A Secure Communication for a Reputation Management Model in Multi-agent System

Jisha Babu
P G Scholar
Amal Jyothi College of Engineering

Krishnalal G
Assistant Professor
Amal Jyothi College of Engineering

ABSTRACT
Recent multi-agent systems are characterized by decentralized control, autonomy and local views. The application of multi-agent systems on open environment such as internet creates new challenges especially with respect to security issues such as authentication, authorization and privacy. The characteristics of multiagent systems introduce vulnerabilities and threats to secured communication. One practical way to minimize the threats is to evaluate the trust and reputation of the agents. But trust is not enough for secure communication. Many trust/reputation models have given solutions, but they fail to properly evaluate trust when malicious agents start to behave in an unpredictable way and also they fail to address the problem of secure communication between agents. In this paper a secure communication for reputation system in multiagent system is proposed. It provides a flexible way to present differentiated trust and combine different aspects of trust that can meet agent's different needs. And also it allows secure file delivery among agents. The idea of using cryptographic protocols MD5 and DES effectively encrypt the file without any overhead to the network.

Keywords
Multiagent system, Security, Cryptography, Trust, Reputation

1. INTRODUCTION
A multi-agent system is one in which many agents run concurrently, communicating amongst themselves and working toward either individual goals or a common objective [9]. Many computer applications such as the Peer-to-Peer[6], E-business systems, Grid[8] and Semantic Web[7] can be viewed as multi-agent systems, as individual components in each system are both autonomous and flexible in their actions. Interactions between agents may change according to the context of the environment. These agents may be autonomous and heterogeneous. Since they are autonomous, they can decide whom to interact with. Also it is difficult to make assumptions about their present or future behavior since they are heterogeneous.

Trust is crucial for success of these communities. Can we trust a product review made by an unknown customer? Can we trust an unknown eBay seller when promising to deliver the good after we win the auction and pay? Can we trust an unknown peer in a P2P network when claiming that the file we are about to download is not a virus? As we are using possibly untrusted peers to relay information, how can we protect the privacy of our actions? How can we prevent eavesdroppers from keeping track of who we contact and what we do? Only if answers to these questions are positive the concerned community can exist and operate successfully. Most of the involved interactions take place between unknown partners and trustworthiness of the others based only on his own experiences with them.

In traditional systems, little information is given to the user to help in the peer-selection and/or file-selection processes. For example, if a user wants to download a file, the user is given a list of peers that have the requested file. The process of selecting the right peer with no a priori information is very risky. To reduce the risk involved in P2P file sharing systems, peers need to reason about trust, and reputation systems are used.

This paper focuses on trust and reputation management in systems where multiagent works. The proposed system is a feedback-based dynamic trust computation model which can effectively detect sudden strategic alteration in malicious behavior with the additional feature of securing transactions between agents. This model considers some factors in determining the trust of an agent such as satisfaction, similarity, feedback credibility, recent trust, historical trust, expected trust and decay of trust. Here used a novel policy of utilizing exponential averaging function to reduce storage overhead in computing the trust of agents. Specifically, this paper addresses the issue of agent identification; i.e., the authentication problem and confidentiality of data. Because in open multi-agent systems agents must interact with other agents with which they are not familiar. In such environment, a security, non-repudiation and authentication technique is critically required.

The rest of this paper is organized as follows: In section 2, we describe the trust and reputation model in its basic form. The proposed security model for the reputation model is described in section 3. Results of the simulation experiments which test the effectiveness are presented in section 4. Section 5 discusses reviews the related earlier work. Section 6 concludes the paper with a discussion of the future work.

2. SECURE TRUST MODEL
Evaluator is an agent/peer who evaluates the trustworthiness of other agents based on social interactions. Here we name it as `p'. Target is the agent that is judged. Let it be `q'. In the following section we describe each and every parameter and processing in detail. The procedure in the model usually follows the steps as shown in Fig 1.

