Secure Leader Election Algorithm Optimized for Power Saving Using Mobile Agents for Intrusion Detection in MANET

Monika Darji[1] and Bhushan Trivedi[2]

[1] L.J Institute of Computer Application, Ahmedabad, India
monikadarji79@gmail.com
[2] GLS Institute of Computer Technology, Ahmedabad, India
bhtrivedi@yahoo.com

Abstract. In this paper we reduce the communication and computation overhead involved in Leader Election for running Intrusion Detection System (IDS) for a cluster of nodes in mobile ad hoc networks using Mobile Agents. To reduce the performance overhead of IDS, a leader node is elected to handle intrusion detection service for the whole cluster. If the cluster head selection is random, it might be possible that resource consumption of the leader is higher and it might just lose all its power. Thus overall lifetime of cluster reduces to lesser than useful lifetime. It is clear that such methods have no guarantee to work. The solution requires balancing the resource consumption among all the nodes and thus increase the overall lifetime of a cluster by electing the most cost- efficient node. We will call it leader-IDS. Our contribution to the election scheme is to design an algorithm which uses mobile agents to gather resource information from all the cluster nodes and use it to elect leader in optimum and secure manner.

Keywords: MANET security, Intrusion detection system, Mobile Agents, Cluster Head Election.

1 Introduction

Mobile Ad hoc NETwork (MANET) [1] is a set of mobile devices like laptops, PDAs, smart phones which communicate with each other over wireless links without a predefined infrastructure or a central authority. The member nodes are themselves responsible for the creation, operation and maintenance of the network using single hop or multi hop communication. The characteristics of MANET includes dynamic topology, lack of fixed infrastructure, vulnerability of nodes and communication channel, lack of traffic concentration points, limited power, less and varying computational capacity, less memory, and limited bandwidth which makes the task of achieving a secure and reliable communication more difficult.

MANETS have no centralized checkpoint where Intrusion Detection System can be deployed. Earlier proposed solutions had each node run IDS to perform local intrusion detection and cooperate with other nodes to perform global intrusion detection [2]. This scheme was inefficient in terms of resource consumption therefore cluster-based detection scheme were proposed [3][4].For Clustering, the nodes in MANET are divided into a set of 1-hop clusters where each node belongs to at least one cluster and a leader node (Cluster Head) is elected to run the IDS for the entire cluster. The leader-IDS election process can be random [3] or decided on the connectivity [4]. The resource availability of different nodes is different which must be considered by an election scheme otherwise some node's batteries dip will faster than others, leading to a loss in connectivity and reduction in overall lifetime of a cluster when these nodes become unavailable due to lack of power. Moreover if a shellfish node is elected then it may impede to run IDS and put the entire cluster at risk. To balance the resource consumption of IDSs among nodes and increase lifetime of a cluster, nodes with the most remaining resources should be elected as the leaders [5][6][7]. But resource information being private information, shellfish nodes may not reveal their remaining resources, during an election of the leader node to conserve their resource and vent other's. To balance the resource consumption among all the nodes and increase the overall lifetime of a cluster a mechanism is given using Vickrey, Clarke, and Groves (VCG) [8] and mechanism design theory in [5][6] for leader election.

While these schemes provide a solution for election of leader-IDS in presence of shellfish nodes, [5] and [7] do not deal with security issues, [6] provides a secure communication using public key infrastructure but all the nodes need to sign their information before sending and also needs to verify other nodes signature before computing costs which increases overhead. Moreover all the above schemes require broadcasts by all the nodes. We propose to use mobile agents to elect leader-IDS. Mobile agents are light weight; computationally efficient, flexible autonomous program agents can halt and ship themselves to another node on the network, and continue execution at the new node. An agent doesn't restart execution from the beginning at the new node; it continues where it left off.

For communication MANET over wireless links, mobile agents can be used for their efficiency in lightweight computation and suitability in cooperative Cluster Head computation. Mobile agents have been used in several techniques for intrusion detection systems in MANETs [10]. Due to its capability to travel through the large network, they can interact and cooperate with nodes, collect information, and perform tasks assigned to them. Opposed to traditional approaches where large amounts of data are transported towards the computation location, mobile agents allows the analysis programs to move closer to the audit data. Mobile agents can reduce the amount of communication through the network and is extensively used for distributed applications.

