International Journal of Network Security & Its Applications (IJNSA), Vol.3, No.6, November 2011

SECURING MOBILE AGENTS IN MANET AGAINST ATTACKS USING TRUST

ChandreyeeChowdhury *SarmisthaNeogy

Dept. of Computer Scienceand 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) especially against threats of the underlying environment where agents may get killed or rerouted by visiting hosts. The best way to defend against this situation is to prevent both the hosts and agents from communicating with the malicious ones. In this regard this paper develops a distributed reputation model of MANET using concepts from Dempster-Shafer theory. The agents (deployed for some purposes like service discovery) while roaming in the networkwork collaboratively with the hosts they visit to form a consistent trust view of MANET. An agent may exchange information about suspected nodes with a visiting host. To speed up convergence, information about an unknown node can be solicited from trusted neighborhood. Thus an inactive node, without deploying agents may also get a partial view of the network. The agents can 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 even in bigger networks.

KEYWORDS

Mobile Agent, Security, Hashcode, Trust, Dempster–Shafer Belief Theory

1. INTRODUCTION

Nowadays mobile agent seems to be a popular choice for designing applicationslike service discovery, network discovery, automatic network reconfiguration etc. for resource constrained environments like mobile adhoc networks (MANET).Many a time task processing is taken up by mobile agents that roam in the network and consequently get the task done.

But securing agents is a big concern particularly when the underlying network typically undergoes continuous topology changes thereby disrupting flow of information over the existing paths.As has been pointed out in [1] security of a mobile agent paradigm emphasizes on protecting and preventing a mobile agent from malicious hosts' attacks by applying cryptographic functions. Unfortunately these countermeasures become insufficientwhen the environment itself brings with it much vulnerability like blackhole[2], grayhole[2] or wormhole[2] attack. Commonly used routing protocols[2] cannot prevent such attacks. In such cases, the agents are either engulfed by a host (blackhole) or is forwarded elsewhere(wormhole). But in either case the agents will never be able to come back to its owner in due time.Thus an agent if happens to pass through such a host will effectively be lost.

DOI : 10.5121/ijnsa.2011.3620

Howeverpreventing a mobile agent from visiting a malicious node solves most of the risk factors. Consequently this technique not only protects the agents but also its owner that is the nodes in MANET. This point onwards, the terms node and host are used interchangeably unless otherwise stated.

Our threat model is as follows. We assume that the adversary can place malicious (wormhole/blackhole/grayhole) nodes at arbitraryplaces in the network, and that these nodes are connectedthrough a communication channel that cannot be observed by othernodes. These nodes either kill or mislead the agents in such a way that they (agents) never come back to their owners in time (within a specified time-out limit). The agents (by using a combination of hashcode and digital signature (as in [3]))can successfully detect any attempt of changing its code upon reaching at a host site. Thus the attacker does not need to know details of above mentioned techniques to fool the nodes to believe that their agents are lost due to adverse MANET conditions.We assume agents are deployed by some distributed applications like service discovery or clustering etc where graceful degradation in performance (as some nodes may become malicious) is acceptable. If an agent needs to visit a particular node in MANET (as in e-commerce) and that node is corrupted then obviously the task cannot be completed in our model, even.

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 istrusted, then it must be secure and vice-versa [4].

This observation leads us to consider security as a property of asystem in a given environment, and trust as a subjective belief resulting fromassessing a system and its environment. As in [1] we define trust as a subjective quantified predictor of the expected future behaviour of a trustee according to a specific agreement elicited from theoutcomes of the previous interactions, both from direct experiences and indirect experiences. Reputation of an individual host refers to certain characteristics related to its trustworthiness. Reputation can be obtained from a set of interaction feedbacks, in a mobile agent system; wheremobile agents describe a visited host's performance in fulfilling its obligations. Indirect experiences can also be considered which is gathered from other trustworthy nodes in the neighbourhood. To speed up convergence, the list of suspicious nodes may be shared among the nodes in MANET via the agents.

In this paper we describe a trust based framework for mobile agent based system in a dynamic and hostile MANET environment. We show how cooperative behaviour of the agents and nodes help to secure MANET and prevent an agent from getting trapped into a suspicious neighbourhood.Thus our definition of trust may range from complete belief to complete disbelief to full uncertainty as well as is described in section 3. This section also describes the basis of different types of observations done by the agents and consequently the nodes.

The next section(section 4) illustrates the way we model mobile agents on MANET in order to detect a malicious agent as well as a malicious platform (depending on trust level defined later) in a distributed way (using the reputation system designed in section 3).Section 5 gives the experimental resultsto show the robustness of our schemefollowed by concluding remarks (in section 6). In the following section (2), state of the art regarding this area of research is elaborated.

