An Efficient Trust Model for Multi-Agent Systems

Akbar S. Namin[1] akbar.siami@nrc.gc.ca
Ruizhong Wei[2] wei@ccc.cs.lakeheadu.ca
Weiming Shen[1, 3] weiming.shen@nrc.gc.ca
Hamada Ghenniwa[3] hghenniwa@eng.uwo.ca

[1]National Research Council Canada – IMTI, London, Canada
[2]Department of Computer Science, Lakehead University, Canada
[3]Department of Electrical and Computer Engineering, University of Western Ontario, Canada

Abstract

Agent-based E-business opens up a computational direction, in which software agents behave on behalf of their owners. In fact, owners of the agents should rely on their employed agents in communicating and exchanging information. However, enabling agents to make decisions and exploiting information to other unknown agents have introduced some security challenges among them is trust relationship while agents are dealing with unknown environments and their residences. Despite existing of some sophisticated proposed approaches addressing the above problem, most of them are suffering from complexity and therefore drawback from development. This paper proposes an efficient trust model for distributed systems mainly focusing on multi-agent systems. We provide a feasible mechanism using some well-known cryptographic techniques such that not only it addresses the above issue but also guarantees the security and resistibility of the model against some attacks.

1 Introduction

Trust is as important in multi-agent systems as it is in human societies. The notion of an agent implies the concept of delegation and delegation is based on trust [2]. Face-to-face business relation in any cooperative environment and its result success depends heavily on trust issues between two parties. As a very primitive definition we can say trust is the efforts of an agent in estimating the truth behaviors of another agent. The risk level will be rising dramatically, while the level of trust drops off. Consequently, the level of trust depends directly on the amount of information about reputation of a partner. Therefore, lacking or having limited information about a partner and its trustworthiness relationship with others may decrease trust issue and consequently increases the level of risk. The situation will be critical if there is a decision making process in choosing the most reliable business partner among enormous number of candidates.

Trust is important not only in the case of needs to a service provided by an unknown entity, but also in choosing a service offered by some candidate service providers, which are offering similar services. As a matter of fact, any business entity to reduce the level of the risk is highly encouraged to have a business relationship and use services provided by those entities, which are most trusted.

Trust and reputation have significant effect on trading and so on online E-business. Some researchers consider trust and reputation are the same and can be used interchangeably. However, technically they are different. Generally, trust is a measurable unit showing the level or degree of the information that two parties have about the other one. This measurable unit varies in different proposed models. Some models consider as a continues variable varying in range of [-1, 1] or [0, 1]; whereas, some believe this value is a discrete value ranging in real number domain showing the

absolute maximum and minimum values as full trust and no trust respectively. To this end, the minimum trust or more strictly no trust occurs whenever there is no information about the party. As a result and in case of no trust, any entity, or agent, might rely on collected information from other agents named as reputation information. Therefore, an agent uses reputation information while there is a decision making on starting a relationship with an unknown agent.

Using systematic security mechanisms such as digital signature, authentication as well as authorization and non-repudiation, as they are, cannot address trust issue. As a matter of fact, an agent might be authenticated and authorized to accomplish an activity, however, there is no guarantee that the authorized agent behaves as it is expected. In the other words, we may able to authorize all unknown agents but we can trust none. Basic security techniques are essential but not enough and require equipping with a mechanism for addressing trust problem.

Some primitive and simple reputation mechanisms have been implemented in some application domain such as eBay and Amazon. In addition, several good research models on trust and reputation have been proposed as well. However, the security issues of trust have not been addressed very well. As an example, the reputation information sited on the eBay simply can be forged and the reliability of the information might be affected under several other attacks. As eBay, any seller may issue a number of good reputation credit for itself and falsely increases its faked reputation. On the opposite side, any competitor seller, might issue a number of bad complaining reputations for a seller and falsely decreases the reputation of an individual seller.

This paper proposes an efficient trust model based on some basic cryptographic techniques. We do not discuss about any model regarding reputation instead we try to provide an efficient and secure solution for trust management in distributed systems particularly multi-agent systems. We defined the notion of "*Trust Path*" dealing with discovering a path representing trust in a graph-based network of agents. We model a trust relationship using hash functions, encryption, and decryption as well as applying a simple combinatorial method for counting. The computationally secure property of hash functions implies a computationally secured trust model.

The rest of the paper is structured as following: In Section 2 we review some literature regarding reputation and trust issues in distributed systems in general. We present an efficient trust model for multi-agent systems in Section 3. We discuss some possible aspects of the model in Section 4. Section 5 discusses the developed prototype of the model and the related issues. Section 6 provides a conclusion and future contribution.

2 Related Work

Trust management is an interesting topic in different areas varying from economics, game theory and sociology to risk management and social psychology. Based on the application domain of trust notation enormous approaches have been proposed. In several research papers, trust and reputation are discussed together as related topics. Also, most models have been proposed in multi-agent systems, since software agents are the main target for the distributed trust. Regarding the context of this paper, we only focus on approaches based on multi-agents systems.

The most on hand cases of developing reputation systems are well-known online trading Web sites such as eBay and Amazon. Reputation mechanism in these systems is a cumulative rating measure for both parties in trading. However, these mechanisms have been known as very naive and are vulnerable under some primitive attacks. As an example, Resnick et al., [13] shows the Pollyanna effect of the e-trading reputation systems. This effect has been addressed as disproportionately positive and rare negative feedbacks. Also, Dellarocas [3] has shown some attacks on these systems as well.

