Mobile Agent Security in MANET Using Reputation

Chandreyee Chowdhury and Sarmistha Neogy

Dept. of Computer Science and Engineering
Jadavpur University
sarmisthaneogy@gmail.com

Abstract. The emerging trend of using mobile agents for mobile adhoc network (MANET) applications intensifies the need for protecting them. Here we propose a distributed trust based framework to protect both the agents and the host platforms (running at the nodes). This paper develops a distributed reputation model of MANET using concepts from Dempster-Shafer theory. The agents and the host platforms work together so that each trusted node may form a consistent trust view of MANET. An agent may share its view of the network with a visited host. To speed up convergence, a node broadcasts information regarding a suspected node. Thus an inactive node, without deploying agents may also get a partial view of the network. The agents use combination of encryption and digital signature to provide privacy and authentication services. Node mobility and the effect of environmental noise are considered. The results show the robustness of our proposed scheme.

Keywords: Mobile Agent, Security, Digital Signature, Trust, Mobility Model, Dempster–Shafer Belief Theory.

1 Introduction

Nowadays mobile agents are used for various kinds of networked applications like service discovery, network discovery, automatic network reconfiguration etc. where the agents roam in the network and consequently get the task done. But securing agents is a big concern particularly when the underlying network is (Mobile AdHoc Network) MANET that typically undergoes continuous topology changes. Applying cryptographic functions [1] are not sufficient rather if we can prevent an agent from visiting a malicious node most of the risk factors are covered. To enforce, we use the concept of trust that has received considerable attention in information security literature. In a way, trust and security are two sides of the same coin, because if a system is secure, it is trusted, and if it is trusted, then it must be secure and vice-versa [2].

This observation leads us to consider security as a property of a system in a given environment, and trust as a subjective belief resulting from assessing a system and its environment. As in [1] we define trust as a subjective quantified predictor of the expected future behavior of a trustee according to a specific agreement elicited from the outcomes of the previous interactions, both from direct experiences and indirect experiences. Reputation of an individual host refers to certain characteristics related to its trustworthiness. In a mobile agent system reputation can be obtained from agent's interaction feedbacks about a visited host's performance in fulfilling its obligations. Indirect experiences can also be considered which is gathered from other trustworthy nodes. Thus agents are encouraged to behave in a cooperative manner so that from their feedbacks, the malicious hosts can be easily and efficiently identified.

In this paper we describe a trust based framework for mobile agent based system(MAS) in a dynamic and hostile MANET environment. This paper shows how the agents' feedbacks (direct experiences) and node dynamicity help the hosts to converge to a consistent view of the trustworthy nodes in the network. This point onwards, the terms node and host are used interchangeably unless otherwise stated.

Additionally, in order to speed up convergence, once the node/s identified a host to be malicious (via the agent/s deployed by it/them), it will broadcast this information (indirect experience) to others. But a receiver will respond to this message promptly if it knows quite well without any uncertainty the sender to be trusted. Thus our definition of trust may range from complete belief to complete disbelief to full uncertainty as well. The following section (2) describes the design of our reputation system. In section 3, state of the art regarding this area of research is elaborated. The next section illustrates the way we model MAS on MANET detect a malicious agent and/or platform (depending on trust level defined later) in a distributed way (using the reputation system designed in section 2). Section 5 gives the experimental results to show the robustness of our scheme followed by concluding remarks (in section 6).

2 Trust Model

A reputation system [3] represents a promising method for fostering trust among complete strangers and for helping each individual to adjust or update its degree of trust towards its corresponding interaction partner and thereby reduce uncertainty. In general, threats found in a reputation system are Strategic rater [4], Strategically malicious visited host [3] that can be addressed by taking opinions from the agents and peers.

Due to the inherent distributed nature of MANET nodes can only have imperfect knowledge about others. Thus it is impossible to know with certainty whether a host is malicious or not; but we can only have an opinion about it, which translates into degrees of belief (how much the host is trustworthy) or disbelief (how much the host is suspicious) as well as uncertainty in case both belief and disbelief are lacking. We express this mathematically [5] as:

$$b+d+u=1 \tag{1}$$

Here b, d, u designate belief, disbelief and uncertainty respectively.