- Send Query-An evaluator agent starts a query for a specific service. A service to hold a file. Actually the evaluator uploads a file to that agent.
- Calculate trust value for each agent
- Choose agent with highest trust value
- Deliver the file securely

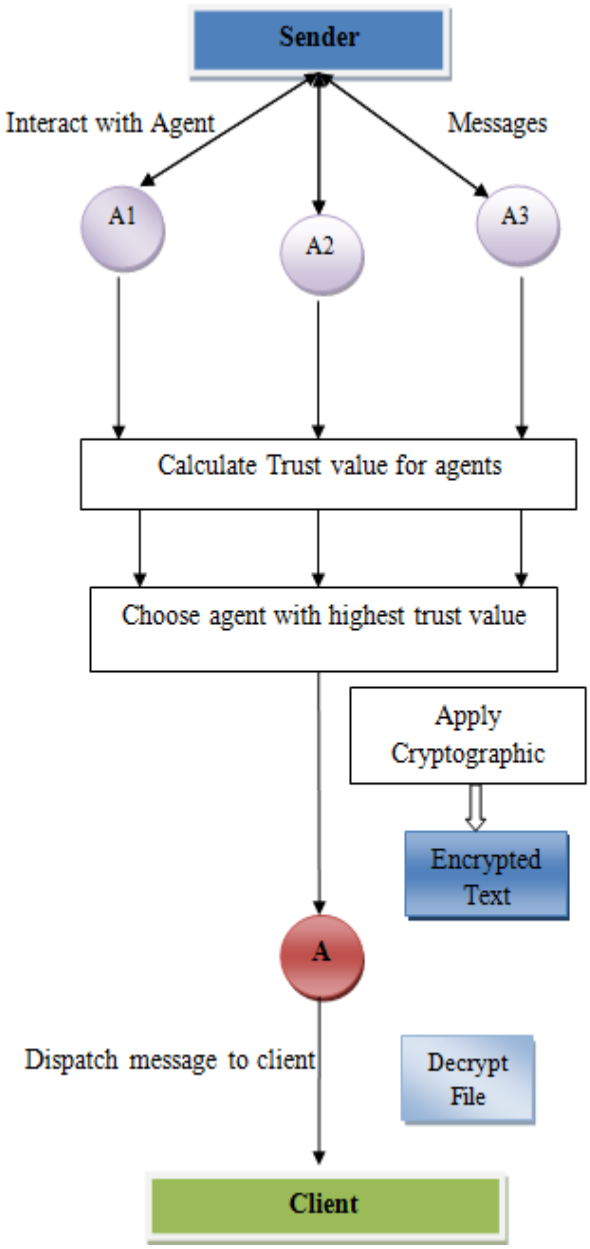

Fig 1. Architectural Model

Trust can be calculated based on different parameters. They can be file quality or file uploading and downloading speed. After responding each agent to the target agent the evaluator agent calculates the response time. Here the response time means the time in which each agent taken to receive file or request and respond to the evaluator. With this response time each parameter should be calculated. In the following section describes each parameter for calculating trust.

2.1 Satisfaction

Let *Sat(p,q)* represent the amount of satisfaction agent *p* has upon agent *q* based on its service up to n transactions in the t^{th} time interval. The satisfaction function is defined as follows:

$$\delta(p,q) = c * Res[p,q] + (1-c) * (Res[p,q] - 1)$$

$$\propto = threshold + c * \frac{Res[p,q]}{1+\delta(p,q)}$$

$$Sat(p,q) = \propto + (1-\propto) * (Res[p,q] - 1)$$

Here, *Res[p,q]* is the time needed to upload a file to each target agent and c is some user defined constant factor which controls to what extent we will react to the recent error δ(p, q). The threshold represents a threshold which is used to prevent $\propto$ from saturating to a fixed value. Initial value of α is set to 1. The threshold value is set to 1.