2. RELATED WORKS

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

2.1. Trust Management in MANET

Trust-based data routing has been extensively studied inwireless networks including MANETs [5][6][7][8]. The basic framework of a Trust Management System (TMS) includes a Reputation System (RS)and a Watchdog. Generally, the RS consists of reputationupdating through direct observation of the Watchdog (that is,first-hand information), reputation integration based on theindirect information from other members (i.e., second-handinformation), and reputation aging. The watchdogs normally monitor the event of data forwarding and count the arrival of ACKs corresponding to data sent out/forwarded. To cope with mobility, in [6] multiple feedbacks are compressed together. But using mobile agents for this purpose (which can already be deployed for functions like service discovery, clustering MANET etc.) will yield far better results as mobile agents are designed in such a way that they can easily cope with frequent disconnections and limited bandwidth characterizing MANET especially delay tolerant networks [6]. In [5] it is shown that mobility reduces uncertainty in trust calculation as it increases the chance of directly interacting with a node.

2.2. Trust in Mobile Agent Based Systems

Trust management system for mobile agents is also well studied [1] in literature. In [9] a distributed reputation management model is proposed that is based on Dempster-Shafer theory of evidence. This system solves some of the problems in e – bay‘s reputation model taking deceptive ratings into account. A trust model is described in [10] for multi-agent systems (MAS) that considers information collected from several sources (interaction trust, witness reputation, role based trust and certified reputation). It also usesDempster-Shafer theory of evidence. In [11], a mobile agent based reputation management system has been proposed for e-business environments. The system uses direct interactions and feedback from customers in a social network using agents, where each customer models trustworthiness of a vendor. The modelling of trustworthiness is done using Dempster -Shafer evidential theory, fuzzy logic. But this model does not fit into a resource constrained dynamic environment like MANET. In [1], a reputation-based trust model is proposed for mobile agents. Bayesian Network based trust computing is used and two algorithms are proposed for strategically malicious trustee prevention.

But most of 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. More importantly, assumption of a trusted third party or a trusted server and with 100% availability is practically not feasible in MANET. So the approaches based on a fully trusted node renders useless in resource constrained dynamic environments like MANET.

Although some works have been done to detect blackholeattack [12] or even wormhole attack[13] in MANET but we did not come across any work that studies its effect on agents in MANET or uses the agents to detect such traps.

3. TRUST MODEL

A reputation system [14] represents a promising method for fostering trust amongcomplete strangers and for helping each individual to adjust or update its degree oftrust towards its corresponding interaction partner and thereby reduce uncertainty.But a reputation system also suffers from threats like strategic rater [15] or strategically malicious host [14]. We attempt to address these threats by taking opinions from the agents and peers.

The main focus of our work is to study and prevent the nodes and agents (deployed by them) in a MANETfrom the effect of network layer attacks like blackhole[2] or wormhole[2] in which the affected agents do not come back to their owner.

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 trustworthy a host is) or disbelief (how much suspected a host is) as well as uncertainty in case both belief and disbelief are lacking. We express this mathematically [4] as:

$$b+d+u=1 \quad (1)$$

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 to ensure that an agent never falls into a blackhole, grayhole[2] or even wormhole trap. In addition, digital signature may be used to prevent or at least detect any attempt to change static code of the agent[16]. To quantify trust, parameters (b, d and u) are updated from direct observations (agent's experience at different nodes) and indirect observations (feedback from neighbouring nodes and others, collected via agents). Both observations are combined towards quantifying trust from (b, d and u). Aging is

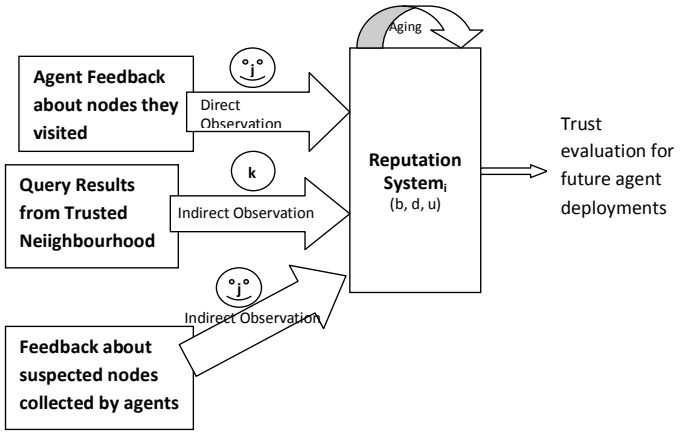

Figure1.Trust evaluation framework at hosts taking feedbacks

also considered in the process that accounts for network dynamicity.

3.1. Direct Observation

As mentioned in [16] each agent may carry the following

$SIGNATURE_{owner}$(code for hashcode computation+ static application code) + dynamic code(if any) + data

Here application code of agent refers to the purpose for which it is deployed by its owner. The signature helps to authenticate the code and checks trustworthiness of the host. The dynamic code part will be meaningful for strong migration [17] which is uncommon in MANET. 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][16].α_{ij}represents the number of good transactions between the agents deployed by $owner_i$and $node_j$. Thus for each positive feedback from agents, $\alpha_{ij}$is incremented as follows:

$$\alpha_{ij(new)} = \omega * \alpha_{ij(old)} + (1-\omega) * p_j^k \quad (2)$$

Here p_j^k represents $agent_k$'s observation about $node_j$. In this case weighted average is taken, where ω (0<ω<1) represents the absolute trust on each agent's observation as this observation may change from time to time taking care of network dynamicity.

