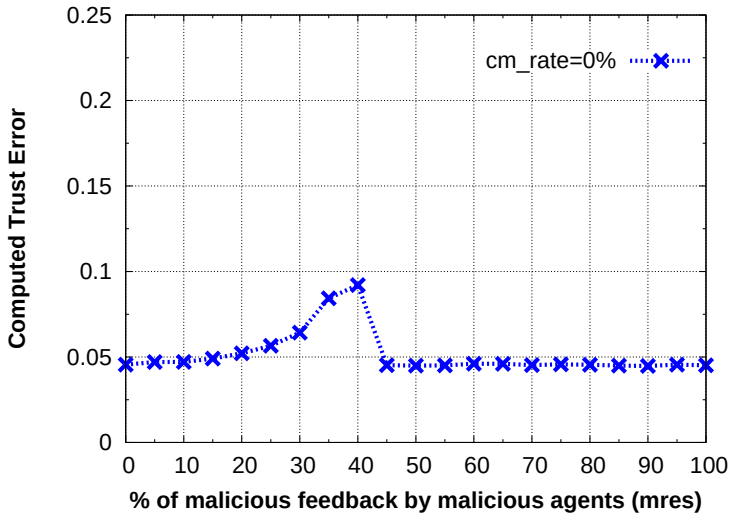
(a)



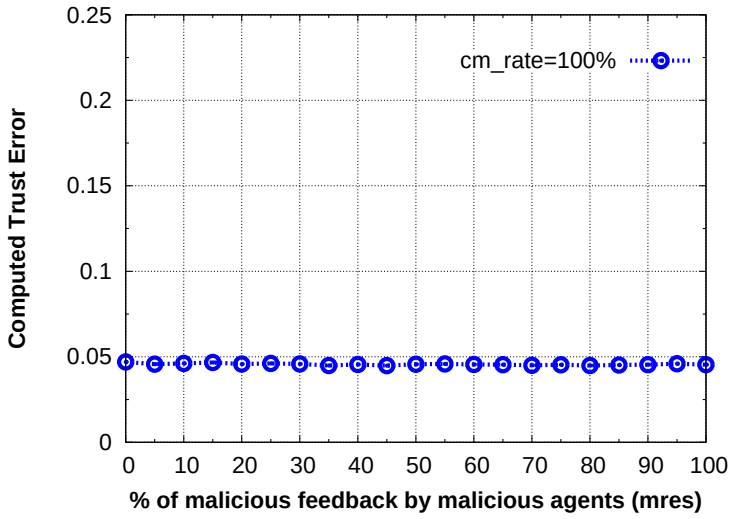
(b)

Figure 4.6: Trust computation error with respect to percentage of malicious agents for (a) noncollusive setting and (b) collusive setting.

Trust's effectiveness against filtering out dishonest feedback submitted by malicious agents, so in this experiment we concentrate on altering 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 again tries to build its reputation back i.e., the agent oscillates between building and milking reputation. For testing such scenario we simulate a environment which contains all good agents except



(a)



(b)

Figure 4.7: Trust computation error with respect to percentage of malicious response by malicious agents for (a) noncollusive setting and (b) collusive setting.

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 total 1000 interactions which are equally divided into four consecutive slots . The malicious agent then alternates between good and malicious nature from one slot to the next starting with good nature.

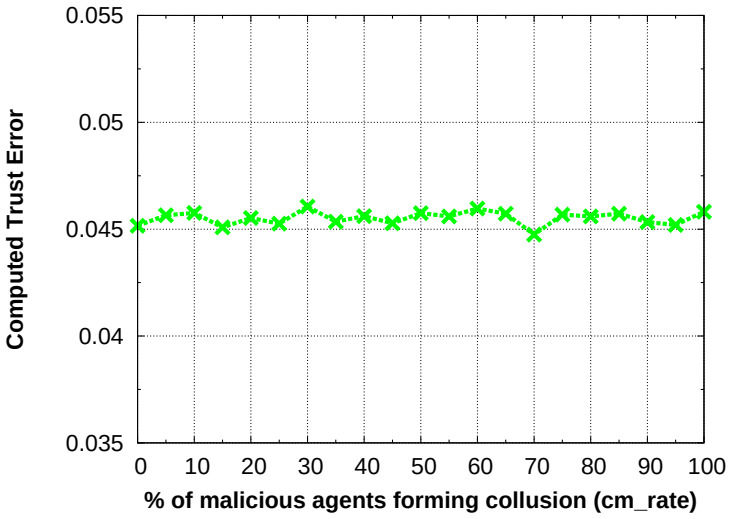


Figure 4.8: Trust computation error with respect to percentage of malicious agents forming collusive group.

Fig. 4.9 shows the computed trust value of the malicious agent under altering behavioral pattern. We see that SECTrust quickly responds to the sudden fall of performance by the malicious agent once it builds up its reputation. The sharp fall in the curve signifies this. Once the trust value diminishes to zero it requires significant number of consecutive good services for its trust value to rise again. From Fig. 4.9 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 the benefit gained from milking it. That is the model successfully incorporates the principle "*quick decline and slow rise of trust value*". Fig. 4.9 also shows the impact of different values of reward factor μ and punishment factor ψ.

4.6 Comparison with other Trust Models

In this set of experiments we will demonstrate the efficiency of SECTrust 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 against 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

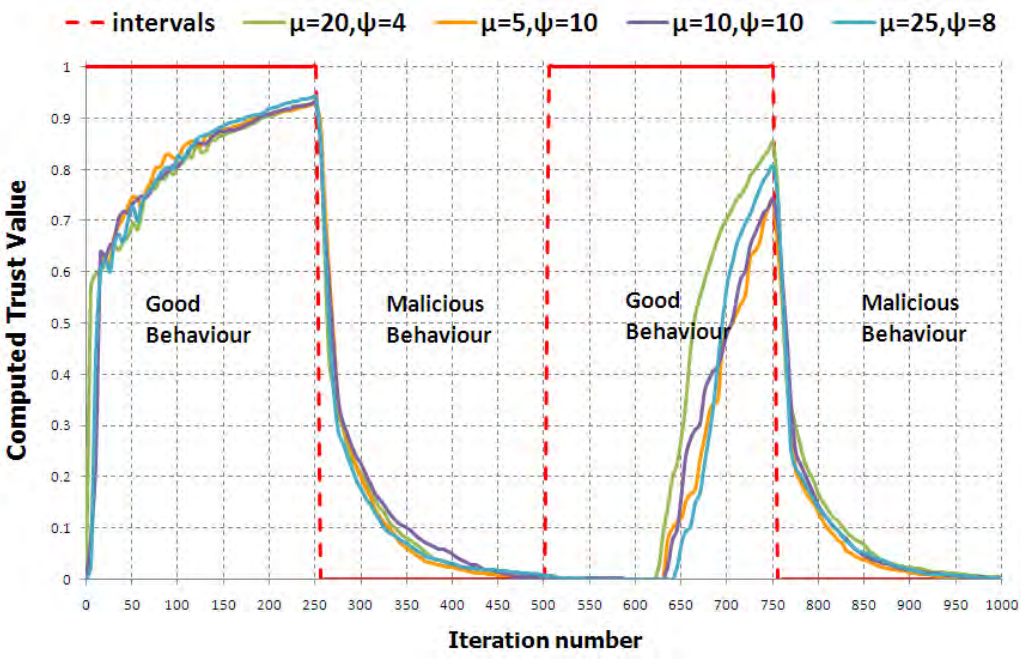


Figure 4.9: Effectiveness of SECTrust against dynamic personality.