2.2 Similarity

Similarity is computed by determining the difference in satisfaction rating over common set of interacted agents and has then used the computed difference rating to define the similarity. The personalized difference in satisfaction rating between agents p and q denoted as *Diff(p,q).* To measure the similarity between agents *p* and *q*, agent *p* first compares *Diff(p,q)* with the similarity deviation constant (τ). We see that as *Diff(p,q)* increases beyond τ , similarity *sim(p,q)* decreases. The similarity function *Sim(p,q)* is defined as follows:

$$diff(p,q) = 2 * (sat(p,x) - (sat(q,x))^2)$$

$$Sim(p,q) = \begin{cases} \mu + \frac{1-a}{20} \\ if\ (diff(p,q) < \tau \\ \mu * (a - (1 - \frac{a}{20}) \\ else \end{cases}$$

2.3 Feedback Credibility

During trust evaluation, feedbacks provided by agents with higher credibility are trust worthier. So they have more weightage than those from agents with lower credibility. To calculate the feedback credibility the three target agents communicate with a third agent we call it as client. When the transaction completes with client each target agent response to the evaluator. Evaluator checks the transaction time needed to complete the transaction. Let FC(p,q) represent the feedback credibility of agent *q* from agent *p*' s viewpoint.

$$FC(p,q) = Sim(p,q) + Tr[p,q]$$

where *Tr[q,x]* is the total time needed to complete the transaction between client and target agent and response to the evaluator.

2.4 Direct Trust

Let *DT(p,q)* is the direct trust between *p* and *q*.

$$DT(p,q) = \begin{cases} 1 - \ln(Sim(p,q) & if\ (sim(p,q) > \theta \\ 0 & else \end{cases}$$

where θ represents the lowest allowed value of similarity. As we can see direct trust is a direct logarithmic function of similarity for its slow rise to the highest attainable value. Thus the agents with higher similarity with respect to the evaluating agent have higher direct trust.

2.5 Indirect Trust

The evaluating agent then aggregates recommendation from other agents along with the feedback credibility of the recommenders. Let *IT(p,q)* represent the indirect trust that agent *p* computes about agent *q*.

$$IT(p,q) = \sum FC(p,q) * DT(x,q)/FC(p,q)$$

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

2.6 Recent Trust

We have defined recent trust as a weighted combination of direct and indirect trust. Direct trust is given higher weight as the evaluating agent performs more and more interactions with the target agent. Let *RT(p,q)* represent the recent trust that agent *p* has upon agent *q*. we set the value of β as

$$RT(p,q) = \beta * DT(p,q) + (1-\beta) * IT(p,q)$$

2.7 Historical Trust

By using an exponential averaging function, we reduce the storage overhead associated with storing the previous recent trusts. Let *HT(p,q)* represent the historical trust that agent *p* has about agent *q*. The function is defined as follows:

$$HT(p,q) = \frac{\rho * HT_{n-1}(p,q) + RT_{n-1}(p,q)}{2}$$

where ρ is the forgetting factor and *HT0* = 0. With historical trust, present malicious agents cannot forget their past and starts acting well. In other words, since we are keeping track of an agent's past behavior, it cannot cheat other agents into believing that it is a good agent by just behaving well in the recent transactions. For an agent to be considered as good, it has to perform in a significantly large number of transactions.

2.8 Expected Trust

We are combining both recent trust and historical trust to get a prediction of the future trust. Let *ET(p,q)* represent the expected trust of agent *q* from agent *p*' s perspective. Expected trust is calculated by the following equation:

$$ET(p,q) = \eta RT(p,q) + (1-\eta)HT(p,q)$$

Initially, η is set to 0.5.

2.9 Decay Model

We apply a decay function on satisfaction metric. The decay function is given as follows:

$$DM = Sat(p,q) * \mathrm{e}$$

where Sat(p, q) represents the value of satisfaction after decay.

2.10 Overall Trust Metric

This is the actual trust value used in prioritizing all agents. It is computed using expected trust and indirect trust. Let *Trust(p,q)* represent the final trust value agent *p* places upon agent *q*.

$$Trust(p,q) = ET(p,q) * IT(p,q)$$

From the equation, it is proven that agents with high expected trust values but with low indirect trust will eventually have low overall trust value. Therefore, an agent will use this equation to select the target agent with the highest trust value as this metric combines all the factors we have discussed so far.