We have designed new algorithm that uses mobile agents to gather resource information from all the cluster nodes and use it to elect leader-IDS in optimum manner. Our scheme is inspired from Ajanta Systems [11] to design mechanisms that allow a mobile agent's owner to ensure security of an agent using three techniques: read only state, append only logs and selective revealing of the state. This work will be able to reduce the battery and bandwidth consumption for leader-IDS election and will be greatly useful for scenarios where mobility of nodes is higher. We justify the correctness of our proposed scheme through analysis.

The rest of this paper is organized as follows: Section 2 reviews related work. Section 3 illustrates our approach. Section 4 illustrates the Leader election scheme without Mobile Agents. Section 5 devises the Leader Election Process with Mobile Agents. Finally, Section 6 concludes the paper and discusses future work.

2 Related Works

The first distributed and cooperative IDS was proposed in [2] which used statistical anomaly detection, this model was extended in [3] where the job of intrusion detection was carried out by a Cluster Head selected in random fashion where each node is equally likely to be elected regardless of its remaining resources. In [4], elects a node with a high degree of connectivity even though the node may have little resources left.

In [5] a unified framework is proposed for balancing the resource consumption among all the nodes and for electing the most cost-efficient node known as leader-IDS. A mechanism is designed using Vickrey, Clarke, and Groves (VCG) to achieve the desired goal. To monitor the behavior of the leader and to catch and punish a misbehaving leader, they have made use of checkers that interact among each other using cooperative game-theoretic model to reduce the false-positive rate. A multi-stage catch mechanism is also introduced to reduce the performance overhead of checkers.

In [7] leader election scheme is proposed for electing most energy remaining node using QA-VCG mechanism by simplifying it to increase the survival time of nodes and to motivate the nodes to behave honestly and reveal their true cost of analysis by giving them incentives. The mechanism derives from QA-VCG an efficient multi-attributes procurement combinatorial auction model.
In [6] which is an extension to [5] leader election in presence of shellfish nodes. To address the selfish behavior, incentives are given in the form of reputation to encourage nodes to honestly participate in the election scheme by revealing of their available resources. The design of incentives is based on a classical mechanism design model, namely, Vickrey, Clarke, and Groves (VCG) [8] which guarantees that truth-telling is always the dominant strategy for every node during each election phase. They propose a series of local election algorithm that can lead to globally optimal election results with low cost. This scheme uses public key infrastructure for secure communication.

Limitation of existing scheme is the number of broadcasts and signature verifications that needs to be performed on all the nodes for gathering resource information to ensure secure communication. We propose a scheme where Mobile agents are used to gather the resource information from the nodes to avoid the broadcasts and signature verifications by each individual node. Instead of all the nodes broadcasting their resource information and vote, one of the nodes creates mobile agent to perform the task in an efficient manner by reducing the battery and bandwidth consumption.

Whenever the agent moves to a node, the node should not be allowed to steal the agent's information. Also the agent should be protected against modifications. The agent also requires protection against attacks from a hostile host which can be maliciously destroying an agent or tempering with an agent such that it may attack its own initiator. Unfortunately fully protecting an agent against all kind of attacks is impossible (Farmer et al., 1996) and alternative approach is to organize the agents in a way that modifications can be detected. This approach has been followed in Ajanta Systems [11]. Ajanta provides three mechanisms that allow an agent's owner to detect that an agent has been tampered with: read only state, append only logs and selective revealing of the state. These three mechanisms has inspired us to device a scheme for secure Cluster Head Election in MANET.

In our work for leader-IDS election, we propose to modify the algorithms in [6] by introducing mobile agents that will be used for gathering resource information and votes in a secure manner and thus reduce the overall battery and bandwidth consumption leading to efficiency and power saving.