the average STR. Since the responders to a transaction request is generated at random we take the mean of 30 experiments for each scenario. We compare our model with SFTrust [39], FCTrust [12], P2P recommendation trust model (for short we will use Recommendation-Trust/Reco-Trust) [31], trust model of users' behavior (for short we will use User-Trust) [34] and dynamic trust model for multi-agent systems (for short we will use MAS-Trust) [20].

First, we calculate STR against the variation of percentage of malicious agents *mal* while keeping *mres*=100% and *cm_rate*=0%. As from Fig. 4.10 we see that SECTrust shows superiority over the remaining trust models as the amount of malicious agents in the network increase beyond 50%. Networks today are home to a significant number of malicious agents, especially the internet hold great threats as it teems with malicious agents. So, in such networks SECTrust would provide the best performance.

In the next experiment we want to observe the impact of collusion on STR. So, for this experiment we set *mal* to 60% because as the number of malicious agents increases their collusive impact becomes greater and we also set *mres* to 100%. Fig. 4.11 represents the computed STR against *cm_rate*. Due to the experimental randomness, the gradient of the curves may vary from experiment to experiment. So, for better interpretation we draw the linear trend (Fig. 4.12) of these curves and then analysis the gradient of the trend lines. Now, from mathematical point of view the more positive the gradient the better since positive gradient means increase in STR which is desirable. In Fig. 4.13 we see that

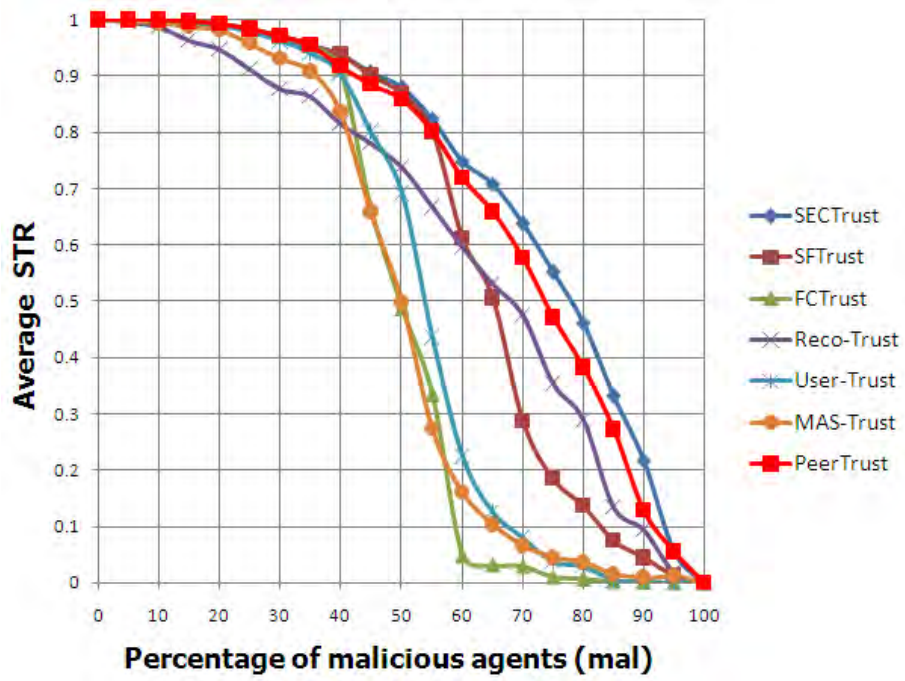


Figure 4.10: Comparing SECTrust with SFTrust [39], FCTrust [12], Reco-Trust [31], User-Trust [34], MAS-Trust [20] and PeerTrust [35] in terms of average STR against *mal*.

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 model remain unaffected by collusion. The main reason behind this is the credibility measure which filters out false feedbacks. Here false high ratings come from agents (namely malicious agents) with low credibility as a result they have no impact on STR.

In the third experiment we will analysis the impact of *mres* on STR. As we saw in Fig. 4.7 that the malicious agents tend to fool other agents by oscillating between good and malicious nature. In this experiment we test three scenarios with *mal* set to 60%, 70% and 80% respectively while *cm_rate* is set to 0% in all the cases. Figs. 4.14-4.16 represents the computed STR against *mres*. From the figures we see that SECTrust out performs all the trust model significantly. This is because our model keeps track of sudden rise and fall of trust by agents and penalizes any agent showing frequent trust fluctuation. Thus, SECTrust can successfully resist strategically altering behavior by malicious agents.

In the next experiment we will test the stability of the trust models under oscillating behavior. Here we run the experiment for a total of 500 iterations with *mres* set to 50% and *cm_rate* set to 0%. However, we divide the 500 iterations into four equal slots and so each slot contains 125 iterations.

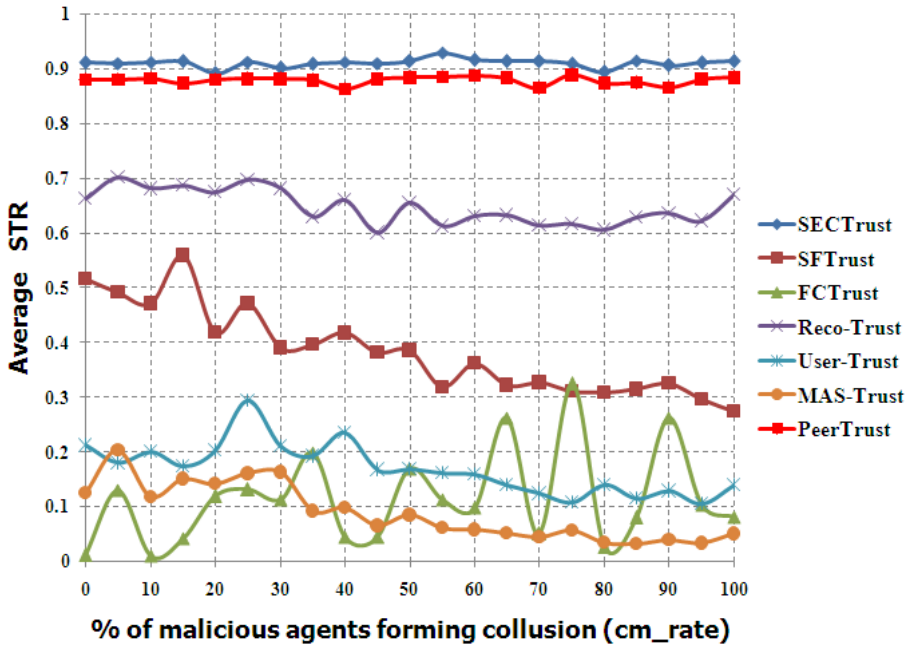


Figure 4.11: Comparing SECTrust with SFTrust [39], FCTrust [12], Reco-Trust [31], User-Trust [34], MAS-Trust [20] and PeerTrust [35] in terms of average STR against *cm_rate*.