3. SECURE FILE DELIVERY USING MD5 AND DES

After finding the agent high trust value, the next step is to forward the secure data by means of encryption technique. We have different methods to extend or strengthen known techniques like increase the number of rounds (as in MD5); add some coding or scrambling steps (as in SHA-1); increase the buffer size and make the mixing step vary with the round. All of these are natural attempts to increase the security of a hash function design. An example of such an assumption is the ideal-cipher model for DES, discussed below.

Building hash functions based on block ciphers is the most popular and established approach. In this approach, the compression function is a block cipher with its two inputs representing a message block and a key. At present, a protocol requiring 2^{128} operations to defeat is considered strong and secure. Here I implemented it as an integration of DES (Data Encryption Standard) along with hash function such as MD5 as shown in Fig 2. Because DES works on Symmetric Key Cryptography, both sender and receiver have the same key.

The output of MD5 operation in each block will be used as input for DES function. MD5 gives 128 bit long output. And DES accepts 64 bit input block at a time. So, first, the output of MD5 is divided into two blocks each of 64 bit long, first with left 64 bits and second with right 64 bits. Then apply DES on both blocks respectively. The output will be again of 64 bit (total of 128 bit). This overall 128 bit output will then be used as 128 bit *CVq* for MD5 processing of next block of input.

The proposed algorithm may be stated as:

Step 1: Begin

Step 2: Append padding bits to the message

Step 3: Append original message length to the O/P of Step2.

And get 512 bit *L* blocks of message.

Step 4: Initialize MD buffer (128 bit)

Step 5: Repeat Steps 6 to 10 for all *L* blocks

Step 6: Generate 128 bit digest of i^{th} block

Step 7: Divide O/P of Step 6 into two blocks of 64 bits each.

Step 8: Encrypt both blocks (O/P of Step7) using DES.

Step 9: Concatenate 64 bit outputs of Step 8.

Step 10: Use the O/P of Step 9 as CV for next 512 bit block.

Step 11: Use final O/P of last *L*th block as hash value to be transmitted to receiver for message integrity along with authentication.

Step 12: End

After encrypting the file the evaluator agent sent the secret file to the target agent. And this file can forwards to the client securely. The strength of this variant is difficult to estimate. The only observation that can be made with certainty is that it is stronger than using only MD5. It may well be that a brute force attack on this method requires, on the average, 2^{196} trials. (Assuming that the intruder has perform attack that equivalent to both attack on MD5 plus attack on DES at the same time).

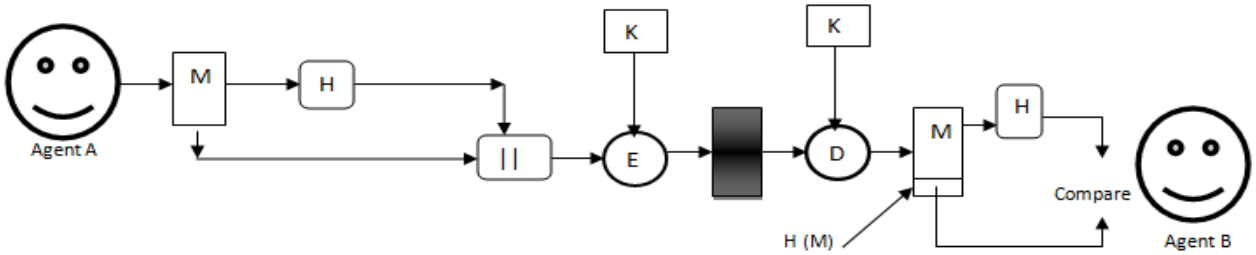

Fig 2. Secure File Delivery using MD5 and DES

4. EXPERIMENTAL DISCUSSION

Table 1 shows the key parameters for the simulations along with their default values. The simulations were run on a 2.20 GHz Core i5 processor machine with 4GB of RAM. The simulation component most specific to file-sharing (as opposed to general resource-sharing) is our query model. We assume total 3 files. Simulation is in a java-based platform hosting one evaluator agent, three target agents and one client agent. For the sake of simplicity, the evaluator agent in the system plays role of calculating the trust of three target agent, either the role of uploading the file to target agent.

