Selection of Nodes for Intrusion Detection in MANETS

[1]S. Shashikanth, [2]K. Srinivas, [3]TP. Shekhar
[1]Dept. of Computer Science, JNTUH University, AP, India
[2]Sree Chaitanya College of Engineering, AP, India

Abstract
In this we study Leader election in the presence of selfish nodes for intrusion detection in mobile adhoc networks (MANETs). To balance the resource consumption among all nodes and prolong the lifetime of a MANET, nodes with the most remaining resources should be elected as the leaders. However, there are two main obstacles in achieving this goal. First, without incentives for serving others, a node might behave selfishly by lying about its remaining resources and avoiding being elected. Second, electing an optimal collection of leaders to minimize the overall resource consumption may incur a prohibitive performance overhead, if such an election requires flooding the network. To address the issue of selfish nodes, we present a solution based on mechanism design theory. More specifically, the solution provides nodes with incentives in the form of reputations to encourage nodes in honestly participating in the election process. The amount of incentives is based on the Vickrey, Clarke, and Groves (VCG) model to ensure truth-telling to be the dominant strategy for any node. To address the optimal election issue, we propose a series of local election algorithms that can lead to globally optimal election results with a low cost. We address these issues in two possible application settings, namely, Cluster Dependent Leader Election (CDLE) and Cluster Independent Leader Election (CILE). The former assumes given clusters of nodes, whereas the latter does not require any pre clustering. Finally, we justify the effectiveness of the proposed schemes through extensive experiments.

Keywords
Leader Election, Intrusion Detection Systems, Mechanism Design and MANET Security

I. Introduction

Mobile Ad hoc Networks (MANET) have no fixed chokepoints/bottlenecks where Intrusion Detection Systems (IDSs) can be deployed. Hence, a node may need to run its own IDS and cooperate with others to ensure security. This is very inefficient in terms of resource consumption since mobile nodes are energy-limited. To overcome this problem, a common approach is to divide the MANET into a set of one hop clusters where each node belongs to at least one cluster. The nodes in each cluster elect a leader node (cluster head) to serve as the IDS for the entire cluster. The leader-IDS election process can be either random or based on the connectivity . Both approaches aim to reduce the overall resource consumption of IDSs in the network. However, we notice that nodes usually have different remaining resources at any given time, which should be taken into account by an election scheme. Unfortunately, with the random model, each node is equally likely to be elected regardless of its remaining resources. The connectivity index-based approach elects a node with a high degree of connectivity even though the node may have little resources left. With both election schemes, some nodes will die faster than others, leading to a loss in connectivity and potentially the partition of network. Although it is clearly desirable to balance the resource consumption of IDSs among nodes, this objective is difficult to achieve since the resource level is the private information of a node. Unless sufficient incentives are provided, nodes might misbehave by acting selfishly and lying about their resources level to not consume their resources for serving others while receiving others services. Moreover, even when all nodes can truthfully reveal their resource levels, it remains a challenging issue to elect an optimal collection of leaders to balance the overall resource consumption without flooding the network. MANET composed of ten nodes labelled from N1 to N10. These nodes are located in 5 one hop clusters where nodes N5 and N9 belong to more than one cluster and have limited resources level. We assume that each node has different energy level, which is considered as private information. At this point, electing nodes N5 and N9 as leaders is clearly not desirable since losing them will cause a partition in the network and nodes will not be able to communicate with each other. However, with the random election model, nodes N5 and N9 will have equal probability, compared to others, in being elected as leaders. The nodes N5 and N9 will definitely be elected under the connectivity index-based approach due to their connectivity indices. Moreover, a naive approach for electing nodes with the most remaining resources will also fail since nodes' energy level is considered as private information and nodes might reveal fake information if that increases their own benefits. Finally, if the nodes N2, N5 and N9 are selfish and elected as leaders using the above models, they will refuse to run their IDS for serving others. The consequences of such a refusal will lead normal nodes to launch their IDS and thus die faster. In a public mobile ad hoc network (MANET), users may be selfish and refuse to forward packets for other users. Therefore, an incentive mechanism must be in place. We adopt the "pay for service" model of cooperation, and propose an auction-based incentive scheme (called iPass) to enable cooperative packet forwarding behaviour in MANET. Each flow pays the market price of packet forwarding service to the intermediate routers. The resource allocation mechanism in our scheme is based on the generalized Vickrey auction with reserve pricing. We prove that in our scheme, user's truthful bidding of utility remains a dominant strategy, users and routers have incentive to participate in the scheme, and packet forwarding always leads to higher social welfare for the whole network. We design a signalling protocol to implement the scheme, and show that it can serve as an explicit rate-based flow control mechanism for the network. Therefore, pass is a joint solution of incentive engineering and flow control in a no cooperative MANET. Simulation results show that pass is able to determine the auction outcome quickly,. A critical element in controlling information access is ensuring that only the appropriate individuals have the cryptographic keys that enable them to decode the disseminated information. For example, to maintain forward confidentiality, when a member leaves the session, the remaining members must be rekeyed to ensure that the departing individual cannot listen in on the future communications. Similarly, backward confidentiality requires rekeying when a new member joins an existing session. Otherwise, the new member would be able to decrypt any past archived exchanges for which he/she was not authorized. Since data cannot be exchanged while member's data keys are being updated, the challenge for any key management system is how to generate and distribute new keys such that the data remains secure while the overall impact on system performance is minimized. Mobility complicates key management by allowing members to not only leave or join a session but also transfer between networks while remaining in the session. Since a mobile user may accumulate information about the local security services for each area he/she visits, the key management system must consider the level of trust to impart to these mobile members and the performance implications should the member leave the session. Furthermore, as a member moves, the network latency between the member and the key management services may change and result in additional performance degradation. This paper explores the problem of rekeying large numbers of coalition partners where both their access to information and their position vary with time. Given the political necessity that individual coalitions maintain separate networks, we consider both the implication of keying complexity and network topology on the performance of various rekeying approaches.