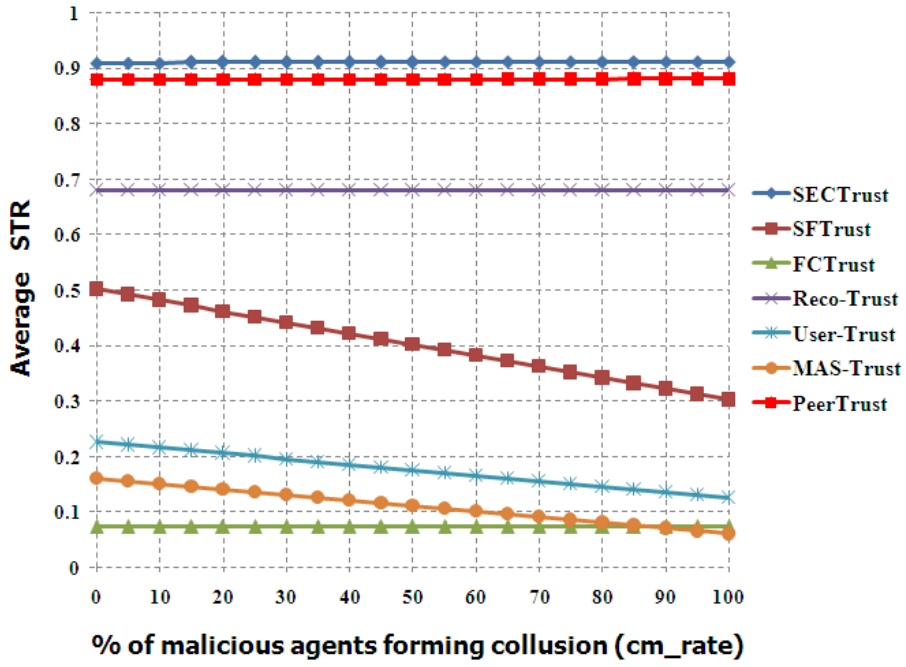


Figure 4.12: Comparing the trend lines of the curves for SECTrust, SFTrust [39], FCTrust [12], Reco-Trust [31], User-Trust [34], MAS-Trust [20] and PeerTrust [35].

Malicious agents oscillate between benevolent and malicious nature from one slot to the next starting with benevolent nature. Then we compute the number of times malicious agents are selected as

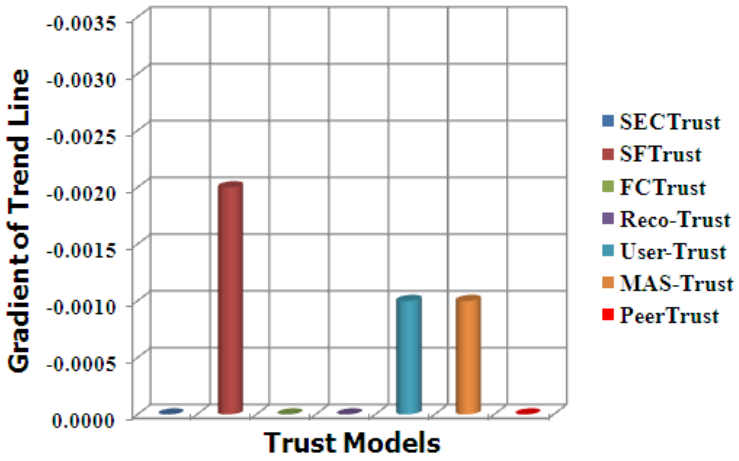


Figure 4.13: Comparing the gradient of the trend lines for SECTrust, SFTrust [39], FCTrust [12], Reco-Trust [31], User-Trust [34], MAS-Trust [20] and PeerTrust [35].

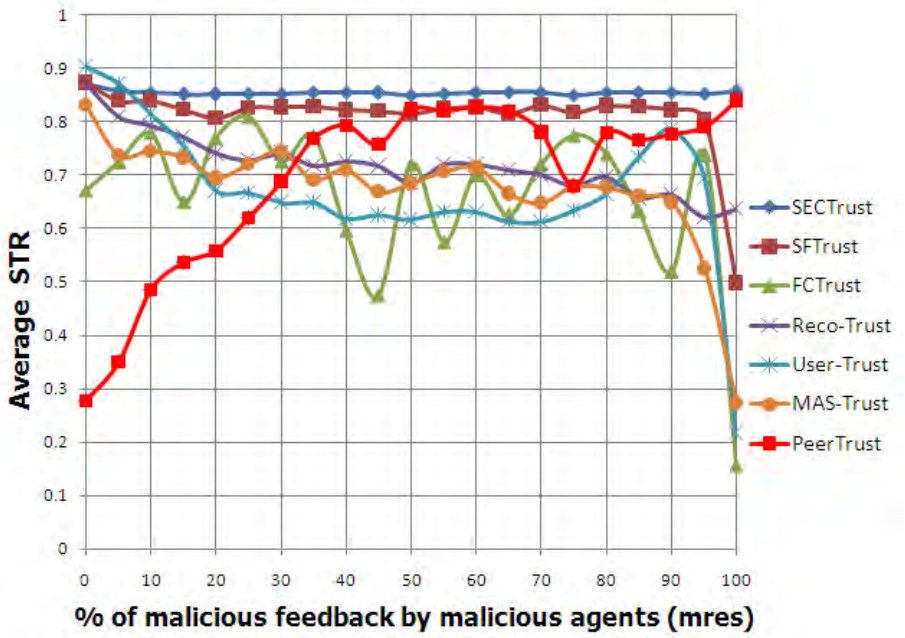


Figure 4.14: Comparing SECTrust with SFTrust [39], FCTrust [12], Reco-Trust [31], User-Trust [34], MAS-Trust [20] and PeerTrust [35] in terms of average STR against *mres* in presence of 60% malicious agents.

service providers to transactions initiated by only good agents. From Fig. 4.17 we see that in the initial slot malicious agents are selected numerous times. This is because in the first slot they start off by behaving good so there is no reason to reject them. However, in the following slots this number should decline as we now know their true nature. We see that our model performs best in isolating the malicious agents from providing service compared to the other trust models.