According to Marsh [9] and Castelfranchi et al., [2] trust is a mental attitude and a trust relation between agents will be achievable by delegations. As another context, trust is the same as evaluating estimation for the trustworthiness of agents.

Marsh [9] is among the first who provides a trust model in distributed systems with main focus on software agents. Marsh modeled trust as a real number ranging from -1 to +1. However, as Marsh shows, the model has some difficulties in computing trust values at extreme values as well as zero ones. Moreover, the model has some difficulties in evaluating trust as a negative value. In fact, the algebraic notations and operations that have been provided are not strong enough to handle the negative values.

A generic model with dual emphasis for trust has bee proposed by Tab et al., [16]. The model emphasizes on trusting the active party responsible for fulfilling a transaction. Doney et al., [4] have considered some trust relationships categories that might be exist in any trading. Furthermore, their model has defined those classes as: Calculative, Prediction, Competence, Goodwill, and Transference.

Building a TrustNet based on adopting game theory and using probability theory together for updating beliefs about other agents has been proposed by Schillo [15]. Also, Mui et al., [11, 12] have proposed an interesting probabilistic model for reputation based on Bayesian network. Moreover, they have described a framework based on game theory for understanding the relative strength of the different notions of reputation.

Zacharia et al., [18] have proposed a framework by which agents participate in some communities and generate a temporal kind of reputation measure based on the performances and recommendations. Barber et al., [1] introduces a belief revision framework in order to encourage information resources to keep performance data and as a result avoiding the possible risk by realizing the bad reputation of the party.

Reputation is defined as "an opinion or view of one about something" by Sabater, et al. [14]. The reputation has been modeled as "individual", "social", and "ontological". Individual refers to impression of one's behavior to others. In social, the reputation of an individual will be influenced by the community of the individual which belongs to. Context-based reputation is defined as ontological one.

A trust model for agents' collaboration using recommender agents have been discussed by Montaner et al [10]. They have equipped the recommender agents by some techniques to look for similar agents in order to receive advices while there is little or no trust information.

Besides the "trust" terminology and its concept, the "distrust" terminology with its specified meaning has been discussed in several papers as well. Golbeck et al., [6] has proposed a graded trust with nine levels ranging from strictly no-trust to completely trust. The model can be considered as a simple model interpretable by human users.

Yu, et al., [17] have discussed the feasibility of applying Dempster Shafer evidence theory. Reputation is evaluated based on the rating propagation and considering the agent's neighbors. Also, the propagated values are weighted by the neighbors. In fact, they model trust by using recommender systems. In their model, agents collect information about other agents including their recommending potential and trustworthy partners. As a result, agents recommend the most reliable agents to users. An extension of Yu et al, model to cover the distrust issue has been discussed by [8] using logic based trust.

Using semantic Web to address the trust relationship has been discussed by Finin et al., [5]. The model is mainly focusing on transforming the Web from a universal database model to an active society of autonomous agents. The model has tried to adopt semantic Web and its related standards to provide trust among agents. As a result they have modeled the Trust-Builder architecture for negotiation trust between unknown parties with respect to their properties.

Content-triggered trust negotiation has been discussed by Hess et al., [7]. The model employs traditional access control and client-server architecture to authorize clients for using servers. A general-purpose access control model has been designed to detect the feasibility of disclosing sensitive information.

In this paper, we discuss an efficient trust model based on collecting of encrypted hash values via the defined notion named as "Trust Path". The model might be viewed as a combinatorial approach in counting a number of objects as well as adopting basic cryptographic techniques to handle a secure counting that resists against some attacks. In fact, attacking and breaking the model in terms of cryptography is computationally hard. This characteristic has been arisen from the one-way property of hash functions.

3 A Trust Model for MAS

This section proposes an efficient model for trustworthiness in multi-agent systems. We envision a network of agents in such a way that their collaboration and resources accessibility are evaluated and checked against their trust relationships. We may look at a multi-agent environment and trust relationships as a weighted directed graph with the agents as nodes and the trust relationships as weighted edges connecting them together. The graph might be either complete representing a trusted environment or not com-

plete showing an environment consisting of some both known and unknown agents. Therefore, in case of lacking trust, or no trust between two agents, there is no edge connecting them. Figure 1 depicts a simple network of agents and their trust relationships together.

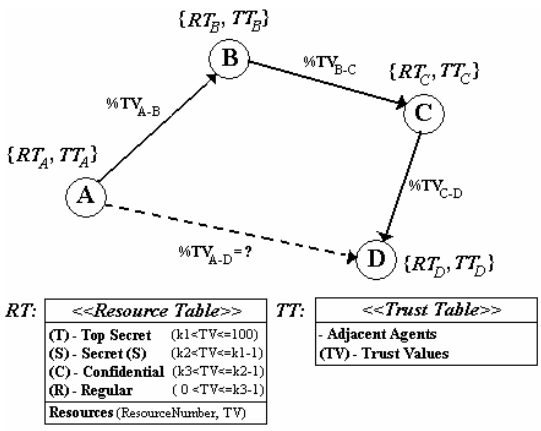

Figure 1 - A network of agents and their trust relationships

Basically, each agent possesses its own resources offering as services to other agents. Resources, regarding their cost and sensitiveness, are valuable properties of an agent, which have to be kept protected and offered to some special customers. Therefore, as a reasonable approach, we classify and label them into some security access levels namely as *Top Secret (T), Secret (S), Confidential (C)*, and *Regular (R)*. In fact, each resource is coupled with a security label to identify its access level to qualified agents. Also, for measuring purposes, we quantify the accessibility to each access level by using the "*minimum required trust values*". In other words, satisfying the minimum required trust value allows an unknown agent using the underlined resources.