A. Motivating Example

Fig. 1, illustrates a MANET composed of ten nodes labeled from N1 to N10. These nodes are located in 5 one hop clusters where nodes N5 and N9 belong to more than one cluster and have limited resources level. We assume that each node has different energy level, which is considered as private information. At this point, electing nodes N5 and N9 as leaders is clearly not desirable since losing them will cause a partition in the network and nodes will not be able to communicate with each other. However, with the random election model , nodes N5 and N9 will have equal probability, compared to others, in being elected as leaders. The nodes N5 and N9 will definitely be elected under the connectivity index-based approach due to their connectivity indices . Moreover, a naive approach for electing nodes with the most remaining resources will also fail since nodes' energy level is considered as private information and nodes might reveal fake information if that increases their own benefits. Finally, if the nodes N2, N5 and N9 are selfish and elected as leaders using the above models, they will refuse to run their IDS for serving others. The consequences of such a refusal will lead normal nodes to launch their IDS and thus die faster. B. Our Proposed Solution In this paper, we propose a solution for balancing the resource consumption of IDSs among all nodes while

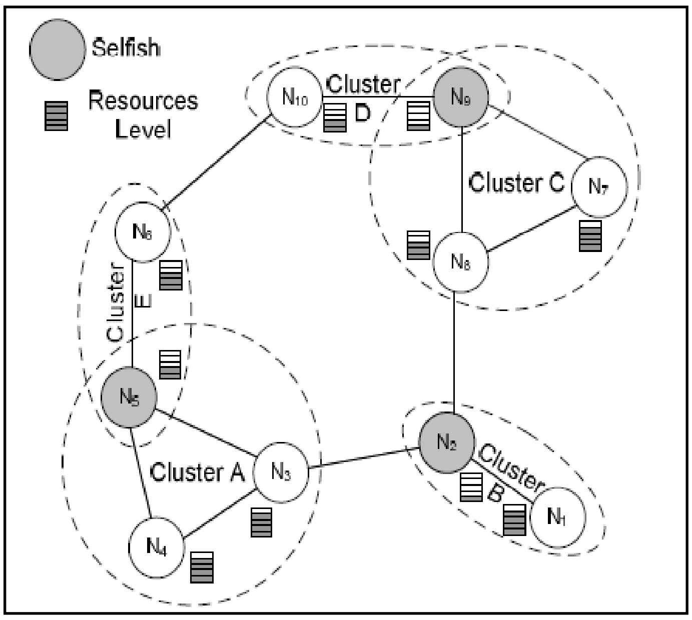

Fig. 1: An Example Scenario of Leader Election in MANET Venting Nodes from Behaving Selfishly