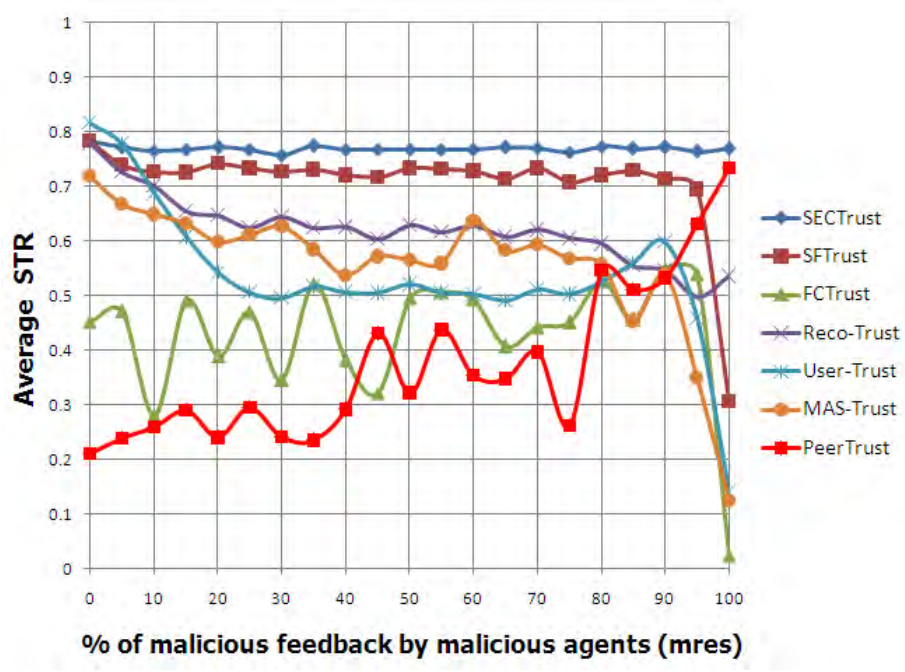


Figure 4.15: Comparing SECTrust with SFTrust [39], FCTrust [12], Reco-Trust [31], User-Trust [34], MAS-Trust [20] and PeerTrust [35] in terms of average STR against *mres* in presence of 70% malicious agents.

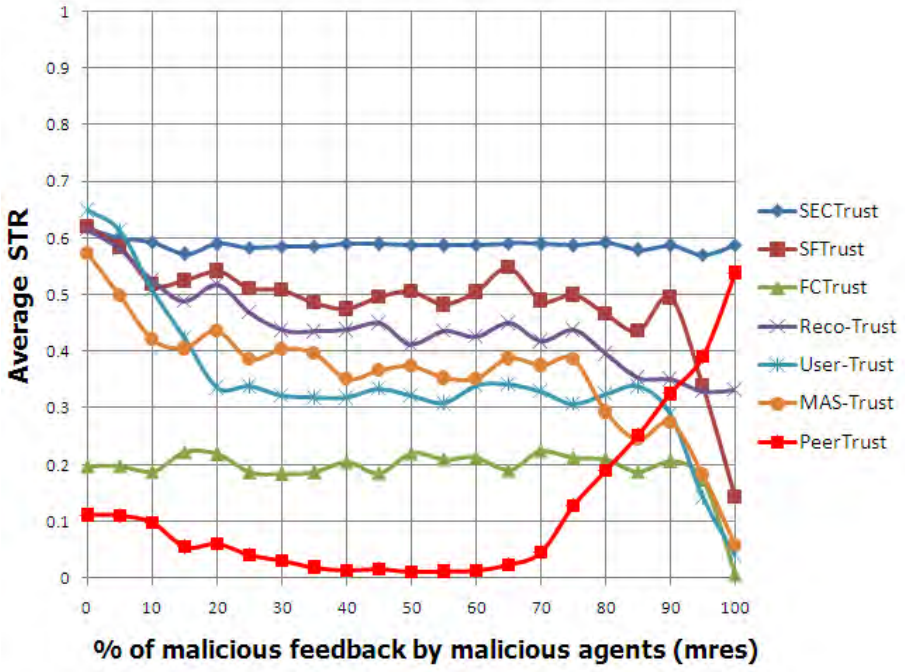


Figure 4.16: Comparing SECTrust with SFTrust [39], FCTrust [12], Reco-Trust [31], User-Trust [34], MAS-Trust [20] and PeerTrust [35] in terms of average STR against *mres* in presence of 80% malicious agents.

Finally we compare the time it takes for the different trust models to compute trust. For this purpose we compute the amount of times it takes for the trust models to execute 200 iterations with

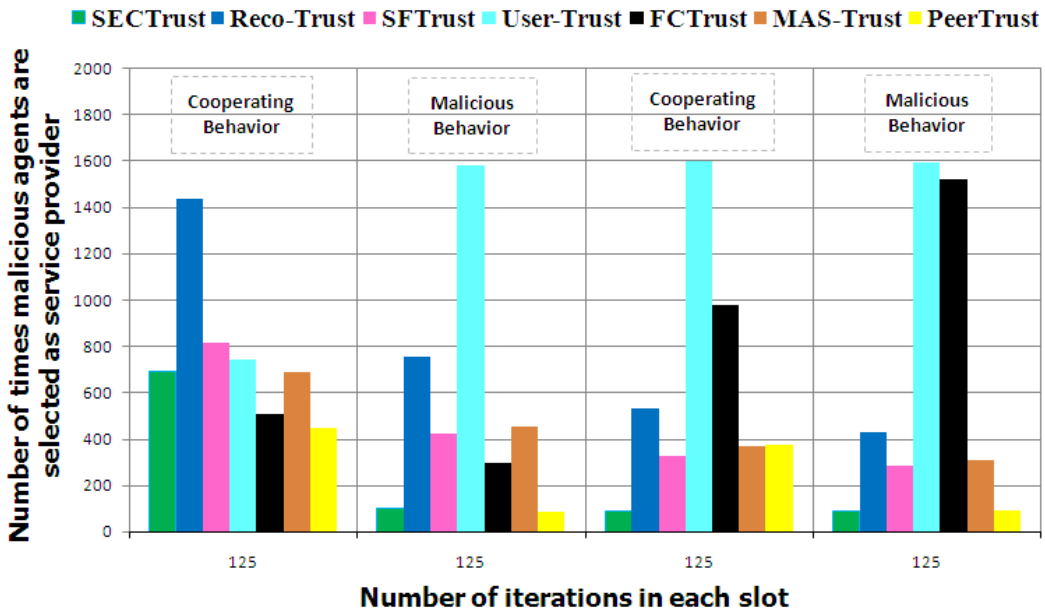


Figure 4.17: Comparing SECTrust with SFTrust [39], FCTrust [12], Reco-Trust [31], User-Trust [34], MAS-Trust [20] and PeerTrust [35] in terms of the number of times malicious agents are selected as service providers.

mal set to 50%, *cm_rate* set to 0% and *mres* set to 100%. We take the average of 30 runs. From Fig. 4.18 we see that MAS-Trust requires the largest amount of time while FCTrust requires the lowest amount. Our trust model requires on average 1.2 seconds to execute 200 iterations which higher than some the remaining trust model. This is understandable as we have considered a large number of components/parameters in our trust model as compared to the other trust models. For example we have considered sudden rise and fall of trust and we have also taken into account the historical trend of agent behavior which have not been considered by SFTrust, FCTrust, Reco-Trust and User-Trust. As a result these trust models fail to effectively filter out malicious agents when they start to show oscillating behaviors. The model that has considered the closest number of components as we have done is MAS-Trust [20] and as we can see from Fig. 4.18 that it requires on average 4.8 seconds to execute 200 iterations which is 4 times what our model requires. The main reason behind this is that we have used exponential averaging function to keep track of any past records which reduces storage overhead and as a result reduces the time required to retrieve the stored values. So from this point of view our trust model greatly reduces the computational time.