In addition to the resource table, each agent constructs a "*Trust Table*" as well. Trust table consists of the identity of trusted agents and their assigned trust values. As we noted above, a trust relationship in multi-agent system looks like a weighted directed graph. If we suppose that there is an edge from node **A** to node **B**, then the trust table of agent **A** consists of identity of agent **B** including the weight value. This is important to note that there is no reason for agent **B** to trust agent **A** with the same trust value. In fact, despite having a trust relationship from agent **A** to agent **B**, to have a trust relationship from agent **B** to agent **A** there is a need to another weighted edge from node **B** to node **A**. More precisely, agent **A** may trust agent **B** with tv_1 and agent **B** based on its internal policy may trust agent **A** with tv_2 such that tv_1 and tv_2 can be either equal or different (i.e. $1 \le tv_i \le 100,$). In fact, In order to have a trust relationship from agent **B** to agent **A**, agent **A** has to submit a request to agent **B** and asks for trust relation. By above assumption, we are ready to describe the model. We divide the model into some stages and discuss each of them in detail.

3.1 Call for Trust

Suppose in Figure 1, agent **A** receives a request from an "*unknown*" agent **D** asking to start a business relationship and using agent **A**'s resources. Hence, agent **D** sends out a message indicating "*CallforTrust*" to agent **A**. Since agent **D** is unknown to agent **A**, therefore, there is no entry and consequently no trust value for agent **D** at agent **A**'s trust table.

3.2 Broadcasting and Trust Path Discovery

By receiving a request, agent **A** searches its own trust table and checks whether there exists any entry and consequently any trust value initiated for the agent **D**. Lacking of an entry represents that agent **D** is indeed an unknown agent. Therefore, agent **A** broadcasts an inform message to all its neighborhoods, which have an entry at agent **A**'s trust table and asks whether any agent has some trust relationship with unknown agent **D**. Since agent **A** needs to discover a trust path to the unknown agent, therefore a possible feedback to the broadcasted message might be the returned value of a function with list of agents located in the path from agent **A** to the unknown agent, as the arguments of the function:

TrustPath[ ] TrustPath(AgentID unknownagent, InitialPath[ ]);

Agent **A** initiates the arguments of the function as the unknown agent's ID as well as its own agent ID as the first element of the *InitialPath [ ]* respectively. Each agent by receiving the message repeats the same routine and looks for its own trust table. In any case, either lacking of any

entry or being an entry, the agent adds its own ID to the list *InitialPath [ ]* and return it to the next agent as *TrustPath* [ ]. However, in case of having a trust relationship with unknown agent **D**, the underlined agent sends back the constructed *TrustPath* [ ] to the sender. Accordingly, the original sender, or agent A, will receive the *TrustPath [ ]* as the trust path to the unknown agent **D**. On the other case, in which there is no entry for the unknown agent **D** at the trust table of any intermediate agent, the intermediate agent forwards the received message cumulated by its ID to the neighbor and trusted agents, which have entries at intermediate agent's trust table.

The above processes continue until agent **A** receives a *TrustPath [ ]* as trust path to the unknown agent **D**. However, if agent **A** does not receive any feedback list, she assumes *"no"* agent knows the unknown agent **D**. In the other words, no agent trusts and has a business relationship with the unknown agent. As a result, agents **A**, with respect to its own internal policy, may initiate a basic and minimum trust value and creates an entry at her trust table for the unknown agent **D** and starts a basic and primitive business relationship with it. This methodology, as starting a basic relationship with unknown traders, seems to be an essential policy in order to find out and attract hidden markets in today business.

There is a possibility that agent **A** discovers more than one trust path to the unknown agent. In this case, agent **A** might follow up different strategy in order to make a secure decision resistance against some possible attacks. We discuss about this case in the following next sections.

This is important to note that, the model is concerned about the privacy of trust relationships among agents. Therefore, agent **A** does not have any information about the trust "*values*" of each intermediate agent sited at the trust path. It can be considered as one of the advantages of using the model, since no one would like to exploit its business and relationships information to others. As a result, the privacy and the trust relationships among agents are protected.

3.3 Generating Hash Values

Agent **A**, by discovering the trust path, generates some hash values. For keeping the generality, we suppose that the total numbers of generated hash values are 100 items (representing percentages). Then agent **A** finds out the public key value of the unknown agent. Basically, the values of public keys are published and registered through an authorized entity such as central authority. However, in the absence of an authorized entity, each agent might ask the unknown agent to deliver its public key value directly by applying a secure key exchange algorithm such as Diffie-Hellman. By acquiring the unknown agent's public key, agent **A** encrypts the generated hash values with the unknown agent's public key. Encryption is essential to prevent other agents sited at the trust path of attacking the system. The hash function can be any well-known hash algorithm such as "MD5" or "SHA" as well as the encryption algorithm can be any well-known encryption algorithm such as "RSA" or Al-Gammal.