To address the selfish behaviour, we design incentives in the form of reputation to encourage nodes to honestly participate in the election scheme by revealing their cost of analysis. The cost of analysis is designed to protect nodes' sensitive information (resources level) and ensure the contribution of every node on the election process (fairness). To motivate nodes in behaving normally in every election round, we relate the amount of detection service that each node is entitled to the nodes' reputation value. Besides, this reputation value can also be used to give routing priority and to build a trust environment. The design of incentives is based on a classical mechanism design model, namely, Vickrey, Clarke, and Groves (VCG) [21]. The model guarantees that truth-telling is always the dominant strategy for every node during each election phase. On the other hand, to find the globally optimal cost-efficient leaders, a leader election algorithm is devised to handle the election process, taking into consideration the possibility of cheating and security flaws, such as replay attack. The algorithm decreases the percentage of leaders, single node clusters, maximum cluster size and increases average cluster size .Last but not least, we address these issues in two possible settings, namely, Cluster Independent Leader Election (CILE) and Cluster Dependent Leader Election (CDLE). In the former, the leaders are elected according to the received votes from the neighbour nodes. The latter scheme elects leaders after the network is formulated into multiple clusters. In both schemes, the leaders are elected in an optimal way in the sense that the resource consumption for serving as IDSs will be balanced among all nodes overtime. Finally, we justify the correctness of proposed methods through analysis and simulation. Empirical results indicate that our scheme can effectively improve the overall lifetime of a MANET. The main contribution of this paper is a unified model that is able to: (1) Balance the IDS resource consumptions among all nodes by electing the most cost-efficient leaders. (2) Motivate selfish nodes to reveal their truthful resources level.The problem of selfishness and energy balancing exists in many other applications to which our solution are also applicable. Like in IDS scheme, leader election is needed for routing and key distribution in MANET. In key management, a central key distributer is needed to update the keys of nodes. In routing, the nodes are grouped into small clusters and each cluster elects a cluster head (leader) to forward the packets of other nodes. Thus, one node can stay alive while others can be in the energy-saving mode .The election of leader a node is done randomly, based on connectivity (nodes' degree) or based on a node's weight (here the weight refers to the remaining energy of a node. We have already pointed out the problems of random model and connectivity model. We believe that a weight based leader election should be the proper method for election .Unfortunately, the information regarding the remaining energy is private to a node and thus not verifiable. Since nodes might behave selfishly, they might lie about their resource level to avoid being the leader if there is no mechanism to motivate them. Our method can effectively address this issue.

B. Paper Outline

The rest of this paper is organized as follows: Section II formulates the problem. Section III describes our leader election mechanism where the cost of analysis function, reputation model and payment design are given. Section IV analyzes our mechanisms against selfish and malicious nodes. Section V devises the election algorithm needed to handle the election process.

II. Problem Statement

We consider a MANET where each node has an IDS and a unique identity. To achieve the goal of electing the most cost efficient nodes as leaders in the presence of selfish and malicious nodes, the following challenges arise: First, the resource level that reflects the cost of analysis is considered as a private information. As a result, the nodes can reveal fake information about their resources if that could increase their own benefits. Second, the nodes might behave normally during the election but then deviate from normal behavior by not offering the IDS service to their voted nodes. In our model, we consider MANET as an undirected graph G = (N,L) where N is the set of nodes and L is the set of bidirectional links. We denote the cost of analysis vector as C = {c1, c2, . . . , cn} where n is the number of nodes in N. We denote the election process as a function vtk(C, i) where vtk(C, i) = 1 if a node i votes for a node k; vtk(C, i) = 0, otherwise. We assume that each elected leader allocates the same budget B (in the number of packets) for each node that has voted for it. Knowing that, the total budget will be distributed among all the voting nodes according to their reputation. This will motivate the nodes to cooperate in every election round that will be held on every time TELECT . Thus, the model will be repeatable. For example, if B = 25 packet/sec and the leder gets 3 votes, then the leader's sampling budget is 75 packet/sec. This value is divided among the 3 nodes based on their reputation value. The objective of minimizing the global cost of analysis while serving all the nodes can be expressed by the following Social Choice Function (SCF): SCF =S(C) =min Clearly, in order to minimize this SCF, the following must be achieved. First, we need to design incentives for encouraging each node in revealing its true cost of analysis value c, which will be addressed in Section III. Second, we need to design an election algorithm that can provably minimize the above SCF while not incurring too much of performance overhead. This will be addressed in

III. Leader Election Mechanism

In this section, we present our leader election mechanism for truthfully electing the leader nodes. To make the paper self-contained, the background on mechanism design is given in Subsection III-A. Subsection III-B formulates our model using the standard mechanism design notations. To achieve the design goal, the cost of analysis function is given in Subsection III-C followed by the reputation system model given in Subsection III-D. Finally, the design of the payment for the two models.