The design of our reputation system is shown in figure 1. It focuses on how to exploit the collected information to quantify the reputation of a node so that an agent can be prevented from migrating to a malicious node. The parameters (b, d and u) are updated from direct and indirect observations. Additionally an aging factor is added for the indirect observation part as dynamicity of MANET may reduce the significance of a message broadcast with time (figure 1).

2.1 Direct Observation

A portion of agent code is meant for computing hashcode(i.e., hashcode computation algorithm) of itself along with data. This part is signed by the agent's owner since the algorithm for hashcode computation is not expected to be changed en-route. Moreover this is also encrypted using public key cryptography in order to hide it in transit. Thus each agent carries the following

ENCRYPT$_{\text{public\_key}}$[SIGNATURE$_{\text{owner}}$(code for hashcode computation + private key of agent)] + application code + data

We encrypt the private key of the agent so that the agent can detect any attempt to break this ciphertext even if the encryption technique is not foolproof. Here application code refers to the purpose for which it is deployed by its owner. Upon reaching a host site an agent takes its own hash code to check if it is attacked in transit or by the current site. Once authenticated it executes the application code and updates the results in its data. Then takes a new hashcode and replaces the old one. If in the mean time the agent finds anything suspicious, it marks that and returns back to its owner. Otherwise the agent moves to a new host site according to the task given. In the end, every agent shares its experience with the owner. Thus we assume that an agent eventually finds its owner whenever it needs. Here we take Beta(α,β) distribution as in [5].α_{ij} represents the no. of good transactions between the agents deployed by owner$_i$and node$_j$. Thus for each positive feedback from agents, $\alpha_{ij}$is incremented. Otherwise β_{ij} is incremented. But β_{ij} may not reflect the exact scenario as an agent may be attacked in transit but it will be able to detect it only when it reaches at a host site and checks its own hash code. Also a node may act as a good host site for an agent (for some time) and behave maliciously to others (later on). Thus there is an uncertainty associated with the agent's observation. To deal with such issue, an approach proposed in [5], leveraging on the Dempster–Shafer Belief Theory [6] is adopted here to quantify the uncertainty of some random variables. Thus the uncertainty in predicting the nature of node$_j$ by node$_i$ is [7]:

$$u_{ij} = \frac{12 * \alpha_{ij} * \beta_{ij}}{(\alpha_{ij} + \beta_{ij})^2 * (1 + \alpha_{ij} + \beta_{ij})} \tag{2}$$

Now if agent$_k$'s observation about node$_j$ is $p_j^{k,}$ then α_{ij} is updated as follows

$$\alpha_{ij(new)} = \omega * \alpha_{ij(old)} + (1 - \omega) * p_j^k \tag{3}$$

Here weighted average is taken, where ω ($0<\omega<1$) represents our absolute trust on each agent's observation as this observation may change from time to time taking care of network dynamicity. Also a malicious host may behave rationally for some time to gain trust from its peers. To tackle this part ω should be close to 1. Again ω behaves as the aging factor for values to close to 0. However if an agent$_k$ deployed by owner$_i$ retracts because it has visited a suspicious host site (node$_j$) then $\beta_{ij}$is updated in a similar way as

$$\beta_{ij(new)} = \omega * \beta_{ij(old)} + (1 - \omega) * p_j^k \tag{4}$$

An agent while visiting a host site may also share is experience with the host. This will also be considered as direct experience. However lesser importance will be given to this agent's observation (p_j^k) so as to protect the host from misleading observations given by suspicious agents. In essence, (1-ω) should be small, for visiting agents.