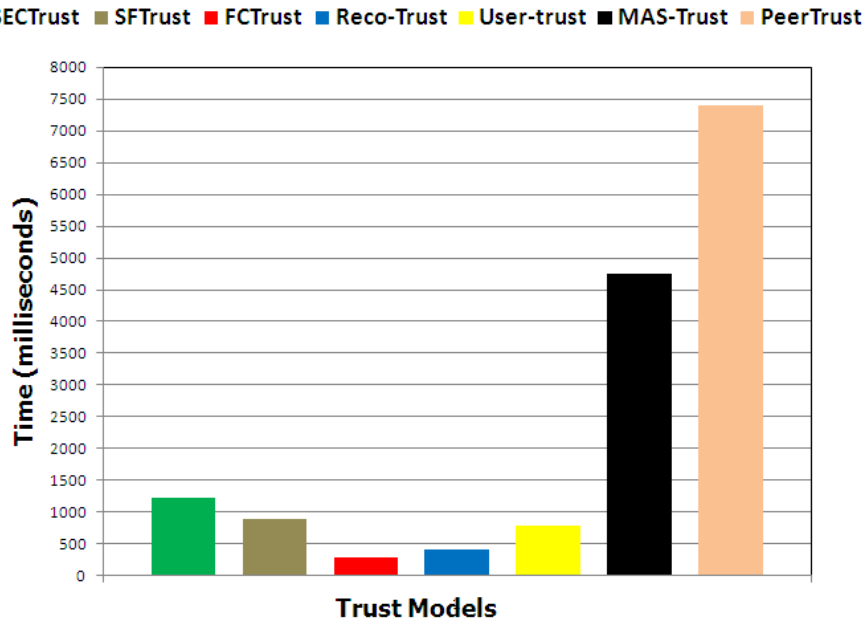


Figure 4.18: Comparing SECTrust with SFTrust [39], FCTrust [12], Reco-Trust [31], User-Trust [34], MAS-Trust [20] and PeerTrust [35] in terms of the amount of time required to execute 200 iterations.

Table 4.2: Comparing the min, max, mean and standard deviation obtained for SECTrust, SFTrust [39], FC-Trust [12], Reco-Trust [31], User-Trust [34], MAS-Trust [20] and PeerTrust [35].

	SECTrust	SFTrust	FCTrust	Reco-Trust	User-Trust	MAS-Trust
Min	0.9468004	0.8621997	0.4800013	0.7888001	0.72919984	0.7911998
Max	0.96550004	0.8984005	0.5096002	0.81939965	0.758999984	0.82559965
Mean	0.956308482	0.883627887	0.492879247	0.805887873	0.746014665	0.807535056
SD	0.006021794	0.008306864	0.007960995	0.009064419	0.007355806	0.008239084

4.7 Statistical Analysis

This section performs some statistical analysis on the trust value computed by our trust model. Since we are considering random transactions in computing trust values, the purpose of this section is to check to what extent our trust model is statistically sound. For this purpose we computed min, max, mean and standard deviation (SD) of the trust values obtained from the 30 runs of the experiment. The following table summarizes the obtained values for the individual trust models.

The following figure highlight graphically the standard deviation obtained for the different trust models. As we can see from Fig. 4.19 the computed standard deviation is very small for all the trust models which signifies that the trust values obtained are more or less uniform.

We also performed chi-square test (χ^2-test). For this experiment we set *mal* to 40%, *mres* to

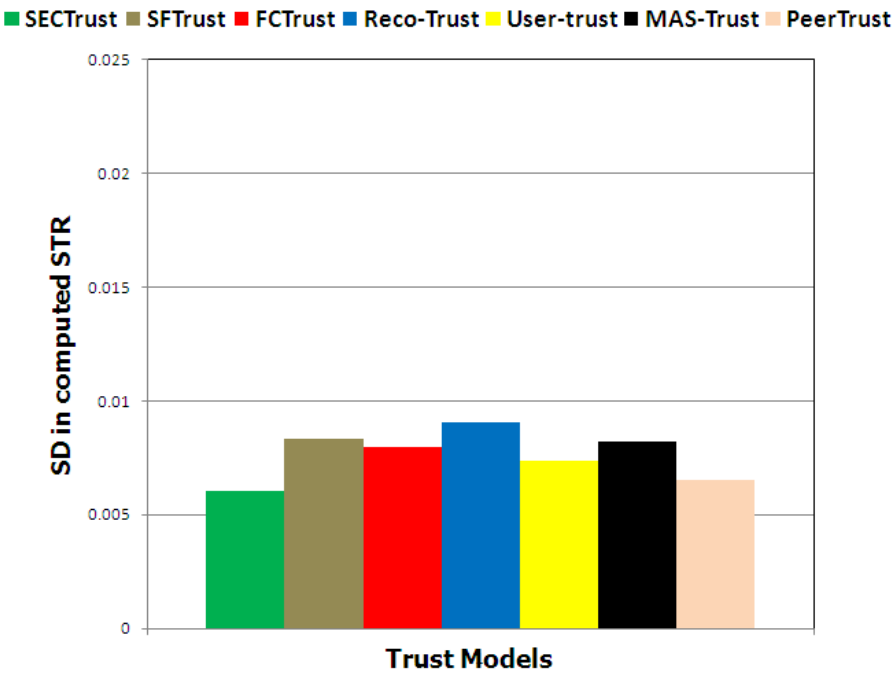


Figure 4.19: Comparing SECTrust with SFTrust [39], FCTrust [12], Reco-Trust [31], User-Trust [34], MAS-Trust [20] and PeerTrust [35] in terms of computed standard deviation.

100% and *cm_rate* to 0%, and then computed the trust value of all the agents in the system from a cooperative/good agent's point of view after 500 iterations. In each interaction each agent in the system performs one transaction with a randomly chosen agent. After computing the trust values of all the agents we categorize the agents based on the computed trust value into two classes namely: good and malicious. Finally, we compare the obtained counts with the expected count (which is 59 good agents and 40 malicious agents as *mal* is set to 40%) to compute the χ^2 value. For simplicity we perform 20 runs. The observed counts of cooperative and malicious agents is given in the Table 4.3.