A. Mechanism Design Background

Mechanism design is a sub-field of microeconomics and game theory]. Mechanism design uses game theory tools to achieve the desired goals. The main difference between game theory and mechanism design is that the former can be used to study what could happen when independent players act selfishly. On the other hand, mechanism design allows a game designer to define rules in terms of the Social Choice Function (SCF) such that players will play according to these rules. The balance of IDS resource consumption problem can be modeled using mechanism design theory with an objective function that depends on the private information of the players. In our case, the private information of the player is the cost of analysis which depends on the player's energy level. Here, the rational players select to deliver the untruthful or incomplete information about their preferences if that leads to individually better outcomes . The main goal of using mechanism design is to address this problem by: 1) Designing incentives for players (nodes) to provide truthful information about their preferences over different outcomes. 2) Computing the optimal system-wide solution, which is defined according to Equation 1.

A mechanism design model consists of n agents where each agent has a private information, known as the agent's type. Moreover, it defines a set of strategies Ai for each agent i. The agent can choose any strategy a_i _ A_i to input in the mechanism. According to the inputs $(a_i, . . . , a_n)$ of all the agents, the mechanism calculates an output $o = o(a_1, . . . , a_n)$ and payment vector $p = (p_1, . . . , p_n)$ where $p_i = p_i(a_1, . . . , a_n)$. The preference of each agent from the output is calculated by a valuation function, vi. This is a quantification in terms of a real number to evaluate the output for an agent i. Thus, the utility of a node is calculated as This means, the utility is the combination of output measured by valuation function and the payment it receives from the mechanism. In direct revelation mechanism , every agent i has a type, . Each agent gives an input to the mechanism. The agent chooses the strategy according to its type, where , which is chosen from the strategy set Selfish, Normal. We assume that normal agents follow the protocol whereas selfish agents deviate from the defined protocol if the deviation leads to a higher utility. Although the prime objective of these agents is not to actively harm others but their presence can passively harm others. Last but not least, the mechanism provides a global output from the input vector and also computes a specific payment for each agent. The goal is to design a strategy-proof mechanism where each agent gives an input based on its real type (known as the dominant strategy) such that it maximizes its utility regardless of the strategies of others. A strategy is dominated by another strategy if the second strategy is at least as good as the other one regardless of the other players' strategy. This is expressed as follows: denotes non-selfishness and denotes selfishness. Note that ui is maximized only when pi is given by the mechanism. The question is: How to design the payments in a way that makes truth-telling the dominant strategy? In other words, how to motivate nodes to reveal truthfully their valuation function? The VCG mechanism answers this question by giving the nodes a fixed payment independent of the nodes' valuation, which is equal to the second best valuation. The design of the payment, according to our scenarios, is given in the following subsections.

B. The Mechanism Model

We treat the IDS resource consumption problem as a game where the N mobile nodes are the agents/players. Each node plays by revealing its own private information (cost of analysis) which is based on the node's type . The type is drawn from each player's available type set Normal, Selfish}. Each player selects his own strategy/type according to how much the node values the outcome. If the player's strategy is normal then the node reveals the true cost of analysis. In section IV a detailed analysis is given. We assume that each Player has a utility function where, is the type of all the other nodes except i. is the valuation of player i of the output knowing that O is the set of possible outcomes. In our case, if the node is elected then vi is the cost of analysis. Otherwise is since the node will not be the leader and hence there will be o cost to run the IDS. is the payment given by the mechanism to the elected node. Payment is given in the form of reputation. Nodes that are not elected receive no payment. Note that, is what the player usually seeks to maximize. It reflects the amount of benefits gained by player if he follows a specific type θi. Players might deviate from revealing the truthful valuation for the cost of analysis if that could lead to a better payoff. Therefore, our mechanism must be strategy-proof where truth-telling is the dominant strategy. To play the game, every node declares its corresponding cost of analysis where the cost vector C is the input of our mechanism. For each input vector, the mechanism calculates its corresponding output and a payment vector p = (p_1, . . . , p_n). Payments are used to motivate players to behave in accordance with the mechanism goals. In the following subsections, we will formulate the following components:

1. Cost of analysis function: It is needed by the nodes to compute the valuation function.
2. Reputation system: It is needed to show how:

- Incentives are used once they are granted.
- Misbehaving nodes are cached and punished.