3 Our Approach

MANET can modeled as undirected graph G = (N, L) where N is the set of mobile nodes and L is the set of bidirectional links. Using Cluster-first approach [12] clusters are formed in a network, and then, the nodes belonging to that cluster elect a leader node. Every cluster has a set of nodes n N and a set of links l L. One-hop neighbor nodes form a cluster and nodes might belong to more than one cluster. It is assumed that each node has an IDS and a unique identity. Nodes can overhear each other using omnidirectional antenna.

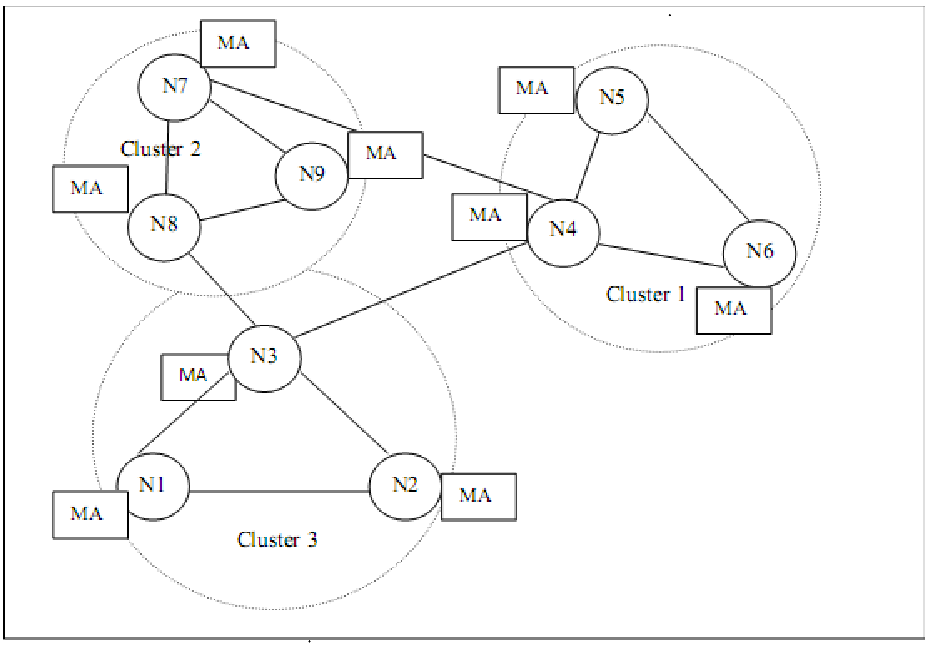

Fig. 1. An example scenario of leader election in MANET

Cluster head election is time consuming and resource intensive and requires broadcast and signature verification by every participating node; we propose to optimize it using mobile agents using our algorithm for leader election which is described below. We have extended the algorithm described in [6] with the introduction of mobile agents. Figure 1 shows a MANET composed of 9 nodes labeled from N1 to N9. These nodes are located in three 1-hop cluster nodes with varying resource and energy level. We assume the nodes N3 and N4 have low energy level. With Random Election Model [3], each node will have equal probability of being elected as leader to run the IDS, if nodes N3 and N4 are elected, they will die faster leading to partitioning in the network. Under Connectivity Index based approach [4], N3 and N4 will score higher and get elected leading to similar circumstances. Moreover, if shellfish nodes are elected as leaders, they will decline to do their job of running IDS. The scheme presented in [6] balances the resource consumption of IDS among all nodes and also prevents shellfish behavior by giving incentives in the form of reputation. We extend the scheme presented in [6] by using Mobile Agents which will move from one node to another in the cluster for gathering resource information and vote information and Cluster Head computation.

4 Leader Election Scheme without Mobile Agent

Leader election scheme without using mobile agents is presented in [5], [6] and [7].

4.1 Requirements of Designing Leader Election Algorithm as Described in [6]

1) To protect all the nodes in a network, every node should be monitored by a leader
2) To balance the resource consumption of IDS service, the overall cost of analysis for protecting the whole network should be minimized.

4.2 Assumptions

1) Every node knows its (2-hop) neighbors
2) Loosely synchronized clocks are available between nodes.
3) Each node has a key (public and private) pair for establishing a secure communication between nodes.
4) Each node is aware of the presence of a new node or removal of a node.