Moreover, it may so happen that an agent successfully visits a number of hosts before falling into a trap. So, every node maintains last L owner ids that sent agents to this node. Thus if $agent_k$ while visiting $node_j$ finds its owner id in L (indicating some agent from the same owner has recently visited this node) then it further increments p_j^kin equation2 accordingly.

An agent may not come back to its owner in time (time-out) due to network latency or presence of wormhole or blackhole. But to detect the exact cause, the owner divides the task into n subtasks (value of n depends on network bandwidth) and deploys n agents. These agents are expected to come back faster and reveal more information about the network. Now if $anagent_k$ does not come back, the $owner_i$ increments β_{ij} as follows

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

Here j represents all nodes that the part $agent_k$ needs to visit in order to complete the subtask given to it. But β_{ij} may not reflect the exact scenario as a node can be strategically malicious [14]. 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 [18] 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 [6][3]:

$$u_{ij} = \frac{12 * \alpha_{ij} * \beta_{ij}}{(\alpha_{ij} + \beta_{ij})^2 * (1 + \alpha_{ij} + \beta_{ij})} \quad (4)$$

An agent while visiting a host site may also share and update its suspected list with the host. Any appended entry to the list will be considered as indirect observation at the agent owner. This is done to prevent a node from having any deceptive information.

The 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 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. Here u_{ij} is calculated using equation 4. Consequently following equation 1, the total certainty (= (1-u_{ij})) is divided into b_{ij} and d_{ij} according to their proportion of supporting evidence as follows (initial observation is based on[5]):

$$b_{ij}(\mathrm{t}) = \begin{cases} \frac{\propto_{ij}}{\propto_{ij}+\beta_{ij}}(1-u_{ij}) & \text{initially} \\ \frac{\frac{\propto_{ij}}{\propto_{ij}+\beta_{ij}}(1-u_{ij})*\omega_1 + b_{ij}(\mathrm{t}-\Delta\mathrm{t})*\omega_2}{\omega_1+\omega_2} & \text{otherwise} \end{cases} \quad (5)$$

$$d_{ij}(t) = \begin{cases} \frac{\beta_{ij}}{\propto_{ij}+\beta_{ij}}(1-u_{ij}) & \text{initially} \\ \frac{\frac{\beta_{ij}}{\propto_{ij}+\beta_{ij}}(1-u_{ij})*\omega_1 + d_{ij}(t-\Delta t)*\omega_2}{\omega_1+\omega_2} & \text{otherwise} \end{cases} \quad (6)$$

Averaging (weighted) is needed to reflect n part agents' behavior in the same tuple (b_{ij}, d_{ij}, u_{ij}).New observation is given a weight of ω_1 and old observation is given a weight of ω_2.Thus old values of b_{ij} and d_{ij} are given lesser weights($\omega_2 < \omega_1$) than recent values to represent aging.

In this way with the help of Dempster–Shafer Belief Theory [18] uncertainty can significantly be reduced even though perfect accuracy could not be achieved.

3.2. Indirect Observation

For faster convergence of trust, nodes share information about suspicious node/samong each other via the agents. 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 an agent suspects a node based on another (preferably trusted) node's observation without ever visiting that node.This second-hand information helps a node to cope with long delays and frequent partitions (formation of disconnected clusters) which are characteristics of MANET.

Let $b_l^{i:j}$represent 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) $node_j$'sbest possible final observation on $node_l$ as predicted by $node_i$as follows

b_j^1=TrustLimit(highest value of trust for a suspected node); d_j^1=(1-TrustLimit); u_j^1=1

Following the approaches proposed in [5] ($b_l^{i:j}$,$d_l^{i:j}$,$u_l^{i:j}$) can be formulated as

$$b_l^{i:j} = b_j^i \times b_l^j \quad (7)$$

$$d_l^{i:j} = b_j^i \times d_l^j \quad (8)$$

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

It can be noted that $node_i$'s disbelief in $node_j$'s observation becomes an uncertainty for predicting $node_l$[3]. Also $node_i$'s uncertainty on $node_j$ amounts to the uncertainty of $node_i$ in predicting $node_l$'s future behavior.

If $node_i$ enters a new network and gets trapped by its very neighbours then no agents deployed will come back. Then $node_i$ will prefer to wait till it moves. Moreover if a significant number of agents do not come back indicating wormhole or blackhole trap in transit, $node_i$ will prefer to request and collect information about suspicious nodes from its trusted neighborhood. The neighbors respond with their final observation (b_j^1, d_j^1, u_j^1) about the nodes (all ls) they suspect. This is updated in the same way using equations 7-9.