3. Payment design: It is needed to design the amount of incentives that will be given to the nodes based on VCG .Information can be used maliciously for attacking the node with the least resources level. Second, if the cost of analysis function is designed only in terms of nodes' energy level, then the nodes with the low energy level will not be able to contribute and increase their reputation values. To solve the above problems, we design the cost of analysis function with the following two properties: Fairness and Privacy. The former is to allow nodes with initially less resources to contribute and serve as leaders in order to increase their reputation. On the other hand, the latter is needed to avoid the malicious use of the resources level, which is considered as the most sensitive information. To avoid such attacks and to provide fairness, the cost of analysis is designed based on the reputation value, the expected number of time slots that a node wants to stay alive in a cluster and energy level. Note that the expected number of slots and energy level are considered as the nodes' private information. To achieve our goal, we assume that the nodes are divided into l energy classes with different energy levels. The lifetime of a node can be divided into time-slots. Each node is associated with an energy level, denoted by , and the number of expected alive slots is denoted by . Based on these requirements, each node i has a power factor We introduce the set of thresholds to categorize the classes as in The reputation of node i is denoted by Ri. Every node has a sampling budget based on its reputation. This is indicated by the percentage of sampling, The notation represents the cost of analysis for a single packet and id's is used to express the energy needed to run the IDS for one time slot. The cost of analysis of each node can be calculated based on energy level. However, we considered energy level, expected lifetime and the present PS of node to calculate the cost of analysis. We can extend the cost of analysis function to more realistic settings by considering the computational level and cost of collecting and analyzing traffic. Our castoff- analysis function is formulated as According to the above formulation, the nodes have an Infinite cost of analysis if its remaining energy is less than the energy required to run the IDS for one time slot. This means that its remaining energy is too low to run the IDS for an entire time-slot. Otherwise, the cost of analysis is calculated through dividing the percentage of sampling by the power factor. The cost of analysis c is proportional to the percentage of sampling and is inversely proportional to the power factor. The rationale behind the definition of the function is the following. If the nodes have enough PS, they are not willing to loose their energy for running the IDS. On the other hand, if PF is larger, then the cost-of-analysis becomes smaller since the nodes have higher energy levels. In the rest of the paper, we will use cost and cost-of-analysis interchangeably. We show the effect of our cost function over PS through an example. Table I shows the PS for 20 nodes divided equally in 4 energy classes where nodes in class 4 have the most resources. Table I indicates that initially nodes belonging to lower energy level have a small budget. As the time goes by, the nodes belonging to lower energy class gains more budget while the budget of higher classes decreases. This justifies that our cost function is able to balance the energy of the nodes and gives a fair budget to all nodes.

C. Reputation System Model

Before we design the payment, we need to show how the payment in the form of reputation can be used to: (1) Motivate nodes to behave normally and (2) punish the misbehaving nodes. Moreover, it can be used to determine whom to trust .To motivate the nodes in behaving normally in every election round, we relate the cluster's services to nodes' reputation. This will create a competition environment that motivates the nodes to behave normally by saying the truth. To enforce our mechanism, a punishment system is needed to prevent nodes from behaving selfishly after the election. Misbehaving nodes are punished by decreasing their reputation and consequently are excluded from the cluster services if the reputation is less than a predefined threshold. As an extension to our model, we can extend our reputation system to include different sources of information such as routing and key distribution with different assigned weights. shows the abstract model of our reputation system where each node has the following components: Reputation System Model • Monitor or Watchdog: It is used to monitor the behavior of the elected leader. To reduce the overall resource consumption, we randomly elect a set of nodes, known as checkers, to perform the monitoring process. The selected checkers mirror a small portion of the computation done by the leader so the checkers can tell whether the leader is actually carrying out its duty. We assume the checkers are cooperative because the amount of computation they conduct for monitoring the leader only amounts to a marginal resource consumption, which is dominated by the benefit of receiving intrusion detection service from the leader Information Exchange: It includes two types of information sharing:

(1) The exchange of reputation with other nodes in other clusters (i.e., for services purposes).

(2) To reduce the false positive rate, the checkers will exchange information about the behavior of the leader to make decision about the leader's behavior. Reputation System: It is defined in the form of a table that contains the ID of other nodes and their respective reputation R. The node that has the highest reputation can be considered as the most trusted node and is given priority in the cluster's services. Therefore, the rational nodes are motivated to increase their reputation value by participating in the leader election. Threshold Check: It has two main purposes:

(1) To verify whether nodes' reputation is greater than a predefined threshold. If the result is true then nodes' services are offered according to nodes' reputation.