Table 1. Simulation Environment

Name	Number
Evaluator agent	1
Target agent	3
Rounds	1
Network topology	Peer-to-Peer Network

4.1 Result and Discussion

Table 2. Final Trust Values for Agents

File Provider	Uploading time	Transaction time	Final trust value
Agent 1	37	6	13.261
Agent 2	42	13	3.061
Agent 3	50	6	1.842

Table II shows the uploading time taken by the evaluator agent in seconds. Transaction time is the total time needed for each target agent to complete the transaction. Based on this information final trust value is calculated. Fig.3 shows the trust rate of three target agent after first round. The experiment shows that the secure trust model helps peers to select file providers.

The cost of computing final hash value will be more than simple, as the proposed solution also includes DES algorithm in between each step. Because of its slow performance, asymmetric cryptography is not a viable alternative for sym-metric algorithms for these applications. And thus DES is a suitable choice for it. One more reason for its slowness is that, at least two encryptions are required to hash a single block, and almost all constructions modify the key after each encryption. But, the analysis of the performance of cryptographic algorithms is closely related to their security: high performance applications require an optimal trade-off between security and speed. MD5 with DES mechanism effectively reduce the overhead to the network. The security of the proposed solution can be split into a consideration osf underlying block cipher.

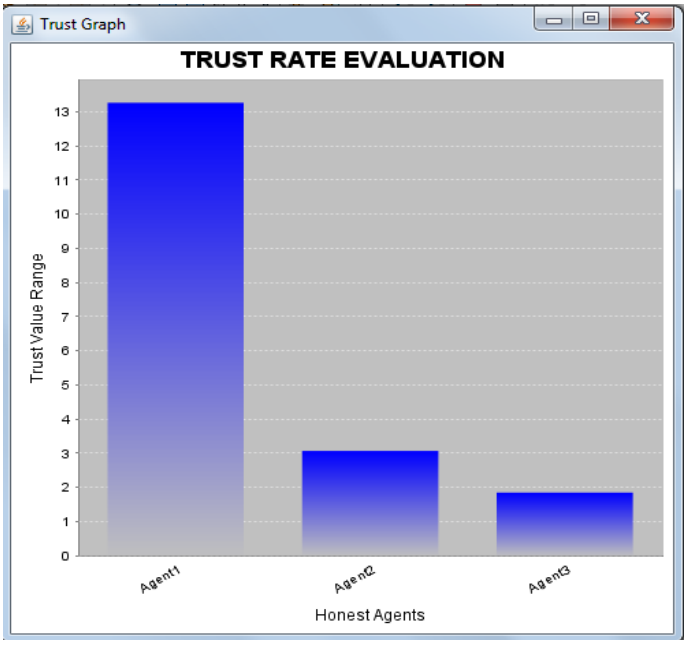

Fig.3 Trust Evaluation

5. RELATED WORK

In this section we look into some important work that goes on trust and reputation system and their security.

The Bayesian network model[1] is an interactive/witness reputation model in which service providers behavior is analyzed in different aspects such as download speed, quality, and type. In this model, a naive Bayesian network represents conditional dependencies between the reliability of the service provider and the analyzed aspects. This approach in unsuitable in the cases where the preferences of the recommender and evaluator agents do not perfectly match and the collected

information does not accurately represent the trustworthiness of the service provider agent.

The FIRE[2] model is a witness-based trust model that collects the required information from other agents in the form of advisors. In this model the trust value is derived from direct experience, witness information, role based rules, and third-party references. The collected data is used by the evaluator agent to compute the trustworthiness of a particular target agent. Target agents propose some colluding referee agents to mislead the evaluating agent. Thus, in these cases the final trust rate would be affected by non reliable information about the target agent.