Thus a node predicts about the future behavior of a node taking indirect feedbacks from all agents in the last time interval (Δt) and updates its view (b, d, u) as follows [5]

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

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

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

Here $b_{i:l}$ represents the indirect belief of $node_i$ about $node_k$. S denotes the set of nodes that shared its view of the network (that $node_i$ received) with the agents deployed by $node_i$in the last time interval.

3.3. Combining Direct and Indirect Observation

After collecting first-hand and second-hand information from the agents/trusted neighborhood, 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 equations, as in [5]

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

$$d_j^{i(f)} = \varphi_1 \times d_{ij} + \varphi_2 \times d_{i:j} \quad (14)$$

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

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 [4][6]

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

Here σ gives relative atomicity based on the principle of indifference [3]. We have taken σ to be 0.5 indicating that among the total uncertainty associated with an agent's visit, there is a 50% probability that the agent will be safe. But we can tune this parameter more accurately meaning, 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 equation (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.

4. IMPLEMENTING OUR MODEL ON MANET

In this paper, we define our mobile agent-based system (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 abstraction of an ad hoc network as in [16]. The nodes move according to Smooth Random Mobility Model [16] and two ray propagation [17] of radio signals is assumed while checking for link existence. Here we try to protect mobile agents from visiting malicious hosts (nodes) and to 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. Also a compromised node may work as a black hole tovisitoragents.

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.

An important use of mobile agents is to collect data from a network like service discovery [19] or clustering in MANET [20] or ecommerce applications [21], etc. An agent starts its journey from a given owner and moves from one node to another at its will. The owner provides a *Priority List* to the agent which contains a list of node ids that are most beneficial migration sites (for the application that deployed that particular agent). A *Suspicious Node List* is also given that indicates potential blackhole or wormhole points.Reachinga trusted site an agent shares and updates its knowledge about suspicious nodes. So, an agent will always try to visit nodes from (*Priority List*–updated *Suspicious Node List*) set. But this movement is successful if the two nodes are connected according to equation (24) and there is no simultaneous transmission in the neighborhood of the intended destination (taken care of by the MAC protocol). Thus, an agent residing at node MN_A moves to node MN_B (connected to MN_A) with probability p_t.

We describe the security model as follows.

4.1. Detailed Algorithm

The following data structures are needed

- **Priority_list of agent** j: PL:-it has two fields- node_id and trust_level (unvisited 0; suspected -1; trusted +1; recent visit by an agent of same owner +2)
- **Suspected node list for agent j**: SL:-two fields-node id and optional provider id if not given by the owner of agent j
- **β**: a positive integer to be kept at node
- **α**: a positive integer to be kept at node
- **Default trust level** : TS (> k(otherwise, a node becomes suspicious))
- **Trust level view of the MANET by 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 the current node i according to equation (18)
- $\mathbf{C_{agent\text{-}id}}$: number of part agents sent for the lost agent designated by 'agent-id'
- **X:**maximum number of malicious nodes in the network
- $\mathbf{T_{agent\text{-}id}}$**:** maximum time an agent can be enroute

Initially the priority lists of all agents have 0 trust level corresponding to every node id in their priority list (PL). So, node i's view of the network will be $(TS,TS,\ldots\ldots)_i$.

The workflow can be divided into two parts: (i) Computation/Action in mobile node and (ii) functions of the agents.

Algorithm I describesthe function of the agents collecting first handinformation about an agent's trust and second hand information about the nodes whom the hosts (visited by the agent) suspect.

Algorithm II , an evolutionary algorithm based on Monte Carlo simulation, is running at the nodes that takes its input from algorithm I and any message received from trusted neighbors (second hand information) to update the distributed trust model and hence the node's trust level view of the network. This in turn affects the route taken by newer agents.

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 priority list provided
     - 1.1.1. Check if the next node to be visited falls in the appended suspected nodes list
   - 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[16]
   - 1.3. Before processing, as in [16][3] hashcode can be used to detect any attempt to change agent's code/data by the node
     - 1.3.1. Gather information needed by the application that deployed this agent
     - 1.3.2. Update computed results
     - *1.3.3.* Hash code should also be computed to take care of updated data
     - 1.3.4. Share and update its suspected node list (if any) provided by the owner with this host
       - 1.3.4.1. Appended entry (if any) will be marked by the id of this host.
   - 1.4. Else 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 (initial position, speed of the nodes)
2. For $t=t_0$ to T repeat the following
   - 2.1. Some nodes may fail following Weibull distribution and others move according to SRMM and as a result a new edge list, E' is formed as in [16][17].
   - 2.2. If an agent comes to this node (MN_j)
     - 2.2.1. If the agent is found to be suspected (authentication fails or it comes from a suspected node) then
       - 2.2.1.1. Kill that agent
     - 2.2.2. Otherwise allow computation at this node
     - 2.2.3. Looking at the suspected node list of this agent, this node updates its indirect observation using equations 7 through 12 depending on how much the node trusts this visiting agent's owner