3.4 Injecting Some Encrypted Hash Values into the Trust Path

As the next stage, agent **A** locates the first agent at the trust path and find outs its trust value from its trust table. (Remember that the first agent at the trust path, indeed, is in the agent **A**'s trust table). As noted above, the trust values kept at the trust table represents percentages of trust to each corresponding agent. Hence, the trust values range from 1 as "*little*" trust to 100 as "*full*" trust. As an example, suppose the first node at trust path has been occupied by agent **B**. The trust value of agent **B** is kept in trust table of agent **A** as a percentages value such as $tv_{A\to B} \in Z^+$ such that $1 \le tv_{A\to B} \le 100 \in Z^+$, the set of positive integers). Consequently, agent **A** injects $tv_{A\to B}$ numbers out of 100 items of the generated and encrypted hash values into the trust path and delivers them to agent **B**. In addition to sending the hash values, agent **A** delivers the trust path to agent **B** in order to inform the identity of the next agent, which is indeed sited at the agent **B**'s trust table.

3.5 Forwarding the Hash Values through the Trust Path

All intermediate agents, including agent **B** in our scenario, follow up the same routine. More precisely, each intermediate agent by receiving hash vales and trust path, identify the next agent's ID at

the trust path and extracts its trust value from its own trust table. Let us describe the processes in terms of our simple scenario depicted as Figure 1.

As it is cleared from figure, the path list has been constructed as: *TrustPath [ ]* = {**A**, **B**, **C**, **D**} and their trust relationships are one way from source to destination agents as shown in Table 1.

Source Agent ID	Destination Agent ID	Trust Value
A	**B**	$\%tv_{A\to B}$
B	**C**	$\%tv_{B\to C}$
C	**D**	$\%tv_{C\to D}$

Table 1 - The intermediate agents and their trust values

Since the trust relationship from agent **A** to agent **B** is $\%tv_{A\to B}$, therefore, agent **A** sends $h_1 = tv_{A\to B}$ numbers out of 100 hash values to agent **B**. Consequently, agent **B**, by receiving $tv_{A\to B}$ numbers of encrypted hash values as well as the trust path, finds out the next node at the trust path and extracts its trust value from the trust table, which is agent **B**'s trust table. As in Figure 1 and Table 1 have been depicted, the trust relationship from agent **B** to agent **C** is $tv_{B\to C}$. Therefore, agent **B** forwards $h_2 = \lfloor (tv_{B\to C} * h_1)/100 \rfloor$ numbers out of h_1 numbers of hash values to agent **C**. Repeatedly, agent **C** with respect to its trust relationship with **D**, which is $tv_{C\to D}$, hands over $h_3 = \lfloor (tv_{C\to D} * h_2)/100 \rfloor$ numbers out of h_2 to agent **D** as trusted agent for **C** and as unknown one for agent **A**.

3.6 Returning the Decrypted Hash Values

The unknown agent, as in our scenario agent **D**, by receiving and collecting h_3 numbers of hash values, decrypts them using its own secret key and sends all h_3 numbers of hash values to the originator agent **A**. In fact the number of h_3 shows the credential and the trust value that the unknown agent is able to acquire from the network and specifically from the trust path. There is no way for agent **D** to inflate, forge, or attack these hash values since the nature of one way hash functions has been known as NP-hard which means it is computationally hard to find two different messages with equal hash values.

3.7 Verifying and Issuing Trust Value

Agent **A**, by receiving h_3 numbers of decrypted hash values from the unknown agent **D**, verifies them by regenerating the hash values and comparing the original hash values with the received ones. The number of verified hash values will be the trust value for the unknown agent **D**. In the other words, suppose that v number of h_3 numbers of hash values have been verified successfully. Therefore, if $v = h_3$ then it means all received values are valid. However, if $v < h_3$ then only v number of hash values are valid and can be treated as trust value for the unknown agent **D**.

4 Discussion

The main significant advantage of the model is protecting the privacy of agents. Agents keep their privacy as well as the values of their trust relationships with other agents. Discovering a trust path does not mean exposing the trust values of sited agents on it. In fact, by discovering a trust path, agents represent that they have "*some*" relationships with the neighborhood agents. Nevertheless, they do not expose the precise values of their trust. As a result, the privacy of their business will be kept secretly. However, the model will be vulnerable while the number of agents in trust path is equal to three (with calling and called agents exclusively). Hence, the discovering process of the trust path should be avoided whenever there is only one agent as intermediate one, which has some relationships with the unknown agent. But in fact, if the privacy of that intermediate agent is not important issue, still model is able to provide an answer. In following we explain different cases, problems, and accordingly possible attacks to the model. As a result, with respect to each case or problem, we may have different approaches. Since the trust problem is envisioned as

a wide issue, therefore, we try to address them to our best knowledge.

4.1 The Trivial Case

Suppose agent **A** has received a request as call for trust from agent **D**. The trivial case occurs whenever neither there is an entry at **A**'s trust table for agent **D** nor agent **A** cannot discover a trust path to agent **D**. Resolving this case depends on the internal policy and chosen strategy on the level of exposing information to unknown agents. As a simple approach, the solution might be initiating a minimum trust value (such as %5) and creating an entry for the unknown agent in the trust table.