4.3 To Start a New Election, the Election Algorithm Uses Four Types of Messages Which Are as Follows

Hello: used by any node k to initiate the election process, Hello (ID_k,Hash($cost_k$))
Begin-Election: used to announce the cost of a node Begin-Election, (ID_k,$cost_k$)
Vote: sent by every node to elect a leader, Vote(ID_k)
Acknowledge: sent by the leader to broadcast its payment, and also as a confirmation of its leadership.

4.4 For Describing the Algorithm, Following Notations Are Used [6]

service-table(k): The list of all ordinary nodes, those voted for the leader node k.
reputation-table(k): The reputation table of node k. Each node keeps the record of reputation of all other nodes.
neighbors(k): The set of node k's neighbors.
leadernode(k): The ID of node k's leader. If node k is running its own IDS, then the variable contains k.
leader(k): A boolean variable that sets to TRUE if node k is a leader and FALSE otherwise.

Our election scheme is similar to that given in [6] but we are using mobile agents.

5 Leader Election Using Mobile Agent

Any node k which belongs to a cluster, initiates the election process by creating a Hello Agent contains Hello Message table whose format is shown in Table 1:

Table 1. Hello Message Table

Node Unique ID	Hash of Nodes Cost*	Signed MD	Checksum
ID_k	Hash($cost_k$)	E(PR_k,ID_k + Hash($cost_k$))	C_{init}

Explanation of Hello Massage Table of Table 1 is given below:
*Cost implies a node's cost of analysis
ID_k: node k's unique identifier
Hash($cost_k$): hash value of the node k's cost of analysis
Hash is needed to ensure non-repudiation (avoid cheating) as all the nodes before knowing others cost values will have to declare their hash of cost and later on cannot change it.
E(PR_k,ID_k + Hash($cost_k$)): The ID_k+Hash($cost_k$) is the message digest signed by node k.
This provides read-only state, authentication and integrity. When the agent arrives at another node, the node can easily detect whether read only state has been tampered with.
Cinit =E(PU_k ,N): PU_k is the public key of the agent's owner node k and N is the secret nonce known only to the node k.

When the agent arrives at a node l, it appends its ID and hash value of cost and new checksum in the Hello Message table as shown in Table 2.

Table 2. Hello Message Table with node k and l's data

Node Unique ID	Hash of Nodes Cost	Signed MD	Checksum
IDk	Hash($cost_k$)	E(PR_k,ID_k + Hash($cost_k$))	C_{init}
ID_l	Hash($cost_l$)	E(PR_l, ID_l + Hash($cost_l$))	C_{new}

Explanation of Hello Massage Table of Table 2 is given below:
ID_l: nodes unique identifier
Hash($cost_l$): hash value of the node's cost of analysis
E(PR_l, ID_l + Hash($cost_l$)): The ID_l+Hash($cost_l$) is the message digest signed by node l.
C_{new} : New checksum calculated by node l. C_{new} = E(PUk, (C_{old} + Signed MD + PU_l))

To allow an agent to collect information while moving between hosts, we provide secure append only table. This table is characterized by the fact that data can only be appended to the log; there is no way that data can be removed or modified without the owner being able to detect this. Initially the table and has only an associated checksum C_{init} When the agent moves to a node l, it appends its ID, hash of its cost and Signed MD, and the new checksum.

When the agent comes back to its owner, owner can verify if the table has been tampered with. The owner starts reading the table at the end by successively computing D(PR_k, C_{new}), on the checksum C_{new}. Each iteration will return a checksum C_{next} for the next iteration, along with Signed MD and public key, PU_l for some node l. The agent owner can then verify if the last element in the table matches signed MD, for each node in each iteration step. The iteration stops when the initial checksum is reached or when a there is a signature mismatch. If the agent is not back after T1 expires, a new agent is launched.