Now we calculate the χ^2 value according to the following equation-

$$\chi^2 = \sum \frac{(O-E)^2}{E} \tag{4.1}$$

where O represents the observed value and E represents the expected value. So, after calculation we obtain $\chi^2 = 2.769$ and the degree of freedom is $(20-1) \times (2-1) = 19$. By consulting the Chi-Square Probability table we find that a χ^2 of 2.769 for 19 degree of freedom is smaller than the value of 30.144 (researchers commonly use a significance level of $p = 0.05$) listed. This signifies that the deviation that is found in the observed values and the expected values are likely caused by chance. We can interpret the result in another way, for a χ^2 value of 2.769 with 19 degree of freedom the chi-square probability is greater than 0.99 i.e., 99% of the time the observed values agree with the

Table 4.3: Observed count of good agents and malicious agents.

Run	Good Agents	Malicious Agents	Run	Good Agents	Malicious Agents
1	54	45	11	57	42
2	58	41	12	57	42
3	57	42	13	58	41
4	59	40	14	58	41
5	57	42	15	57	42
6	58	41	16	58	41
7	58	41	17	58	41
8	57	42	18	58	41
9	56	43	19	59	40
10	59	40	20	59	40

expected value.

4.8 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 STR against the number of agents in the network. We set *mal* to 40%, *mres* to 100% and *cm_rate* to 0%. Fig. 4.20 shows that the computed 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.

4.9 Decay of Trust with Lapse of Time

This experiment emphasizes the importance of attenuation of trust with lapse of time in absence of interaction. Now a days agents show highly dynamic personality as result trust values should not remain static in absence of interaction. By including a time decay model we are establishing the principle "*the more recent the transaction the more reliable it is*". Fig. 4.21 shows the comparison among no decay and the attenuation function defined in [34] with our decay model. From the figure it is observable that the attenuation function defined in [34], initially has a higher degradation rate than later on. However, it should be the reverse i.e., initially the degradation rate should be smaller and as more and more time elapse the degradation rate should increase or we can also have an uniform decay

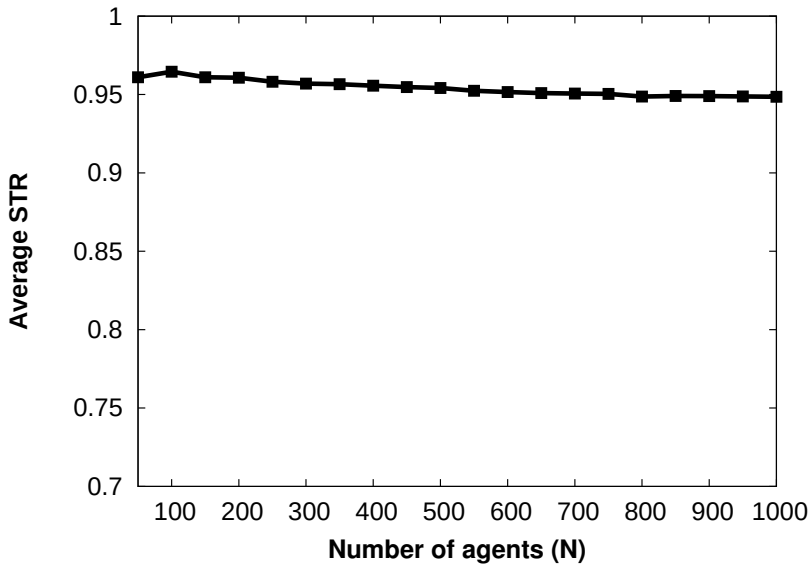


Figure 4.20: Scalability of SECTrust.

rate which our decay function incorporates.

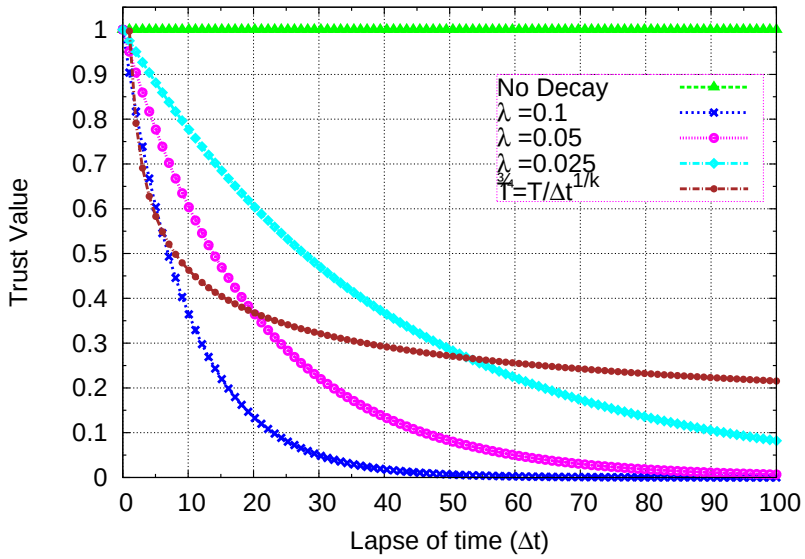


Figure 4.21: Decay of trust value with lapse of time.

4.10 Summary

In this chapter we have evaluated the performance of SECTrust under different scenarios. We have carried out experiments to tune the different parameters of our trust model to attain the best possible result. Experiments revealed that SECTrust can quickly adapt to the strategic adaptations made by

malicious agents including collusion and dynamic personality. From the comparative study with other existing trust models SECTrust proved to be more robust and effective against attacks from malicious agents.

Chapter 5

Conclusion

In this last chapter, we draw the conclusion of our thesis by briefly describing the major contributions made by the research work associated with this thesis followed by some directions for future research.

5.1 Major Contributions

The contributions that have been made in this thesis can be enumerated as follows:

- Our trust model SECTrust provides a reputation based trust mechanism to resolve some of the threats and risks present in a multi-agent environment by enabling agents to choose reputed agents while avoiding untrustworthy agents. For example, the simplest form of malicious attack is to provide some sort of corrupted service (i.e., virus attack, DOS attack etc.) in response to a transaction. However, with the presence of a reputation-based trust mechanism the agent who receives malicious service will be able to provide negative feedback to other agents about the service provided by a malicious agent and thereby, help others to avoid the malicious agent in future.

- We have also addressed one of the common vulnerabilities associated with reputation-based trust mechanism such as shilling attacks where adversaries attack the system by submitting false ratings to confuse the system. Shilling attack is often associated by collusion attack where malicious agents group up together and collaborate to raise each other's rating by making fake transactions. SECTrust prevents such threats by assigning credibility to each feedback provider. By doing so, SECTrust discards feedbacks submitted by malicious agents and thereby nullify collusive attack.