However, as a more interesting technique, will be using the "*reputation"* information of the unknown agent in order to assign a trust value. Generally, any entity in the case of lacking enough information about an unknown entity is encouraged to use the reputation information that might be achieved as a feedback from its former partners. The reputation information might give a clue to make a decision. But taking reputation information into account does not solve the original problem yet. In fact, the problem is still open changing to: how much we can trust the provided reputation information.

4.2 Multi Trust Path

We discussed about the model whenever the called agent discovers exactly one trust path. In most cases, since we are dealing with a network of agents, we encounter to choose one among some discovered trust paths. The result will be worsening if the number of final returned hash values of each trust path changes dramatically. Let us imagine a scenario depicted in Figure 2.

Suppose agent **D**, as the caller agent, sends out a "call for trust" message to agent **A**. Consequently, agent **A** broadcasts a message to discover a trust path. However, agent **A** receives two paths as:

*TrustPath1 [ ] = {***A**, **B**, **C**, **D***}; TrustPath2 [ ] = {***A**, **E**, **D***}*

Therefore, agent **A** has to choose one of the discovered paths. But the question is which path is more secure such that it does not inflate the requested trust value intentionally? Since agent **A** does not know any thing about the trust values among the other agents except its neighbors, therefore as a reasonable reasoning, agent **A** may decide to continue with the path that has started with a neighborhood agent with highest trust value. As a result, agent **A** may choose *TrustPath1,* which starts with agent **B** with trust value equal to 70 rather that *TrustPath2* starting with agent **E** with trust value equal to 30. The evaluation of both paths shows that the choice is not a suitable choice and will result in issuing a far trust value.

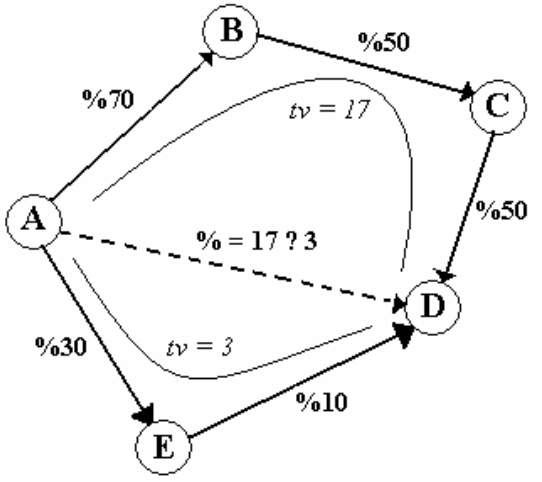

Figure 2 - Multi Trust Path

The evaluations of trust values for two trust paths are:

$$TrustPath1 : tv_{A \to D} = 17$$
$$TrustPath2 : tv_{A \to D} = 3$$

The computed values show that the *TrustPath2* seems more secure than the other one. Although, we might treat these cases as decision making mechanisms for agents for developing the trust model, however, we may solve these situations by using some techniques.

As an approach to deal with multi trust path situation will be running the model as much as the number of discovered paths and with *different* classes of hash values. In fact, each path will have a class of hash values and as a result there would be no interfering problem. Thus, agent **D** collects hash values for all paths and returns them to agent **A**. Accordingly, agent **A**, by receiving the decrypted hash values for different paths may have a better reasoning to decide and issue a more secure trust value for agent **D**.

As another approach, might be using some heuristic methods such as considering: the shortest path or the reputation information, etc.

4.3 Malicious Intermediate Agents

By malicious we mean all kinds of possible attacks to the model caused by either any faked employed agent, to inflate the trust value, or any competitive agent trying to drop off the trust value. In following, we argue that in either case the model is not vulnerable. Figure 3 depicts a simple scenario showing both possibilities and the final issued trust values:

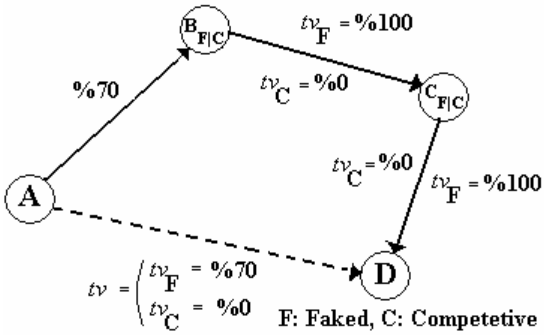

Figure 3 - Faked and competitive agents

In a faked case agent **B** and **C** are acting maliciously trying to inflate the trust values, the *upper bound* of the number of hash values that agent **D** may collect is equal to the maximum value of trust value issued for agent **B**, which is %70. In fact, there is no way for agent **D** claiming for more trust value. Also, it would be reasonable for agent **A** to have an *equal* trust relationship with an unknown agent (agent **D**) that has fully trusted relation (%100) with one of the trusted agent (agent **B**). As a complementary approach, we may consider a penalty for each neighborhood agent for introducing a malicious agent. The penalty will be such as decreasing its trust value.