(2) To verify whether a leader's behavior exceeds a predefined misbehaving threshold. According to the result, the punishment System is called. Service System: To motivate the nodes to participate in every election round, the amount of detection service provided to each node is based on the node's reputation. Each elected leader has a budget for sampling and thus only limited services can be offered. This budget is distributed among the nodes according to their reputation. Besides, this reputation can also be used for packet forwarding. Packets of highly reputed nodes should always be forwarded. On the other hand, if the source node has an unacceptably low reputation then its packet will have less priority. Hence, in every round, nodes will try to increase their reputation by becoming the leader in order to increase their

services. Punishment System: To improve the performance and reduce the false-positive rate of checkers in catching and punishing a misbehaving leader, we have formulated in a cooperative game-theoretical model to efficiently catch and punish misbehaving leaders with low false positive rate. Our catch-and-punish model was made up of k detection-levels, representing different levels of selfish behaviours of the leader-IDS. This enables us to better respond to the misbehaving leader-IDS depending on which detection-level it belongs to. Hence, the percentage of checkers varies with respect to the detection-level. Once the detection exceeds a predefined threshold, the leader will be punished by decreasing its reputation value.

D. CILE Payment Design

In Cluster Independent Leader Election (CILE), each node must be monitored by a leader node that will analyze the packets for other ordinary nodes. Based on the cost of analysis vector C, nodes will cooperate to elect a set of leader nodes that will be able to analyze the traffic across the whole network and handle the monitoring process. This increases the efficiency and balances the resource consumption of an ID in the network. Our mechanism provides payments to the elected leaders for serving others (i.e., offering the detection service). The payment is based on a per-packet price that depends on the number of votes the elected nodes get. The nodes that do not get any vote from others will not receive any payment. The payment is in the form of reputations, which are then used to allocate the leader's sampling budget for each node. Hence, any node will strive to increase its reputation in order to receive more IDS services from its corresponding leader.
Weights

E. Leader Election

To start a new election, the election algorithm uses four types of messages. Hello, used by every node to initiate the election process; Begin-Election, used to announce the cost of a node; Vote, sent by every node to elect a leader; Acknowledge, sent by the leader to broadcast its payment, and also as a confirmation of its leadership. For describing the algorithm, we use the following notation:
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 neighbours. leader node(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 is a leader and FALSE otherwise.Initially, each node k starts the election procedure by broadcasting a Hello message to all the nodes that are one hop from node k and starts a timer T1. This message contains the hash value of the node's cost of analysis and its unique identifier (ID). This message is needed to avoid cheating where further analysis is conducted in Section VI-B. Algorithm 1 (Executed by every node)
/* On receiving Hello, all nodes reply with their cost */

```
Algorithm 1 (Executed by every node)                .
/* On receiving Hello, all nodes reply with their cost */
1. if (received Hello from all neighbors) then
2. Send Begin-Election (IDk, costk);
3. else if(neighbors(k)=Ø) then
4. Launch IDS.
5. end if                                            .
```

On expiration of T1, each node k checks whether it has received all the hash values from its neighbors. Nodes from whom the Hello message has not received are excluded from the election. On receiving the Hello from all neighbors, each node sends Begin-Election as in Algorithm 1, which contains the cost of analysis of the node 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. Algorithm 2 (Executed by every node) /* Each node votes for one node among the neighbors */

```
Algorithm 2 (Executed by every node)                .
/* Each node votes for one node among the neighbors */
1. if (∀ n € neighbor(k), ∃ i € n : $c_i \leq c_n$) then
2. send Vote(IDk, IDi , costj≠i);
3. leader node(k):= i;
4. end if                                            .
```

On expiration of T2, the node k compares the hash value of Hello to the value received by the Begin-Election to verify the cost of analysis for all the nodes. Then node k calculates the least-cost value among its neighbors and sends Vote for node i as in Algorithm 2. The Vote message contains the IDk of the source node, the IDi of the proposed leader and second least cost among the neighbors of the source node cost j≠i node k sets node i as its leader in order to update later on its reputation. Note that the second least cost of analysis is needed by the leader node to calculate the payment.

IV. Leader Election Algorithm

To run the election mechanism given in Section III, we propose a leader-election algorithm that helps to elect the most cost-efficient leaders with less performance overhead compared to the network flooding model. We devise all the needed messages to establish the election mechanism taking into consideration cheating and presence of malicious nodes. Moreover, we consider the addition and removal of nodes to/from the network due to mobility reasons. Finally, the performance overhead is considered during the design of the given algorithm where computation, communication and storage overhead are derived.