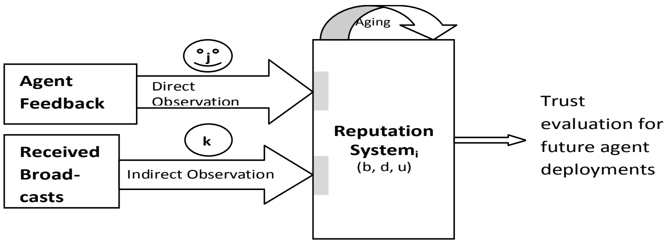

Fig. 1. Trust evaluation framework at hosts taking direct feedbacks from agent$_j$ and indirect observation from node$_k$

Now these values of α_{ij} and β_{ij} are fed to the reputation system that maps these to a tuple (b_{ij}, d_{ij}, u_{ij}). Here u_{ij} is calculated using eqn 2. Consequently following eqn 1, the total certainty (= (1-u_{ij})) is divided into b_{ij} and d_{ij} according to their proportion of supporting evidence as follows [5]:

$$b_{ij} = \frac{\propto_{ij}}{\propto_{ij}+\beta_{ij}}\left(1-u_{ij}\right) \tag{5}$$

$$d_{ij} = \frac{\beta_{ij}}{\propto_{ij}+\beta_{ij}}\left(1-u_{ij}\right) \tag{6}$$

Here b_{ij} gives node$_i$'s belief in node$_j$'s behavior as safe host site for agents deployed by node$_i$. Similarly d_{ij} indicates node$_i$'s disbelief and u_{ij} reflects node$_i$'s uncertainty of predicting node$_j$ as a safe host site for its agents.

In this way with the help of Dempster–Shafer Belief Theory [6] uncertainty can be significantly reduced even though perfect accuracy could not be achieved.

2.2 Indirect Observation

For faster convergence of trust the nodes share information among each other. Upon finding a node to be suspicious, each node broadcasts a message to all nodes in the network to inform about that (suspicious) node. A node is suspected if its $b<u<d$. This information indirectly influences a node's view of the network. The influence is indirect as the sender of the broadcast message may not be a trusted node at all. Thus to prevent a malicious node from influencing others through malicious broadcasts, receivers update their views depending on their belief on sender of the received broadcast massage. The broadcast message has the following format (shown in figure 2). Here (b,d,u) is the tuple obtained from final observation of the sender about the suspicious node. Obviously b will be closer to 0 and d and/or u will be close to 1. The duration field stops this message from hopping in the network infinitely, by making it invalid once duration elapses. This second-hand information helps a node to cope with long delays and frequent network partitions.

Sender id	Code indicating message broadcast	(b,d,u) of the sender about the suspicious node	Duration

Fig. 2. Format of broadcast message

Let $b_l^{i:j}$ represents belief (b) of node$_i$ on node$_l$ while taking indirect observation from node$_j$. So this parameter depends on two factors-(i) node$_i$'s belief on node$_j$ and (ii) node$_j$'s final observation on node$_l$ as indicated in the broadcast message received by node$_i$. Thus following the approaches proposed in [5] ($b_l^{i:j}$,$d_l^{i:j}$,$u_l^{i:j}$) can be formulated as

$$b_l^{i:j} = b_j^i \times b_l^j \tag{7}$$

$$d_l^{i:j} = b_j^i \times d_l^j \tag{8}$$

$$u_l^{i:j} = b_j^i \times u_l^j + d_j^i + u_j^i \tag{9}$$

It can be noted that node$_i$'s disbelief in node$_j$'s observation becomes an uncertainty for predicting node$_l$. Also node$_i$'s uncertainty on node$_j$ amounts to the uncertainty of node$_i$ in predicting node$_l$'s future behavior. Thus a node predicts about the future behavior of a node taking indirect feedbacks from all broadcasts that it has received in the last time interval (Δt) and updates its view (b, d, u) as follows [5]

$$b_{i:l} = \sum_{k \in S} \frac{b_l^{i:k}}{|S|} \tag{10}$$

$$d_{i:l} = \sum_{k \in S} \frac{d_l^{i:k}}{|S|} \tag{11}$$

$$u_{i:l} = \sum_{k \in S} \frac{b_k^i \times u_l^k + d_k^i + u_k^i}{|S|} \tag{12}$$

Here $b_{i:l}$ represents the indirect belief of node$_i$ about node$_k$. S denotes the set of nodes that sent message broadcasts (that node$_i$ received) in the last time interval.