In the competitive case, the calling agent **D** does not receive any hash values and therefore cannot claim for any trust relationship. As the requirement of the model this case cannot be happened, since only agents with issued trust value greater than zero are allowed to be entered in trust tables. However, if we suppose that the competitive agent intentionally has inserted an entry for the unknown agent with issued trust value equal to zero, then the case is again true since is similar to the case where there exist no other agents trusting the unknown agent **D**. In fact, discovering a trust path with %0 trust value is exactly equivalence to discovering no trust path. However, as we mentioned in trivial case, the calling agent **D** will still have chance to receive a minimum trust value by the called agent **A**.

4.4 Dealing with Long Trust Paths

The number of intermediate agents sited at the trust path may have direct impact of the final result. In fact, the more intermediate agents, the less accurate trust value might be. For instance, consider a discovered trust path with length 20 (i.e. 18 intermediate agents) and each couple of agents sited on the trust path has a trust value equal to 95. As a result the calling agent will be able to collect 35 numbers of hash values and eventually 35 will be assigned the final trust value for the calling agent. However, this value might not be so accurate, since the result is not only depended on the trusted value between each couple agents, but also on the length of the trust path as well. We may deal with this case by employing the following techniques:

4.4.1 Enforcing the Path Length into the Model

In this technique, we may simply enforce the length of the trust path as a parameter into the model such that the length of the path will have an inverse impact on the final value. More precisely, instead of computing the trust value based on the existing trust values, we may compute $TV_{final} = (tv_{Initial}) * f(n)$, in which $tv_{Initial}$ is the trust value computed by the above model and $f(n)$ is a function to adjust the trust value with the length of the trust path.

4.4.2 Enforcing Threshold Values for Different Security Levels

As a second approach, we may set some policies for forwarding the hash values. A policy will be as a threshold value representing the sensitivity of the requested resource and consequently showing a minimum required trust value in order to forward the hash values. More precisely, suppose a

calling agent sends her request for accessing some resources that have been categorized based on their security levels such as Top Secret, Secret, Confidential, and Regular. The called agent based on the security level of the requested resource may set a policy for intermediate agent showing that they are allowed to forward the hash values to the next node whenever their trust relationship is higher or equal to a threshold value. For example for Top Secret resource the threshold would be higher such as 95 as well as for Regular ones the threshold will be lower such as 50. As a result, the called agent can control the behavior of the model more effectively.

This technique might be considered vulnerable since we may deal with some malicious intermediate agents.

4.5 Privacy of Intermediate Agents through the Trust Path

Since through the model, besides the hash values, the called agent forwards the trust path to the next node as well, therefore the identity of all intermediate agents involved in the trust path will be known for all participants. This might be a shortcoming of the model since no agent would like to expose her protected relationships with others. To protect the privacy of the intermediate agents and their relationships, the called agent can apply the simple cryptographic techniques such as public key and encryption. More precisely, suppose a trust path has been discovered as: *TrustPath [ ]* = $\{A_1, A_2, A_3, ..., A_n\}$ such that A_1 and A_n are called and calling agents respectively. The called agent A_1 instead of sending the clear trust path will send the following protected trust path preferably:

$$\{A_1, A_2, en(A_3)_{A_2^{pk}}, en(A_4)_{A_3^{pk}}, ..., en(A_n)_{A_{n-1}^{pk}}\}$$

In other words, each agent's ID will be encrypted by previous agent's public key. Therefore, any agent will be able to decrypt the next agent's ID on the list. As a result, any involved agent will know only its predecessor and successor and has no knowledge about the other agents involved in the trust path. Hence, the privacy of the relationships between agents has been kept.

5 Implementation Architecture

We introduced an efficient approach for trust management in multi-agent systems and in general for distributed environment with straightforward and feasible implementation. The model deals with hash function, public and secret keys encryption, and their verifications, and eventually an agent platform as an environment that agents can live. The implementation is a part of our research activities regarding agent-oriented computing and its integration with service-oriented paradigm. We are working on different aspects of integration of services and agents together to overcome some existing shortcomings of each particular technology by integrating and taking advantages of both.

We have chosen Jini technology as the developing tools for the model. Jini, an open source from SUN micro system, is a service-oriented platform to deploy and develop a service-oriented environment. However, Jini has proved its potential outstanding capabilities in being an agent platform as well. Technically, agents can be developed and deployed such as services and register through the service discovery. The service discovery issues a unique service ID for the registered service and provides some features for service locator to search and communicate with the services. In fact, the service discovery behaves like the directory facilitator agent in any agent platform and provides a unique agent ID for each agent. It seems that Jini can satisfy all requirements of agents' platform as expected in FIPA specification and we believe Jini has a strong potential to be a powerful agent platform.

Figure 4 depicts our scenario in the implemented prototype.

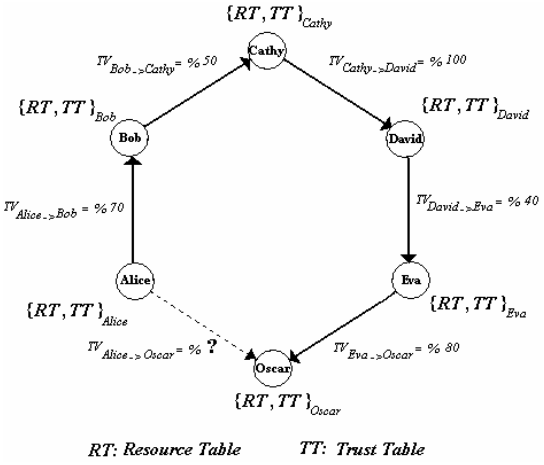

Figure 4 - The scenario of the developed prototype

Through the prototype, we have developed six agents as Jini services. Agents have been named as: Alice, Bob, Cathy, David, Eva, and eventually Oscar. The developed agents register through service (agent) discovery and receive an agent ID. By registration, agents show their existences and readiness for interaction. Jini provides a service discovery named as "reggie" that behaves as directory facilitator or discovery agent enabling agents register themselves through it and make them discoverable by other agents. Figure 5 depicts the deployed agents including registrar.