Algorithm 1 (Executed by Hello Agent owner node k)

```
/*On receiving Hello Agent, all nodes append their data in Hello Message table*/
1. if (Hello Agent returns after visiting all neighbors) then
2. Perform verification of ID and Hash of Node Cost against Signed MD
3. Validate the checksum to ensure integrity of table data
4. Send Begin-Election Agent only to the nodes who have placed data in Hello Message table;
/* Begin-Election agent contains Begin-Election Message table which allows each node to put their costs of analysis*/
4. else if(neighbors(k)= ) then
5. Launch IDS.
6. end if
```

When hello Agent returns, node k checks whether it has received all the hash values from its neighbors. Nodes from whom the entries are not received are excluded from the election. On receiving the entries from all neighbors, node k creates Begin-Election Agent containing Begin-Election table as shown in Table 3, in which allows nodes to enter their cost of analysis, and then, starts timer T2. If node k is the only node in the network or it does not have any neighbors, then it launches its own IDS.

Table 3. Begin-Election Table

Node Unique ID	Nodes Cost	Signed MD	Checksum
ID_k	$cost_k$	E(PR_k,ID_k +costk)	C_{init}

When Begin-Election agent returns, the node k compares the hash value of Hello Message table to the value received in the Begin-Election Message table to verify the cost of analysis for all the nodes.

```
Algorithm 2 (Executed by Begin-Election Agent owner node k)
/*On receiving Begin-Election Agent, all nodes append their actual cost in Begin-Election table*/
1. if (Begin-Election Agent returns after visiting all neighbors) then
2. Perform verification of ID and Node Cost against Signed MD
3. Validate the checksum to ensure integrity of table data
4. Calculate Hash of Nodes Cost and compare to Hash of Nodes Cost from Hello Message table
4. Send Vote Agent to the nodes;
/* Vote Agent contains read only Vote Message table which allows each node to find nodes with least costs of analysis and cast their votes*/
```

Node k then creates a read only Vote Agent containing Vote Message table containing ID, hash of cost, actual cost of analysis that it has obtained from each node as shown in Table 4. Vote Agent is used by a node to view cost of analysis of other nodes and cast their vote for any one neighboring node and start a timer. Note that cost of analysis is lesser if a node has higher energy level. Reputation value of all neighbouring nodes is maintained by each node

Table 4. Read Only Vote Message Table

Node Unique ID	Hash of Nodes Cost	Nodes Cost
ID_l	Hash($cost_l$)	$Cost_l$
ID_i	Hash($cost_i$)	$Cost_i$

```
Algorithm 3 (Executed by every node)
/*Each node votes for one node among the neighbors*/
if (cost of neighbor node i is less than cost of node l) then
send Vote $PR_l(ID_l,ID_i)$ ;
leadernode(l) :=i;
endif
```

On receiving Vote Agent containing Vote Message table, the node l checks cost of analysis for the neighboring nodes to calculate the least-cost value and sends Vote for node i as in Algorithm 3. Vote message contains digitally signed ID_l of the source node and the ID_i of the proposed leader. Then, node l sets node i as its leader in order to update later on its reputation. The second least cost of analysis is needed by the leader node to calculate the payment [6]. If node l has the least cost among all its neighbors, then it votes for itself and starts a timer.

Algorithm 4 (Executed by Elected leader-IDS node)[6]

```
/*Send Acknowledge message to the neighbor nodes*/
1. Leader(i) := TRUE;
2. Compute Payment, $P_i$;
3. update service-table(i);
4. update reputation-table(i);
5. Acknowledge = $P_i$ + all the votes;
6. Send Acknowledge(i);
7. Launch IDS.
```

When the timer expires, elected nodes calculate their payment. The elected node i calculates its payment P_i given in the form of reputation where truth telling is dominant strategy as given in [6].Updates the service table and reputation table for the voting nodes, sends an Acknowledge message to all the voting nodes as in Algorithm 3. The Acknowledge message contains the payment and all the votes the leader received. The leader then launches its IDS.

As described in [6] each node verifies the payment received from leader node and updates its reputation table. Leader nodes run the IDS for inspecting packets, during an interval T_{ELECT} after which election process is carried out again to choose new leader-IDS.