2.2.4. Also the node shares it's (if nonempty) suspected node list with the current visitor.

2.3. If an agent owned by this node comes back containing at most one suspected node in its PL then

2.3.1. Call Update_Trust().

2.4. If an agent does not come back and time out occurs,

2.4.1. Divide the job of that agent into n parts and spawn n agents which carry n priority sub lists.

2.4.2. Start a timer ($T_{agentid}$) for these n new agents to indicate that it is arepeat attempt.

2.4.3. The agent ids are also given in such a way to indicate each one as (1/n)th part of 1 task-that of the lost agent.

2.4.4. Set $C_{agent\text{-}id}$to n.

2.5. If a part agent comes back, decrease $C_{agent\text{-}id}$ by one.

2.5.1. Call Update_Trust() method.

2.6. If a $T_{agentid}$expires, find its corresponding $C_{agentid}$.

2.6.1. If $C_{agentid}$>X then deploy the lost agents again asking them this time to follow different route.//this algorithm can tolerate maximum X suspicious nodes

2.6.2. Else if 0<$C_{agentid}$<X then ask recommendation from trusted neighborhood regarding the suspected nodes (mentioned in the priority sub lists of $C_{agentid}$lost agents).

2.7. Receive information from trusted neighborhood about other nodes and update the indirect observation following equations 7 through 12.

2.8. Hence update comprehensive (b,d,u) for the nodes visited using equations 13 through 17.

2.9. Compute how much this node trusts others in the network following equation 18.

2.10. If the resulting trust level of any node falls below Trust_threshold demanded by the deployer application, then append the node id to the suspected node list.

2.11. The PL for each agent containing trusted node ids is also formed and kept with the owners.

2.12. Deploy the agents.

2.12.1. Equip the agents with the suspected node list (what to avoid) and a priority list (what to follow).

3. Stop.

Update_Trust()

1. Uptate the results
2. Update direct observation of this node
   2.1. If a node is found to be trusted, α is incremented according to equation 3
   2.2. Otherwise β is updated according to equation 4
      2.2.1. Also *learn* to avoid the existing route followed by the agents towards this node
   2.3. Using equations 2, 5 and 6 update yield values of b_{ij}, d_{ij} and u_{ij} for all j visited by the agent
3. Update indirect observation of this node.
   3.1. If any new entry is found in the suspected node list then

3.1.1. Update this information depending on how much the owner trusts the information provider according to equations 7 through 12

4. Kill the agent (Algorithm – I, steps 1.4 and 1.5)
5. Return

Step 1.3 of algorithm-I is optional, it isneeded to protect the priority list (kept as part of agent's data) from corruption. Individual node failures are considered in step 2.1 of MN_code(). But we did not consider the fault tolerance of the nodes. Here a node failure is treated to be irrecoverable. It may be pointed out that step 1.4 of Agent_code() actually corresponds to step 2.4 of MN_code(). Here we assume that a host eventually detects a malicious agent. Creation of part agents will be continued unless all of them come back or decision can be made about the nodes found in the priority lists of missing agents.

As can be seen, agents in our system migrate and collect feedback about the trustworthiness of the nodes they visit. So, they work like watchdogs[6]. The reputation system at the nodes based on the first hand and second hand information updates its view of the network and accordingly guides (providing priority list and suspected node list) the agents it deploys.

5. EXPERIMENTAL RESULTS

The simulation is carried out in Java and can run in any platform. For simplicity, in our simulation the PL tells the agents which nodes to visit. After visiting all the nodes from the PL

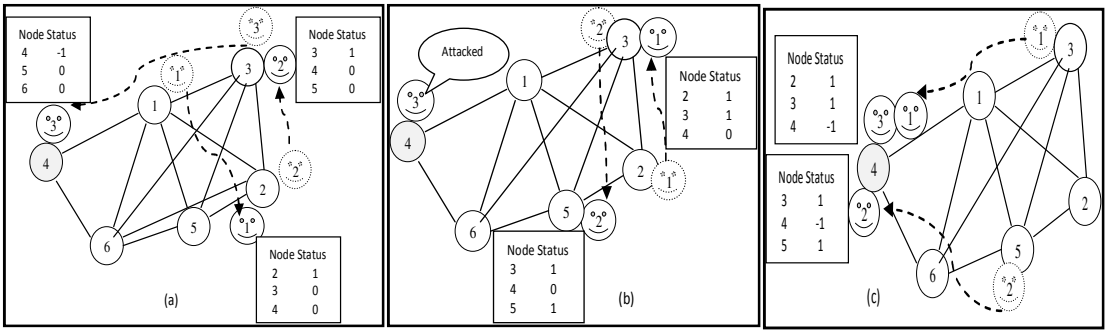

Figure 2(a) Network graph at time instant t=t0 and the position of the agents
(b) Network graph at time instant t=t0+Δt and the position of the agents
(c) Network graph at time instant t=t0+2Δt and the position of the agents