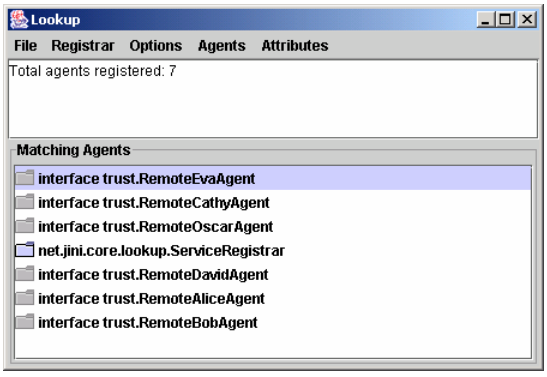

Figure 5 - The deployed agents

Each agent possesses its own local resources and trust table. The resource table keeps track of resources accessibility, which is measured by assigning a minimal trust value for use. As an example, Figure 6 shows the resource table of Alice agent including the minimum required trust values to access and use each of the resources.

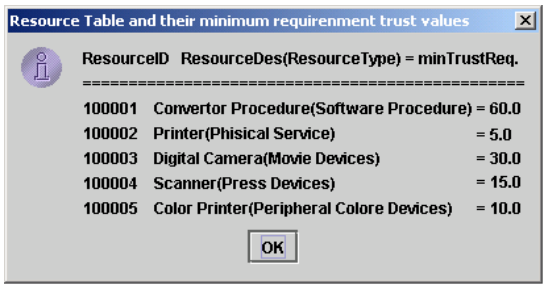

Figure 6 - The resource table of Alice agent

There exist some trust relationships between agents with respect to the privacy of all business relationships and therefore trust values are protected between two individual agents. In other words, an agent may find other agents by searching agent discovery; however, it cannot realize whether there exists any relationship between them including their trust relationship values. The trust values between the coupled agents in our scenario have been shown in Table 2.

Source Agent ID	Destination Agent ID	Trust Value
Alice	Bob	% 70
Bob	Cathy	% 50
Cathy	David	% 100
David	Eva	% 40
Eva	Oscar	% 80

Table 2 - The trust values among agents

As the scenario taken by the prototype, assume Oscar after checking the capabilities and services offered by Alice, would like to start a relationship with Alice. Figure 7 depicts brief capabilities of Alice to Oscar.

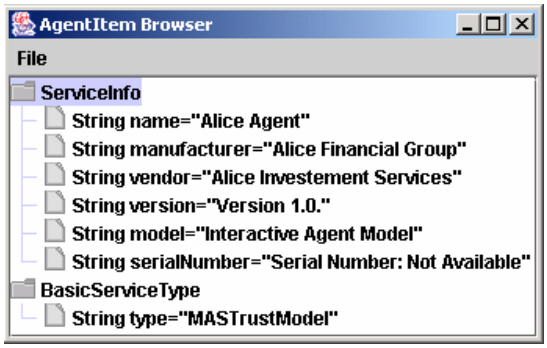

Figure 7- Alice's Capabilities shown to Oscar

Therefore, Oscar by reasoning about Alice and her capabilities, sends out a message to Alice namely as "*Call for Trust*" and waits for any feedback. Alice, by receiving the message, looks up into its trust table to find out whether there is an entry for Oscar or not. Figure 8 depicts Alice's trust table. The address shown in Figure 8 is the agent ID of Bob. Consequently, Figure 9 represents the personal information of Bob.

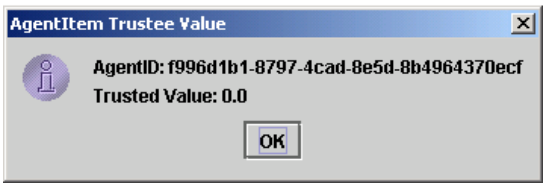

Figure 8 - Alice's trust table

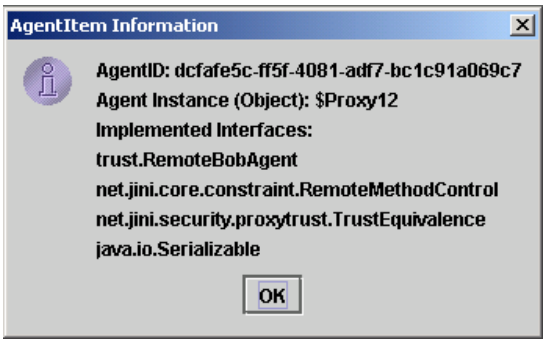

Figure 9 - - Bob identifications

Lacking of any entry in the trust table of Alice for Oscar will result in untrustworthiness and thus Oscar will be as an unknown agent. As a result, Alice multicasts a message "*Path Discovery*" to all its known agents sited at the trust table (Bob in our case). Accordingly, Bob and other agents that receive the multi-case message repeat the same routine. Eventually, they construct and send the trust path to Alice. Alice by receiving the trust path, generate the encrypted hash values and regarding the trust relationships with agents sited at the trust table as well as at trust path sends out a number of hash values to the first node. Each intermediate receiver agent repeats the same scenario until receiving some hash values by Oscar. Oscar returns the decrypted hash values to Alice and consequently Alice by verifying the returned hash values issues a computed trust value for Oscar, which is in our case equal to 11 (Figure 10).