2.3 Combining Direct and Indirect Observation

After collecting first-hand information from the agents and second-hand information from broadcast messages, a node attempts to integrate them all to come to a unified conclusion about future behavior of the nodes. Thus the comprehensive belief ($b_j^{i(f)}$), disbelief ($d_j^{i(f)}$) and uncertainty ($u_j^{i(f)}$) of node$_i$ on node$_j$ are derived from the following eqns, as in [5]

$$b_j^{i(f)} = \varphi_1 \times b_{ij} + \varphi_2 \times b_{i:j} \tag{13}$$

$$\mathrm{d}_j^{i(f)} = \varphi_1 \times d_{ij} + \varphi_2 \times \mathrm{d}_{i:j} \tag{14}$$

$$u_j^{i(f)} = 1 - b_j^{i(f)} - d_j^{i(f)} \tag{15}$$

Where

$$\varphi_1 = \frac{\gamma \times u_{i:j}}{(1-\gamma)\times u_{ij} + \gamma \times u_{i:j} - 0.5 \times u_{ij} \times u_{i:j}} \quad (16)$$

$$\varphi_2 = \frac{(1-\gamma)\times u_{ij}}{(1-\gamma)\times u_{ij} + \gamma \times u_{i:j} - 0.5 \times u_{ij} \times u_{i:j}} \quad (17)$$

Here γ ($0<\gamma<1$) indicates a node's confidence on the agents it deployed. Larger values of γ (>0.5) means a node tends to trust its agents whereas smaller values (<0.5) indicates that a node tends to trustothers' recommendations. Now trust can be quantified from the comprehensive belief, disbelief and uncertainty as [2][7]

$$T_{ij} = b_j^{i(f)} + \sigma \times u_j^{i(f)} \quad (18)$$

Here σ gives relative atomicity based on the principle of indifference. Here the possibility that an agent's visit to a host will be safe or unsafe indicates two mutually exclusive and collectively exhaustive states. The principle of indifference states that if all (say n) possibilities are indistinguishable except for their names, then each possibility should be assigned a probability equal to 1/n. Thus here σ could be 0.5. Among the total uncertainty associated with an agent's visit, there is a 50% chance that the agent will be safe. But we can tune this parameter more accurately in a sense that for higher values of disbelief, there is a possibility that $\sigma<0.5$ and vice versa.

Consequently depending on the trust values calculated from eqn 18 and the safety requirement of the applications (running at the nodes) that deploys agents, an owner decides an agent's task route or asks it to avoid suspicious host sites.

3 Related Works

This section summarizes the literature related to trust management schemes in MANETs and mobile agent based systems.

Trust-based data routing has been extensively studied in wireless networks including MANETs [5], [7]. The basic framework of a Trust Management System (TMS) includes a Reputation System (RS) and a Watchdog like the one in figure 1. The watchdogs normally monitor the event of data forwarding and counts the arrival of ACKs corresponding to data sent out/forwarded. To cope with mobility, in [7] multiple feedbacks are compressed together. But using mobile agents for this purpose will yield far better results as agents are designed to cope with frequent disconnections and limited bandwidth that characterizes MANET especially delay tolerant networks [7]. In [5] it is shown that mobility increases the chance of direct interaction with a node.

Trust management system for mobile agents is also well studied [1] in literature. In [8] a distributed reputation management model is proposed that is based on Dempster-Shafer theory of evidence. A trust model is described in [9] for MAS that considers the information provided from several sources (interaction trust, witness reputation, role based trust and certified reputation). It also uses Dempster-Shafer theory of evidence. In [1] a reputation-based trust model is proposed for mobile agents. Bayesian Network based trust computing is used for strategically malicious trustee prevention.

But these works are not focused on MANET and so the effect of dynamic topology changes, noisy environments, and more importantly mobility are not considered in these works. Thus, securing mobile agents and nodes in MANET by using the notion of trust is a comparatively new research paradigm.

4 Our Work

In this paper, we define our MAS (S) to be consisting of M independent agents deployed by k owners that may move in the underlying MANET. To describe our model we will take help of the following abstraction of an Ad hoc network. Here we try to protect mobile agents and prevent trusted nodes from sending agents to malicious ones. We assume the compromised nodes can send malicious agents to mislead a node about its trust level.