Table 1. Default values of our configuration

Parameter	Default Values	Parameter	Default Values
Mobility Model	SRMM	Length of priority list	10
Time	80 min	Trust View default(b,d,u)	(0,0,1)
N	25	Trust threshold(k)	0.49

successfully, the agent moves back to its owner. We have taken an instance where there are 6 nodes and 3 agents in the network. The connectivity graph, agents and their corresponding PL are shown in figure 2. Due to smooth movement of the nodes (according to SRMM) no drastic change can be observed in the connectivity graph in subsequent time instants. In our example MN_4 is treated as a malicious node that can launch routing attack that prevents visitor agents from coming back to their owners. As can be observed agents 1, 2 and 3 eventually get stuck at MN_4.Thus according to step 2.4 of MN_Code() time out (6*average propagation delay) occurs and the owners MN_1, MN_2 and MN_3 spawn agents 1 and 4, 2 and 5, 3 and 6 respectively with lesser number of nodes in PLs. Here we subdivide only one PL into two unequal parts and spawn two agents accordingly. The division is carried out according to a factor that is initialized to 0.5 but is decreased by 0.15 in each iteration till it reaches 0.2(in an order to group all suspicious nodes in one sublist). Now nodes with $(d_i^{j(f)}-b_i^{j(f)})> \varepsilon(=0.0028)$ are put in the smaller sublist. Thus agent 1 now needs to visit MN_2 and MN_3 while agent 4 visits MN_4. Moreover,while visiting MN_2 and MN_3, agent 1 finds that some agent from the same owner (MN_1) has recently visited these nodes. This observation is reflected in the status (=2 instead of 1) of agent 1's PL. Clearly this time direct observation (b_{14}=0.2, d_{14}=0.11, u_{14}=0.7) of agent 1 gets reflected in the final observation($b_1^{4(f)}$=0.05298,$d_1^{4(f)}$=0.02796, $u_1^{4(f)}$=0.91906)of its owner's (MN_1) trust view. This process goes on. While updating direct observation in equation 5 and 6, for simplicity (major change is not expected in simulation time=80min) old and new values are given equal weights. As soon as trust view of any node goes below 0.49, that node is declared to be malicious and is appended in the suspected list of agents (removed from its PL as well) spawned by the detector node. Thus the nodes try to overcome routing attacks without any overhead of control messages.

We have done a series of experiments to validate our algorithm and found some interesting results. For simplicity whenever an agent goes missing, 2 agents are spawned as is explained in the example. The simulation parameters are summarized in table-I.Any change to it is explicitly mentioned. We introduce a metric called the ratio of agents attracted during time period t that is defined as follows

$$Ratio of Agents Attracted(t) = \frac{No. of agents going through malicious nodes till time t}{Total no. of agents deployed till time t} \quad (19)$$

Experiments are done to show timely variation of this metric while the MANET has 3 malicious nodes (MN_3, MN_4 and MN_5) and 5 malicious nodes(MN_3, MN_4, MN_5, $MN_{11}$and MN_{13}). Results plotted in figure 3 clearly show that agents gradually overcome network hostility. This is evident from the steady slope of the curve especially after T=8min. As number of malicious nodes increases more agents are affected but eventually the agents detect them by the trust calculation. Since the curve well stabilizes at around 80 min with 3 malicious nodes, simulation time is kept at 80min in our experiments.

Also we show variation of the ratio (equation 19) with no. of nodes (N) in figure 4 while MN_3 and MN_4 launch blackhole/wormhole attack.It can be observed that as the network gets bigger, the ratio gradually declines (due to increased amount of indirect observation) and eventually (N=35 onwards) reaches a steady state.Also more number of agents (M=20) implies richer direct observation resulting in even faster convergence of trust. Arrival of steady state for both M=10 and 20 indicates the scalability of our scheme with moderate accuracy.

Another metric named ratio of successful agents is defined as follows

$$Ratioof\ successful\ Agents(t) = \frac{No.ofagentscame\ back\ to\ ownertilltimet}{Totalno.ofagentsdeployedtilltimet} \quad (20)$$

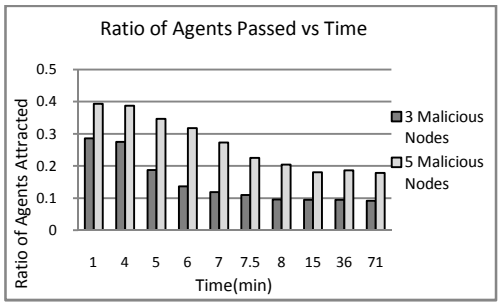

Fig. 3 Timely Variation of ratio of agents attracted by the malicious nodes

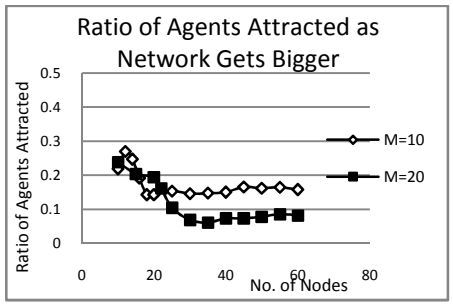

Fig. 4 Variation of ratio of agents attracted by the malicious nodes as network gets bigger

All agents deployed by an owner may not come back within stipulated time as some of them may be rerouted by malicious nodes (wormhole), engulfed by them (blackhole) or lost due to network partitioning. Agent code/data may get modified also that can be detected by the owner (by generating and checking hashcode[16]). Considering MN_3 and MN_4 to be malicious nodes, the effect of MANET size on agent success is found Momentary drop in agent success can be observed when M and N values are almost comparable in figure 5. But as MANET becomes bigger, agents manage to provide a steady success rate. Both figures 4 and 5 confirm the fact that bigger networks are not detrimental for agent success if the level of hostility remains same.