- Our trust model handles one of the challenging threats that most trust models fail to handle which is the strategic alteration of behavior by malicious agents. By cleverly alternating between good and malicious nature they try to remain undetected while causing damage. SEC-Trust keeps track of sudden rise and fall of trust and thereby can easily penalize such oscillating behavior.

- We have also address the attenuation of trust with elapse of time in absence of interaction which many of the previous models have not considered.

- We have greatly reduced storage overhead by using an exponential averaging function to keep record of the previous trends or trust values.

5.2 Future Work

Any research on any topic always makes ways for future research works and ours is not an exception. Trust models need to adapt to new challenges and requirements that arise as new malicious activities evolve. Some of the issues that we found requires further investigation are give below-

- As we have previously stated we have addressed only a handful of the threats that exist in MAS. Other threats like 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. This can be looked into for further analysis.

- Some of the most complex malicious attacks like the sybil attack [37] (similar to Whitewashing Attack) where an agent creates multiple identities and switches from one to another with the intent of removing its bad history severely affect the trust model. Friedman and Resnick [10] 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 new identity becomes unprofitable. So, some kind of metric that takes into account the cost of entry into and exit from a community could be incorporated into the trust model.

- In case of choosing agents based on highest trust value there is always the probability that some of the agents (with high reputation) will have heavy load in providing services while other capable agents (with comparatively low reputation) will have relatively low work load even

though they could also provide the required services. So, a load balancing scheme based on the trust metric could be developed to provide a more efficient utilization of resource.

- Our trust model can be easily formulated for use in Wireless Sensor Networks (WSN). The main advantage with our approach is that the storage requirement for the different components is minimal as we have used exponential averaging functions to define them. This is critically important for sensors in WSN as they have limited space and computational power. Some other parameters like geographical distance of service providers should be taken into consideration because long distance transmissions require substantial amount of energy and sensors try to avoid this situation as they are always attempting to conserve energy.

Bibliography

[1] Gnutella, http://www.gnutella.com, last visited: 6 may, 2010.

[2] Jbuilder, http://www.embarcadero.com/products/jbuilder, last visited: 15 february, 2010.

[3] Multi-agent systems, http://www.cs.cmu.edu/ softagents/multi.html, last visited: 6 may, 2010.

[4] R. Andersen, C. Borgs, J. Chayes, U. Feige, A. Flaxman, A. Kalai, V. Mirrokni, and M. Tennenholtz, "Trust-based recommendation systems: An axiomatic approach," *Proceeding of the 17th ACM international conference on World Wide Web (WWW)*, pp. 199–208, 2008.

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

[6] E. Carrara and G. Hogben, "Reputation-based system: A security analysis," *European Network and Information Security Agency (ENISA)*, pp. 1–32, October 2007.

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

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

[9] 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.

[10] E. J. Friedman and P. Resnick, "The social cost of cheap pseudonyms," *Journal of Economics and Management Strategy*, vol. 10, pp. 173–199, 1998.

[11] D. Heckerman, "A tutorial on learning with bayesian networks," *Learning in graphical models*, pp. 301–354, 1999.

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

[13] 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.

[14] T. D. Huynh, N. R. Shadbolt, and N. R. Jennings, "Developing an integrated trust and reputation model for open multi-agent systems," *Proceedings of the 7th International Workshop on Trust in Agent Societies*, pp. 65–74, 2004.

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

[16] N. R. Jennings, T. D. Huynh, and N. R. Shadbolt, "FIRE: An integrated trust and reputation model for open multi-agent systems," *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI)*, pp. 18–22, 2004.

[17] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision Support Systems*, vol. 43, no. 2, pp. 618–644, 2007.

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

[19] V. R. Lesser, "Cooperative multiagent systems:A personal view of the state of the art," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 1, pp. 133–142, 1999.

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

[21] S. P. Marsh, "Formalising trust as a computational concept," Ph.D. dissertation, Department of Mathematics and Computer Science, University of Stirling, 1994.

[22] J. J. Qi and Z. Z. Li, "Managing trust for secure active networks," *Central and Eastern European Conference on Multi-Agent Systems(CEEMAS)*, pp. 628–631, 2005.

[23] 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.

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

[25] 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.

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

[27] D. Senn, "Reputation and trust management in ad hoc networks with misbehaving nodes," Diploma Thesis DA-2003.27, 2003. [Online]. Available: www.infsec.ethz.ch/people/dsenn/diploma_thesis.pdf

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

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

[30] K. Sycara, "Multi-agent systems," *AI Magazine*, vol. 19, no. 2, pp. 79–92, 1998.

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

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

[33] 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.

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

[35] 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.

[36] L. Xiong and L. Liu, "A reputation-based trust model for peer-to-peer ecommerce communities [extended abstract]," *Proceedings of the 4th ACM conference on Electronic commerce(EC)*, pp. 228–229, 2003.

[37] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "Sybilguard: Defending against sybil attacks via social networks," *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications (ACM SIGCOMM)*, pp. 267–278, 2006.

[38] Y. Zhang, K. Wang, K. Li, W. Qu, and Y. Xiang, "A time-decay based P2P trust model," *Proceedings of the 2009 International Conference on Networks Security, Wireless Communications and Trusted Computing*, vol. 2, pp. 235–238, 2009.

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

Appendix A

Algorithms Related to Building the Trust Model

Algorithm 2 update_satisfaction(int id,int feedback)

Input: $Target\ agent's\ id$ **id** $and\ recent\ satisfaction\ degree$ **feedback**

Output: $Update\ satisfaction\ factor\ for\ target\ agent$

$S_{cur} \leftarrow feedback$

$\delta[id] \leftarrow |S[id] - S_{cur}|$

$\xi[id] \leftarrow c * \delta[id] + (1 - c) * \xi[id]$

$\alpha \leftarrow threshold + c * \frac{\delta[id]}{1+\xi[id]}$

$S[id] \leftarrow \alpha * S_{cur} + (1 - \alpha) * S[id]$

Algorithm 3 calculate_DT(int id)

Input: $Target\ agent's\ id$ **id**

Output: $Calculate\ the\ direct\ trust\ of\ the\ target\ agent$

$DT[id] \leftarrow get\_Satifaction(id) * get\_Credibility(id)$

Algorithm 4 calculate_IT(int id, agent target)

Input: $Target\ agent's\ id$ **id** $and\ the\ target\ agent$ **target**

Output: $Calculate\ the\ indirect\ trust\ of\ the\ target\ agent$

$numerator \leftarrow 0$

$denummerator \leftarrow 0$

$W \Leftarrow compute\_recommender\_set\ (target)$ //$using\ algorithm\ 1$

for $each\ x \in W$ **do**

$numerator \leftarrow numerator + Cre(x) * get\_DT(x, id)$

$denummerator \leftarrow denummerator + Cre(x)$

end for

if $denummerator > 0$ **then**

$IT[id] \leftarrow numerator/denummerator$

end if

Algorithm 5 calculate_RT(int id, agent target)