In our previous works [10] we have described our model of MANET, which is also adopted here. The mobility of nodes in MANET can be simulated using smooth random mobility model (SRMM) [11]. (x_i,y_i) represent location of $node_i$ at an instant according to SRMM. Now the average received power (p_r) is a function of the distance between the transmitter and the receiver. Here we take the two-ray model for radio propagation in order to show how the transmitted signal with power (p_t) suffers from multipath propagation while reaching the receiving end.

In this scenario we can think of a mobile agent as a token visiting one node to another in the network (if the nodes are connected) based on some strategy as needed by the underlying applications to accomplish its task. Mobile agents are deployed for various purposes like service discovery [12]. Thus an agent starts its journey from a given owner and moves from one node to another depending on a P*riority list* as explained below.

The following data structures are needed Priority_list of agent j: node_id and trust_level(unvisited 0; suspected -1; trusted +1)

(α ,β): positive integers to be kept at node Default trust level : TS (> k)

Trust level view at $node_i$:(Trust $level_1$, Trust $level_2$, Trust $level_3$,………………..$)_I$ where trust $level_1$ represents the trust value assigned to node id=1 by $node_i$ according to eqn 18.

Initially the priority lists (PL) of all agents have 0 trust level for all nodes. Accordingly $node_i$'s view of the network will be $(TS,TS,……)_{i.}$

The workflow can be divided into two parts: (i) Computation/Action in mobile node and (ii) functions of the agents. Algorithm I gives the function of the agents that helps to collect first hand information. Then algorithm II running at the nodes takes its input from algorithm I and any broadcast message received by the node to update the distributed trust model and hence the node's trust level view of the network. Steps followed by each agent

```
Algorithm – I: Agent_code()
  1. While task given to the agent is not completed
    1.1. Move to an agent site (MN) (unvisited) according to the PL provided.
```

```
        1.2. If that destination falls in the same cluster as it is now residing, the agent
             moves to the new destination with probability p
        1.3. Before processing, take hashcode of the agent's own code and data.
        1.4. If the hashcode matches with the one stored in a *secured way*(see section
             2.1) in the agent's data, then
            1.4.1. Gather information needed by the application that deployed
                   this agent.
            1.4.2. Update the computed results.
            1.4.3. Compute hashcode of the code and data and store it in a
                   secured way.
            1.4.4. Share status of the PL with this trusted node.
        1.5. Go to step 2.//inference: most likely agent's visit was not safe
      2. Retract back to the owner.
      3. Stop.
Steps followed by every mobile node (host platform)
    Algorithm – II: *MN_code()*
1. Input network configurations.
2. For $t=t_0$ to T repeat the following.
    2.1. Some nodes may also fail because of software/hardware failure according to
         Weibull distribution. If a node fails then go to step 3.
    2.2. Nodes move according to SRMM and received signal power($P_r$) is
         calculated according to Two ray propagation model as in [10]
    2.3. If an agent comes to this site/node ($MN_j$)
        2.3.1. If the agent is found to be suspected (authentication fails) then it is
               killed.
        2.3.2. Otherwise allow computation at this node.
            2.3.2.1. Update direct observation of this node according to the agent's
                     shared experience
                2.3.2.1.1. If a node is found to be trusted,$\alpha$ is incremented using to
                           eqn 3.
                2.3.2.1.2. Otherwise $\beta$ is updated according to eqn 4.
                2.3.2.1.3. Using eqns 2, 5 and 6 update yield values of $b_{ij}$, $d_{ij}$ and $u_{ij}$
                           for all j visited by the agent
    2.4. If an agent owned by this node comes back containing at most one
         suspected node in its PL then
        2.4.1. Update the results.
        2.4.2. Update direct observation of this node.
            2.4.2.1. If a node is found to be trusted $\alpha$ is incremented according to
                     eqn 3.
            2.4.2.2. Otherwise $\beta$ is updated according to eqn 4.
                2.4.2.2.1. Also *learn* to avoid the existing route followed by the
                           agents towards this node
            2.4.2.3. Using eqns 2, 5 and 6 update yield values of $b_{ij}$, $d_{ij}$ and $u_{ij}$ for all
                     j visited by the agent
        2.4.3. Kill the agent (Algorithm – I, steps 1.4 and 1.5).
    2.5. Whenever a message regarding suspected node id is received from a trusted
         node, then update the indirect observation according to eqns 10 through 12.
```