The next experiment again introduces another metric called the node success ratio defined as

$$Node\ success\ ratio(t) = \frac{No.ofNodes\ that\ can\ prevent\ their\ agents\ from\ routing\ attackstilltimet}{Totalno.ofnodes\ working\ tilltimet} \quad (21)$$

The dependence of successful detection and subsequent deletion of malicious nodes from PL by any node on the number of agents they deploy is indicated in figure 6. It is seen that with 50 agents, up to 3 malicious nodes can be successfully detected within 80 minutes and no nodes in that case will be sending their agents to MN_3, MN_4 or MN_5. Also it can be observed that all curves reach a local maxima when number of agents is approximately equal to number of nodes (=25). This is because at this point all nodes get the direct observation from agents, that is, agents tend to cover the entire network.

In the next experiment our model is tested with increasing MAS size. We define a metric called ratio of false negatives that is defined as follows:

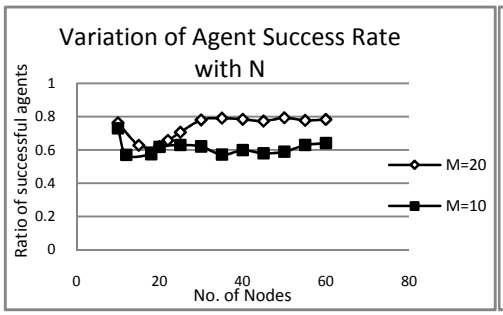

Figure 5 Variation of agent success rate with no. ofnodes(N)

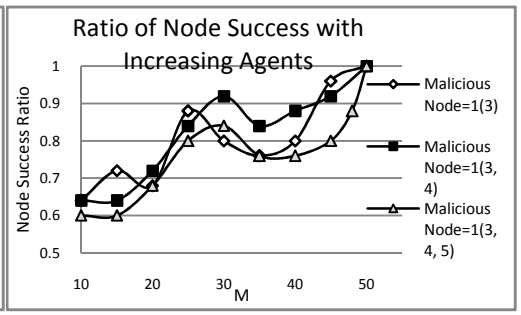

Figure. 6 Variation of node's success ratio with no. of agents(M)

$$Ratio\ of\ False\ Negatives = \frac{No.of\ undetected\ malicious\ nodes}{Total\ No.of\ malicious\ nodes} \quad (22)$$

Let us presume we know the total number of malicious nodes in the network at a time instant. So it is checked if our algorithm running at the nodes can successfully detect all the malicious nodes. The results show(figure 7) that as more nodes participate for some job and hence deploy agents (which in turn also gains direct experience) to traverse various parts of the network more malicious nodes are eventually detected. Thus for greater MAS size the ratio ultimately drops to 0 indicating successful detection of all malicious nodes. For bigger network more agents are needed to achieve the same value of false negatives hence needing more bandwidth. Interestingly with M=2*N, the ratio of false negative hits 0. It also portrays correctness of our algorithm as all malicious nodes can be detected by deploying sufficient no. of agents.

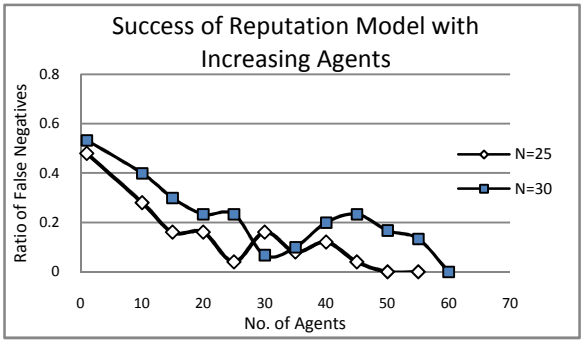

Figure.7Success of the reputation model proposed in detecting malicious nodes

6. CONCLUSION

This paper provides a trust based framework for securing the hosts and preventing the agents from visiting or passing through a compromised node specially a blackhole/wormhole trap (from where the agents won't make a successful return in time) 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 without any central coordinator (for example a trusted third party). If an agent does not come back, new agents with smaller PLs are spawned to get better visibility of the MANET. Perfect detection of malicious nodes relies on how minutely the owners divide the PL of missing agents (and give it to new agents). Direct observations of the agents that come back play a very important role in the detection process. If any node is found to be malicious, its entry gets removed from the PL and appended in the suspected list of agents that are further deployed. The scheme enables an agent to share information with others about suspicious nodes, thus helping in faster convergence of trust. Hence nodes visited by an agent can know about MANET hostilities without deploying agents. Trust is quantified using a tuple (b,d,u). For faster convergence of trust (consistent (b,d,u)s), newer nodes may ask for indirect information from trusted neighborhood. 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 according to our scheme even for larger MANET, nodes can detect all the malicious nodes and eventually prevent themselves and their agents from network hostilities.