Regret[3] is a decentralized trust and reputation system designed for e-commerce environments. The system takes into account three different sources of information: direct experiences, information from third party agents and social structures. The direct trust, witness reputation, neighbourhood reputation and system reputation are introduced in Regret where each trust and reputation value can has an associated reliability measure. It does not address the collusion problem associated with computing global reputation. No security is provided during the communication between agents.

In the EigenTrust[4] algorithm assigns to each peer in the system a global trust value based on peers history of uploads. This trust value reects the experiences of all peers with the peer. Normalizing local trust values will not make the distinction between peers who the requester peer did not interact with and peers that performed more unsatisfied transactions than satisfied ones. The scheme requires reputations for each provider peer to be computed on-demand which requires cooperation and collaboration from a large number of peers in performing computations.

Freenet[5] is not a private P2P network. Freenet uses a combination of signatures of data and signatures of pseudonyms of the peer which provide data. The Freenet network effectively hides the origin of data, and only allows a peer which holds the private key of the signatures to inject data under the given pseudonym. The injected data is distributed among the peers in the DHT. It is done using a stochastic algorithm. Although files are encrypted by a randomly generated encryption key, the confidentiality is not guaranteed. Indeed, the decryption key is stored along with the file's identifier, and any requester could read the file content.

6. CONCLUSION AND FUTURE WORK

In this paper a novel approach for evaluating multi-agents and to achieve trust and reputation among agents is proposed. It can also ensure secured communication among agents. The simpler design of algorithm is easier to optimize and analyze the security. The presented system makes use of already established MD5 and DES, that are widely accepted and being used all over the world. The proposed solution is of typical importance for applications where message integrity and message/sender authentication are of equal importance. It may be particularly applicable to environments where these security requirements have made the implementation of certain security services prohibitively expensive. Experiments confirm that these approaches can achieve better effectiveness, efficiency and exibility compared to the systems that do not use them. However, these methods are not perfect, there is need to do more improvements so that these approaches can be applicable to real-world environments.

When a malicious peer can create multiple identities, then they can switch its identity easily and acting malicious. One way to discourage participants from changing identifiers is that to pay dues. Or make an entry fees to get into the network. Another possible way is that a peer is given a single identifier that is unrelated to the person's true identity; We call these once-in-a-lifetime identifiers. Thus the whole system contains unique peers. This can be a future work to this model.

7. REFERENCES

[1] Y. Wang and J. Vassileva, `Bayesian Network-Based Trust Model' Proc. IEEE/WIC Intl Conf. Web Intelligence (WI 03), pp. 372-378, Oct. 2003.

[2] N.R. Jennings, T.D. Huynh, and N.R. Shadbolt, `FIRE: An Integrated Trust and Reputation Model for Open Multi-Agent Systems', Proc. 16th European Conf. Artificial Intelligence (ECAI 04), pp. 18-22, 2004.

[3] J. Sabater and C. Sierra, `REGRET: A Reputation Model for Gregarious Societies', Proc. Fourth Workshop Deception, Fraud and Trust in Agent Societies, pp. 61-69, 2001.

[4] S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina, `The EigenTrust Algorithm for Reputation Management in P2P Networks', Proc. 12th ACM Intl World Wide Web Conf. (WWW 03), pp. 640-651, 2003.

[5] Clarke, I, Hong, T, Miller, S, Sandberg, O, Wiley,B: `Protecting Free Expression Online with Freenet', IEEE Internet Computing article (2002)

[6] A. Oram, editor. Peer-to-Peer: Harnessing the Power of Disruptive Technologies. OReilly Associates, Inc., Sebastopol, CA, USA, 2001.

[7] T. Berners-Lee, J. Hendler, and O. Lassila. `The semantic web. Scientific American', May 2001.

[8] I. Foster, C. Kesselman, and S. Tuecke, `The Anatomy of the Grid: Enabling Scalable Virtual Organizations', Intl J. High Performance Computing Applications, vol. 15, no. 3, pp. 200-222, 2001.

[9] M. Woolridge and M. J. Wooldridge. Introduction to Multiagent Systems. John Wiley Sons, Inc., New York, NY, USA, 2001