A. Objectives and Assumptions

To design the leader election algorithm, the following requirements are needed: (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 is minimized. In other words, every node has to be affiliated with the most cost efficient leader among its neighbors. Our algorithm is executed in each node taking into consideration the following assumptions about the nodes and the network architecture:

- Every node knows its (2-hop) neighbors, which is reasonable since nodes usually maintain a table about their neighbors for routing purposes.
- Loosely synchronized clocks are available between nodes.
- Each node has a key (public, private) pair for establishing a secure communication between nodes.
- Each node is aware of the presence of a new node or removal of a node. For secure communication, we can use a combination of TESLA and public key infrastructure. With the help of TESLA, loosely synchronized clocks can be available. Nodes can use public key infrastructure during election and TESLA in other cases. Recent investigations showed that computationally limited mobile nodes can also perform public

key operations. This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. k has the least cost among all its neighbors then it votes for itself and starts timer T3.

```
Algorithm 3 (Executed by Elected leader node)      .
/* Send Acknowledge message to the neighbor nodes */
1. Leader(i) := TRUE;
2. Compute Payment, $P_i$;
3. updateservice−table(i);
4. updatereputation−table(i);
5. Acknowledge = $P_i$ + all the votes;
6. Send Acknowledge(i);
7. Launch IDS.                                      .
```

On expiration of T3, the elected node i calculates its payment using equation 5 and sends an Acknowledge message to all the serving nodes as in Algorithm 3. The Acknowledge message contains the payment and all the votes the leader received. The leader then launches its IDS. Each ordinary node verifies the payment and updates its reputation table according to the payment. All the messages are signed by the respective source nodes to avoid any kind of cheating. At the end of the election, nodes are divided into two types: Leader and ordinary nodes. Leader nodes run the IDS for inspecting packets, during an interval SELECT , based on the relative reputations of the ordinary nodes. We enforce reelection every period SELECT since it is unfair and unsafe for one node to be a leader forever. Even if the topology remains same after SELECT time, all the nodes go back to initial stage and elect a new leader according to the above algorithms. Example 2: Continue from Example 1. To illustrate the election algorithm, we consider the same network topology presented in Figure 3. To elect a new leader in the 10th round, every node sends a Hello message that contains the node's ID and the hash value of the computed cost. after receiving the Hello messages, the nodes send a

A. Begin-Election Message

according to Algorithm 1. Nodes reveal their cost of analysis to the mechanism based on their type (Selfish or Normal). As mentioned, the corresponding cost is given in the second row of Table II. Then, nodes 7, 8, 9 and 10 vote for node 9 using the Vote message as in Algorithm 2. Similarly, node 6 votes for node 5; nodes 3, 4 and 5 vote for node 3; nodes 1 and 2 vote for node 1. After getting the votes, leader nodes 1, 3, 5 and 9 will calculate their payment using equation 5 as shown in Example 1. Respectively, the payment for elected leaders N1, N3 and N5 will be 45, 95 and 40. Finally, the leader nodes will send Acknowledge message using Algorithm 3 to all neighbors and run their own IDS. Upon receiving the Acknowledge, all the neighboring nodes increase the reputation of the elected Leaders, as shown in the third row.

B. Adding a New Node

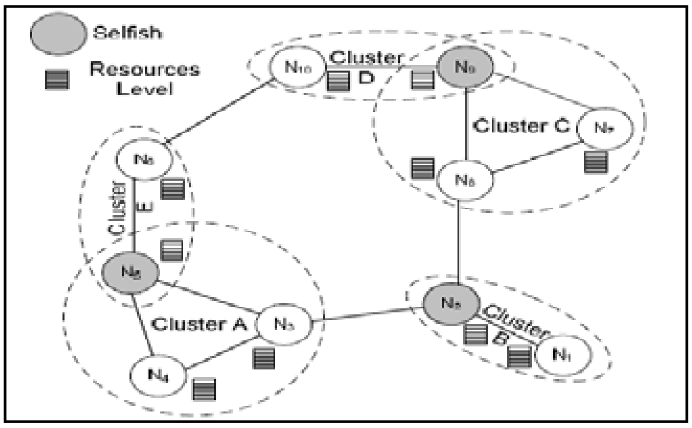

Fig. 1:

When a new node is added to the network, it either launches its own IDS or becomes an ordinary node of any leader node. To include a new node to the IDS service, four messages are needed: Hello, Status, Join and Acknowledge. Hello is sent by a new node n to announce its presence in the network. This Hello message is similar to the one presented in the previous section. Upon receiving the Hello, all the neighbours of the new

```
Algorithm 4 (Executed by neighboring nodes)      .
/* The neighboring nodes send 'Status' to new node */
1. if (leader(k) = TRUE) then
2. Status := Costk;
3. else
4. Status := leader node(k);
5. end if;
6. send Status(k, n);                              .
```

C. Performance Analysis

In this section, we analyze the performance overhead of our proposed leader algorithm. In summary, our algorithm has four steps. In the first 3 steps, all the nodes broadcast Hello, Begin-Election and Vote messages consecutively. In the last step, only the leader node sends an Acknowledge message to others.