REFERENCE

[1] S. Songsiri, "MTrust: A Reputation-Based Trust Model for a Mobile Agent System", In the Proc. of the Third international conference on Autonomic and trusted computing, vol. 4158, pp. 374–85, 2006.

[2] L. Qian, N. Song, X. Li, "Detection of wormhole attacks in multi-path routed wireless ad hoc networks: a statistical analysis approach", Journal of Network and Computer Application. 30(1), pp.308-330, January, 2007.

[3] C. Chowdhury, S. Neogy,"Mobile Agent Security in MANET using Reputation", Proc. 1st International Conference on Parallel, Distributed Computing Technologies and Applications (PDCTA 2011) , pp.158-168, 2011.

[4] A. Jøsang. "Trust-Based Decision Making for Electronic Transactions", In L. Yngstr¨om and T. Svensson, editors, Proc. of the 4th Nordic Workshop on Secure Computer Systems (NORDSEC'99). 1999.

[5] F. Li, J. Wu, "Mobility reduces uncertainty in MANETs", in the Proc. of INFOCOM'07, pp. 1946–1954, 2007.

[6] N. Li, S. K. Das, "A trust-based framework for data forwarding in opportunistic networks", Ad Hoc Networks, Elsevier, in press.

[7] T. Anantvalee, J. Wu. "Reputation-based system for encouraging the cooperation of nodes in mobile adhoc networks", In Proc. of ICC'07, pp. 3383–3388, 2007.

[8] V. Balakrishnan, V. Varadharajan, P. Lucs, U. K. Tupakula. "Trust enhanced secure mobile ad hoc network routing", In Proc. of AINAW'07, pp. 27–33, 2007.

[9] B Yu, M Singh,"Detecting Deception in Reputation Management", Proc. of the Second International Joint Conference on Autonomous Agents and Multi Agent Systems, ACM Press, Melbourne, Australia, pp.73-80, 2003.

[10] P.Lu, B. Li, M.Xing and L. Li,"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.

[11] E.Sathiyamoorthy, N.Ch.S.NIyengar, V.Ramachandran, "Mobile Agent Based Trust Management Framework using Fuzzy Logic in B2C E-Business Environment", in the International Journal of Computer Theory and Engineering, Vol. 2, No. 2 , pp. 308-312, April 2010.

[12] S. Kurosawa, H. Nakayama1, N. Kato1, A. Jamalipour., and Y. Nemoto,"Detecting Blackhole Attack on AODV-Based Mobile Ad Hoc Networks by DynamicLearningMethod,"International J. Network Security, Vol.5, No.3, pp.338–346, Nov. 2007.

[13] Xiaomeng Ban, RikSarkar, JieGao, "Local Connectivity Tests to Identify Wormholes in Wireless Networks", in Proc. of the 12th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'11), May, 2011.

[14] A. Whitby, A.Jøsang, J. Indulka. "Filtering out unfair Ratings in Bayesian ReputationSystems".The Icfain Journal of Management Research, 4(2), pp.48-64, February 2005.

[15] W.T. Luke Teacy, J. Patel, N. R. Jennings, M. Luck. "Coping with Inaccurate reputation Sources: Experimental Analysis of A Probabilistic Trust Model". AAMAS 2005.

[16] C. Chowdhury, S. Neogy,"Mobile Agent Security based on Trust Model in MANET", Proc. 1st International Conference on Advances in Computing and Communication (ACC 2011) 2011.

[17] C. Chowdhury, S. Neogy, "Reliability Estimate of Mobile Agent Based System for QoS MANET Applications", in the Annual Reliability and Availability Symposium, pp.1-6, 2011.

[18] G. Shafer, A Mathematical Theory of Evidence, Princeton University Press, Princeton, NJ, 1976.

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

[20] M. K. Denko, "The use of mobile agents for clustering in mobile ad hoc networks", in the Proc. of the 2003 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology, pp.241-247, 2003.

[21] M.A. Tarig, "Using secure-image mechanism to protect mobile agent against malicious host", Proc. of World Academy and Science, Engineering and Technology (WASET), pp. 439–444, 2009.

Authors

SarmisthaNeogyis faculty in Jadavpur University at present and is in teaching profession since last eighteen years. She has been an active researcher in the areas of distributed systems, fault tolerance, mobile computing and security in wireless networks.

ChandreyeeChowdhury is a junior faculty in the department of Computer Science and Engineering at Jadavpur University. She received M. E in Computer Science and Engineering from Jadavpur University in 2005. Currently she is pursuing Ph. D under the guidance of Dr. Neogy. Her research interests include reliability and security in wireless networks.