6 Conclusions

A mobile agent framework is deployed for communication among the nodes for Cluster Head Election related information. In our proposed algorithm, we use the autonomy and mobility associated with mobile agent technology to present an efficient and flexible and secure system, to deal with weak connectivity and inadequate bandwidth in MANETs and device a method to effectively and efficiently elect a cluster head. The above algorithm is cost efficient and secure leader election drastically reduces the computation overhead specially in case of high mobility networks where the frequency of elections is more.

In the original algorithm [6], each normal node signs three messages and verifies 3•Ngi•+ 1 messages, where Ngi is the number of neighboring nodes. On the other hand, the leader node signs four messages and verifies 3•Ngi•messages. Each node must find the least cost node which requires O(log(Ngi)). Therefore, each node approximately performs O(Ngi) verifications, O(1) signatures, and O(log(Ngi)) to calculate the least-cost node. Thus, the computation overhead for each node is approximately O(Ngi).

In our algorithm each normal node signs three messages but verification is done only by mobile agent owner therefore the computation overhead of all normal nodes is less than O(Ngi).Communication overhead of each node broadcasting message in the first three steps is reduced using mobile agents and only leader node needs to broadcast a final Acknowledge message.

Thus by modifying the algorithm in [6] to introduce mobile agents we can achieve reduction in communication and computation overhead leading to lesser power consumption. This technique of Secure Leader Election can be useful in Wireless Sensor Networks. As a future direction we would like to extend this model to provide efficient Intrusion Detection using Secure Mobile Agents.

References

1. Murthy, C.S.R., Manoj, B.S.: Ad Hoc Wireless Networks. Pearson Education (2008)
2. Zhang, Y., Lee, W.: Intrusion Detection in Wireless Ad-Hoc Networks. In: Proc. MOBICOM 2000, pp. 275–283. ACM Press, Boston (2000)
3. Huang, Y.-A., Lee, W.: A Cooperative Intrusion Detection System for Ad Hoc Networks. In: Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks, SASN 2003, pp. 135–147. ACM, NY (2003)
4. Kachirski, O., Guha, R.: Efficient Intrusion Detection Using Multiple Sensors in Wireless Ad Hoc Networks. In: Proc. IEEE Hawaii Int'l Conf. System Sciences, HICSS (2003)
5. Otrok, H., Mohammed, N., Wang, L., Debbabi, M., Bhattacharya, P.: A game-theoretic intrusion detection model for mobile ad hoc networks. Comput. Commun. 31(4), 708–721 (2008)
6. Mohammed, N., Otrok, H., Wang, L., Debbabi, M., Bhattacharya, P.: Mechanism Design-Based Secure Leader Election Model for Intrusion Detection in MANET. IEEE Transactions on Dependable and Secure Computing 8(1) (2011)
7. Zeng, Chen, Z.: A Cluster Header Election Scheme Based on Auction Mechanism for Intrusion Detection in MANET. In: International Conference on Network Computing and Information Security (2011)
8. Anderegg, L., Eidenbenz, S.: Ad Hoc-VCG: A Truthful and Cost-Efficient Routing Protocol for Mobile Ad Hoc Networks with Selfish Agents. In: Proc. ACM MobiCom (2003)
9. Agents, `http://www.java2s.com/Article/Java/SOAservices/The_Architecture_of_Aglets_IBMs_Mobile_Java_Agents_Under_the_Hood.html`
10. Albers, P., Camp, O., Percher, J.-M., Jouga, B., Ludovic, M., Puttini, R.: Security in ad hoc networks: a general intrusion detection architecture enhancing trust based approaches. In: Proc. of the First International Workshop on Wireless Information Systems, WIS 2002, pp. 1–12 (2002)
11. Tripathi, A.R.: A Security Architecture for Mobile Agents in Ajanta. In: Proceedings of the 20th International Conference on Distributed Computing Systems, ICDCS 2000, pp. 402–409. IEEE Computer Society, Washington, DC (2000)
12. Krishna, P., Vaidya, N.H., Chatterjee, M., Pradhan, D.K.: A Cluster-Based Approach for Routing in Dynamic Networks. Proc. ACM SIGCOMM Computer Comm. Rev. (1997)