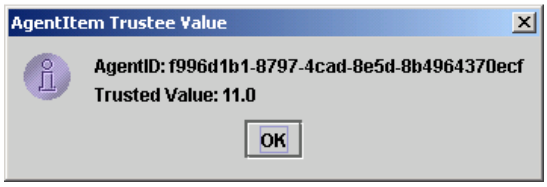

Figure 10 - Issued trust value for Oscar by Alice

6 Conclusion

We present an efficient model addressing trust relationship in multi-agent systems. As we noted throughout the paper, most trust models are too complex to develop and operate. However, the discussed model despite simplicity is able to address the most problems associated with some other proposed models. The model applies some basic cryptography concepts mainly hash function and counting. Any particular agent interested in having a business relationship with other agent, simply asks for trust relationship. Consequently, the target agent, by discovering a trust path, generates some hash values and injects part of them encrypted into the trust path. The hash values by traversing and filtering out through the trust path receives to the unknown calling agent. The number of returned decrypted hash values to the called agent represents the trust value issued for the calling agent.

The model is offering a secure trust management through the multi-agent environment. The presented trust model is computationally secure, since hash functions are computational secure. Furthermore, the privacy of involved agents in trust path is protected and neither calling nor called agent has any idea about the trust relationship and the trust values between nodes at trust path. In fact, the called agent in order to evaluate and assign a trust value to the calling agent does not need to have any information about the trust relationships between other agents including the calling agent.

References

[1] Barber, K. S. and Kim. J., Belief revision process based on trust: Agents evaluating reputation of information sources. In *Proceedings of the Agents-2000 Workshop on Deception, Fraud, and Trust in Agent Societies*, pages 15–26, 2000.

[2] Castelfranchi, C., and Falcone, R., Principles of Trust for MAS: Cognitive Anatomy, Social Importance, and Quantification, ICMAS'98, 1998.

[3] Dellarocas, C., Immunizing Online Reputation Reporting Systems against Unfair Ratings and Discriminatory Behavior, In *Proceeding of the 2nd ACM Conference on Electronic* Commerce, 2000.

[4] Doney, P.M., and Cannon, J. P., An Examination of the Nature of Trust in Buyer-Seller Relationships, *Journal of Marketing*, 1997. 61(2): 35-51.

[5] Finin T., and Joshi, A., Agents, Trust and Information Access on the Semantic Web, SIGMOD Record, 31(4), 2002.

[6] Golbeck, J., Parsia, B., and Hendler, J., Trust networks on the semantic web. In *Proceedings of Cooperative Intelligent Agents*, 2003.

[7] Hess, A., Holt, J., Jacobson, J, and Seamons, K. E., Content-Triggered Trust Negotiation, ACM Transaction on Information and System Security, 7(3), pp. 428-456, 2004.

[8] Josang, A., An algebra for assessing trust in certification chains, In *the Network and Distributed Systems Security* (NDSS'99) Symposium, The Internet Society, 1999.

[9] Marsh, S., *Formalizing Trust as a Computational Concept*. Ph.D. Thesis, University of Stirling, 1994.

[10] Montaner, M., Lopez, B., and Rosa, J., Developing Trust in Recommender Agents, AAMAS'02, July 2002, Bologna, Italy.

[11] Mui, L., Mohtashemi, M., Ang, C., Szolovits, P., and Halberstadt, A. Ratings in Distributed Systems: A Bayesian Approach, 11th Workshop on Information Technologies and Systems (WITS), New Orleans, 2001.

[12] Mui, L., Mohtashemi, M., and Halberstadt, A., A Computational Model for Trust and Reputation, 35th Hawaii International Conference on System Sciences, 2002.

[13] Resnick, P., and Zeckhauser, R., Trust Among Strangers in Internet Transactions: Empirical Analysis of eBay's Reputatoin System, *NBER Workshop on Empirical Studies of Electronic* Commerce Paper, 2000.

[14] Sabater, J., Sierra, C., REGRET: A reputation Model for Gregarious Societies, 4^{th} Workshop on Deception, Fraud and Trust in Agent Societies, 2001.

[15] Schillo, M., Funk, P., and Rovatsos, M., Using trust for detecting deceitful agents in artificial societies. *Applied Artificial Intelligence*, 14:825–848, 2000.

[16] Tan, Y. H., and Thoen, W., An outline of a trust model for electronic commerce, *Applied Artificial Intelligence*, 114(8):849–862, 2000.

[17] Yu, B. , and Singh, M.P., Towards a Probabilistic Model of Distributed Reputation Management, *4th Workshop on Deception, Fraud and Trust in Agent Societies*, Montreal, Canada, 2001.

[18] Zacharia, G., and Maes, P., Trust management through reputation mechanisms. *Applied Artificial Intelligence*, 14:881–908, 2000.