2.6. Hence update comprehensive (b,d,u) for visited nodes using eqn 13 through 17.
2.7. Compute the trust of this node for other nodes in the network following eqn 18.
2.8. If the resulting trust level of any node falls below Trust_thresholddemanded by the deployer application then advertise the node id to be a suspected one to the rest of the nodes.
2.9. The owners create PL for each agent containing trusted nodes ids.
2.10. Deploy the agents.
3. Stop.

In step1.4 of algorithm I if the current host platform (where the agent currently resides) is found to be malicious, then most likely the data part of the agent is changed(corrupted PL), not the code. So to save network bandwidth the agent can be asked to move back to its owner (so that the owner may update its trust level accordingly). Here we assume some means to authenticate an agent at a host site so that a host eventually detects a malicious agent. Agents in our system work as watchdogs [7]; they migrate and collect feedback about the trustworthiness of the nodes they visit. The reputation system at the nodes updates its view of the network based on the first hand and second hand information and accordingly guides (providing PL) the agents it deploys.

5 Experimental Results

The simulation is carried out in java and can run on any platform. MANET environment is simulated according to section 4. For simplicity, in our simulation the PL tells the agents which nodes to visit. The agent moves back to its owner at the end of its journey. We have done a series of experiments to show the robustness of our proposed algorithm. The default values for the experiments are shown in table-I. We will explicitly mention any change in these values for individual experiments. By step 1.5 of algorithm I, whenever an agent finds a suspected node, it comes back to its owner immediately. This strategy saves bandwidth but makes detection of other malicious nodes in the network a time consuming task. Hence the sharp slope of the curve (in figure 3) indicates that as more nodes in the network become compromised, the time to detect all of them increases even further. But if the agents are expected to visit more nodes (longer PLs) the probability of discovering a suspicious node increases. As more and more nodes gets visited, the direct experience becomes richer-which also improves the indirect and hence the overall trust convergence process (see figure 3). Thus for a more or less connected network, MANET becomes more secured with longer trails for the agents. Now the effect of direct and indirect observation on the reputation system is shown in figure 4. Initially indirect observation is not significant. But with increasing time (54 sec onwards) the difference became more apt. This explains the detection of suspicious nodes first from direct experience (by some nodes which), then broadcasting messages to others so that indirect experience can prove to be helpful to the rest. But in both cases the system eventually reaches a steady state.

Now we introduce a metric called the ratio of agents passed that is defined as follows

$$Ratio\ of AgentsPassed(t) = \frac{No.of agentsgoingthroughmaliciousnodestilltimet}{Totalno.of agentsdeployedtilltimet} \quad (19)$$

We take a case (see figure 5) where MN_3 behaved maliciously from the beginning of simulation but MN_{10} was compromised during the simulation. In the beginning performance varies as MN_{10} become malicious at different time instants. But it can be observed that eventually the ratio of agents passed becomes almost independent of the point when a node is compromised. This explains the robustness of our protocol.

Table 1. Default values of our configuration

Parameter	Default Values	Parameter	Default Values
M	20	Trust View default(b,d,u)	(0,0,1)
N	25	Trust_threshold	0.499
Length of priority list	0.5N	Minimum required P_r	18 dBm