Input: $Target\ agent's\ id$ **id** $and\ the\ target\ agent$ **target**

Output: $Calculate\ the\ recent\ trust\ of\ the\ target\ agent$

$numerator \leftarrow 0$

$W \Leftarrow compute\_recommender\_set\ (target)$ //$using\ algorithm\ 1$

for $each\ x \in W$ **do**

$numerator \leftarrow numerator + get\_InteractionCount(x, id)$

end for

if $|W| > 0$ **then**

$M \leftarrow numerator/|W|$

end if

$I \leftarrow get\_InteractionCount(own, id)$

if $I + M > 0$ **then**

$\beta \leftarrow \frac{I}{I+M}$

end if

$Prev\_RT[id] \leftarrow RT[id]$

$RT[id] \leftarrow \beta * get\_DT(id) + (1 - \beta) * get\_IT(id)$

Algorithm 6 calculate_HT(int id)

Input: $Target\ agent's\ id$ **id**

Output: $Calculate\ the\ historical\ trust\ of\ the\ target\ agent$

$HT[id] \leftarrow \frac{\rho * HT[id] + get\_PreviousRT(id)}{2}$

Algorithm 7 calculate_ET(int id)

Input: $Target\ agent's\ id$ **id**

Output: $Calculate\ the\ expected\ trust\ of\ the\ target\ agent$

$I(id) \leftarrow get\_InteractionCount(own, id)$

if $I(id) = 0$ **then**

$ET[id] \leftarrow 0$

else if $I(id) > 0$ && $get\_RT(id) > 0$ && $get\_HT(id) = 0$ **then**

$ET[id] \leftarrow get\_RT(id)$

else if $I(id) > 0$ && $get\_RT(id) = 0$ && $get\_HT(id) > 0$ **then**

$ET[id] \leftarrow get\_HT(id)$

else if $I(id) > 0$ && $get\_RT(id) > 0$ && $get\_HT(id) > 0$ **then**

$ET[id] \leftarrow \eta[id] * get\_RT(id) + (1 - \eta[id]) * get\_HT(id)$

end if

if $get\_RT(id) - get\_HT(id) > \varepsilon$ && $\eta[id] \neq 1$ **then**

$\eta[id] \leftarrow \eta[id] + 0.1$

else

$\eta[id] \leftarrow \eta[id] - 0.1$

end if

Algorithm 8 calculate_Similarity(int id, agent target)

Input: $Target\ agent's\ id$ **id** $and\ the\ target\ agent$ **target**

Output: $Calculate\ the\ similarity\ of\ the\ target\ agent$

$temp1 \leftarrow 0$

$temp2 \leftarrow 0$

$sat\_diff \leftarrow 0$

$W \Leftarrow PS(own) \bigcap PS(target)$

if $|W| > 0$ **then**

  for $each\ x \in W$ **do**

    $temp1 \leftarrow get\_satisfaction(own, id)$

    $temp2 \leftarrow get\_satisfaction(target, id)$

    $sat\_diff \leftarrow sat\_diff + (temp1 - temp2)^2$

  end for

  $difference[id] \leftarrow \sqrt{sat\_diff/|W|}$

end if

if $difference[id] < \tau$ **then**

  $sim[id] \leftarrow sim[id] + \frac{1-sim[id]}{\mu}$

else

  $sim[id] \leftarrow sim[id] - \frac{sim[id]}{\psi}$

end if

Algorithm 9 calculate_Credibility(int id)

Input: $Target\ agent's\ id$ **id**

Output: $Calculate\ the\ credibility\ of\ the\ target\ agent$

if $get\_similarity(id) < 0.01$ **then**

  $Cre[id] \leftarrow 0$

else

  $Cre[id] \leftarrow 1 - \frac{\ln(get\_similarity(id))}{\ln \theta}$

end if

Algorithm 10 calculate_RRT(int id, agent target)

Input: $Target\ agent's\ id$ **id** $and\ the\ target\ agent$ **target**

Output: $Calculate\ the\ reliability\ in\ recent\ trust\ of\ the\ target\ agent$

$N[id] \leftarrow get\_InteractionCount(own, id)$

$W \Leftarrow compute\_recommender\_set\ (target)$ //$using\ algorithm\ 1$

for $each\ x \in W$ **do**

$N[id] \leftarrow N[id] + get\_Credibility(id) * get\_InteractionCount(x, id)$

end for

if $N[id] < maxN$ **then**

$Prev\_RRT[id] \leftarrow RRT[id]$

$RRT[id] \leftarrow \sin\left(\frac{\Pi * N[id]}{2 * maxN}\right)$

else

$Prev\_RRT[id] \leftarrow RRT[id]$

$RRT[id] \leftarrow 1$

end if

Algorithm 11 calculate_RHT(int id)

Input: $Target\ agent's\ id$ **id**

Output: $Calculate\ the\ reliability\ in\ historical\ trust\ of\ the\ target\ agent$

$RHT[id] \leftarrow \frac{\rho * RHT[id] + get\_PreviousRRT(id)}{2}$

Algorithm 12 calculate_AT(int id)

Input: $Target\ agent's\ id$ **id**

Output: $Calculate\ the\ accumulated\ misused\ trust\ of\ the\ target\ agent$

if $get\_RT(id) - get\_HT(id) > \varphi$ **then**

$AT[id] \leftarrow AT[id] + \frac{get\_RT(id) - get\_HT(id)}{\omega}$

else if $get\_RT(id) - get\_HT(id) < -\varphi$ **then**

$AT[id] \leftarrow AT[id] + get\_HT(id) - get\_RT(id)$

else

$AT[id] \leftarrow AT[id]$

end if

Algorithm 13 calculate_DR(int id)

Input: $Target\ agent's\ id$ **id**

Output: $Calculate\ the\ deviation\ reliability\ of\ the\ target\ agent$

if $get\_AT(id) < maxAT$ **then**

$DR[id] \leftarrow \cos\left(\frac{\Pi * get\_AT(id)}{2 * maxAT}\right)$

else

$DR[id] \leftarrow 0$

end if

Algorithm 14 calculate_confidence(int id)

Input: $Target\ agent's\ id$ **id**

Output: $Calculate\ the\ confidence\ factor\ of\ the\ target\ agent$

if $(get\_RRT(id) + get\_RHT(id)) * get\_DR(id) > 1$ **then**

$C[id] \leftarrow 1$

else

$C[id] \leftarrow (get\_RRT(id) + get\_RHT(id)) * get\_DR(id)$

end if

Algorithm 15 update_recommnder_credibility(int id, agent target)

Input: $Target\ agent's\ id$ **id** $and\ the\ target\ agent$ **target**

Output: $Update\ of\ the\ cerdibility\ of\ the\ recommenders$

$W \Leftarrow compute\_recommender\_set\ (target)$ //$using\ algorithm\ 1$

for $each\ x \in W$ **do**

$calculate\_Similarity(x's\ id, x)$

$calculate\_Credibility(x's\ id)$

end for