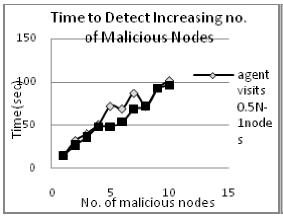

Fig. 3. System performance as threat to the agents increases

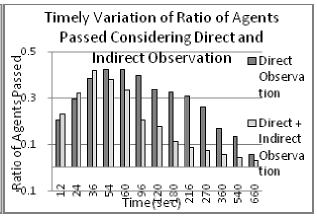

Fig. 4. Effect of direct and indirect observation in ratio of agents attracted

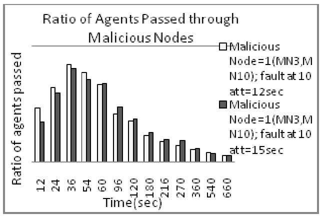

Fig. 5. Ratio of agents passed where MN_{10} is compromised in run time

6 Conclusions

This paper provides a trust based framework for securing the hosts and preventing the agents from visiting or passing through a compromised node in MANET. Possible modification in data is detected by taking hash code of an agent's data and code. Our model establishes trust among the nodes in a totally distributed manner and does not assume any central coordinator (for example a trusted third party). But the agent owners (nodes) are given the responsibility of killing malicious agents and creating new agents. If any node is found to be malicious, its entry gets removed from the PL of agents that are further deployed. The scheme enables an agent to share information about the network with the nodes it trusts. Trust is quantified using a tuple (b,d,u). For faster convergence of trust (consistent (b,d,u)s), the nodes share indirect information through broadcasts. This proves to be beneficial for nodes waiting for an agent response or particularly nodes which have not deployed agents at all. Also, to protect from malicious broadcasts, the nodes only listen to (and update trust level) broadcast messages from the senders they trust. SRMM is used to simulate the movement of the nodes. The protocol is validated and results are shown in section 5. It can be observed that for a larger MANET longer time will be necessary to detect all compromised nodes. In run time whenever a node becomes malicious, it is detected eventually and the system always reaches a steady state.

References

[1] Songsiri, S.: MTrust: A Reputation-Based Trust Model for a Mobile Agent System. In: Yang, L.T., Jin, H., Ma, J., Ungerer, T. (eds.) ATC 2006. LNCS, vol. 4158, pp. 374–385. Springer, Heidelberg (2006)

[2] Jøsang, A.: Trust-Based Decision Making for Electronic Transactions. In: Yngström, L., Svensson, T. (eds.) Proc. of the 4th Nordic Workshop on Secure Computer Systems (1999)

[3] Whitby, A., Jøsang, A., Indulka, J.: Filtering out unfair Ratings in Bayesian Reputation Systems. The Icfain Journal of Management Research 4(2), 48–64 (2005)

[4] Luke Teacy, W.T., Patel, J., Jennings, N.R., Luck, M.: Coping with Inaccurate reputation Sources: Experimental Analysis of a Probabilistic Trust Model. In: AAMAS 2005 (2005)

[5] Li, F., Wu, J.: Mobility reduces uncertainty in MANETs. In: Proc. of INFOCOM 2007, pp. 1946–1954 (2007)

[6] Shafer, G.: A Mathematical Theory of Evidence. Princeton University Press, Princeton (1976)

[7] Li, N., Das, S.K.: A trust-based framework for data forwarding in opportunistic networks. Ad Hoc Networks (in press)

[8] Yu, B., Singh, M.: Detecting Deception in Reputation Management. In: Proc. of the 2nd International Joint Conf. on Autonomous Agents and Multi Agent Systems, pp. 73–80 (2003)

[9] Lu, P., Li, B., Xing, M.L., Li, L.: D-S Theory-based Trust Model FIRE in Multi-agent Systems. In: The Proc of the 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, pp. 255–260 (2007)

[10] Chowdhury, C., Neogy, S.: Reliability Estimate of Mobile Agent Based System for QoS MANET Applications. In: The Annual Reliability and Availability Symposium, pp. 1–6 (2011)

[11] Bettstetter, C.: Smooth is better than sharp: a random mobility model for simulation of wireless networks. In: The Proc. of the Fourth ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, pp. 19–25 (2001)

[12] Meier, R.T., Dunkel, J., Kakuda, Y., Ohta, T.: Mobile agents for service discovery in ad hoc networks. In: Proc. 22nd International Conference on Advanced Information Networking and Applications, pp. 114–121 (2008)