1. Computation Overhead

Each node i signs its messages sent in the first 3 steps. Also, each node verifies the messages it received in these steps. In the 4th step, the leader node signs the Acknowledge message and others verify. Hence each normal node signs 3 messages and verifies 3|Ngi| + 1 messages where Ngi is the number of neighboring nodes. On the other hand, the leader node signs 4 messages and verifies 3|Ngi| messages. Note that 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 O(Ngi) + O(1) + O(log(Ngi)) $\approx$ O(Ngi). Since our algorithm involves more verification than signing, nodes can use the public key cryptosystem of to verify a signature in 0.43s. Since leader election will take place after a certain interval, this computational overhead is tolerable.

2. Communication Overhead

Each node i broadcasts one message in the first 3 steps and only the leader node broadcasts a final Acknowledge message in the 4th step. Hence, the total communication overhead of our

ISSN : 0976-8491 (Online) | ISSN : 2229-4333 (Print)

election algorithm is 3|Ng|1 O(Ngi), where |Ngi| is the number of neighboring nodes.

3. Storage Overhead

According to the algorithm, each node maintains a reputation-table, neighbors list and two variables: Leader node and leader. The leader node keeps an extra service-table. Hence, each normal node needs |Ni| + |Ngi| + 2 storage and the leader node needs |Ni| + |Ngi| + |Vi| + 2. Knowing that |Ni| is the number of nodes in the network, |Vi| is the number of votes the leader node received where |Ni| > |Ngi| > |Vi|. Therefore, the total storage for each node is in the order of O(Ni). For CDLE, the network has to be initially clustered. Hence there is an extra overhead for clustering. A comparison of different clustering algorithms is presented

D. Removing a Node

When a node is disconnected from the network due to many reasons; such as, mobility or battery depletion, then the neighbor nodes have to reconfigure the network. We assume that whenever a node dies, its neighbors are aware of it. At first a Dead(n) message is circulated to all neighbors to confirm the removal of node n. On receiving the Dead(n) message, the neighbor node k checks whether node n is its leader node or not. If node n is the leader node of node k, then node k announces a new election and updates its reputation table. On the other hand, if node n is an ordinary node then its leader node updates its serving list.

```
Algorithm 5 (Executed by neighboring nodes)     .
/* The neighboring nodes reconfigure the network and */
/* declare new election if necessary*/
1. if (leadernode(k) = n) then
2. leadernode(k):= NULL;
3. updatereputation(k);
4. send Begin − Election as in Algorithm 1;
5. end if;
6. if (leader(k) = TRUE) then
7. if (n € service(k)) then
8. updateservice();
9. end if;
10. end if;                                        .
```

V. Correctness and Security Properties of the Algorithm

In this section, we discuss the correctness and security properties of our election algorithm. We prove that our algorithm satisfies the requirements and provides the necessary security properties for secure election.

A. Algorithmic Correctness

Here, we prove that our algorithm achieves our objectives mentioned in section V-A. Proposition 1: Our algorithm confirms that each node is monitored by a leader node. Proof: It is easily noticeable that after executing the election algorithm, each node is assigned a role. According to Algorithm 2, a node is either a leader or ordinary within a finite time. Note that an ordinary node could be a checker that monitors the behavior of the leader. After receiving Hello and Begin-Election messages from all the neighbor nodes within (T1 + T2) time, nodes are sorted according to their cost of analysis. By executing Algorithm 2, each node sets its variable leader node(k) to k if node k has the least cost of analysis. Nodes can not do anything but to send the Vote message to the deserving candidate. If a node does not have any neighbor, it becomes the leader node according to Algorithm 1. Besides, if a node loses its connection with the leader due to change in the network topology, it can always get associated with another leader through Algorithms 4 and 5. Thus, in all cases a node is either a leader or ordinary (monitored by a leader node).Proposition 2: The overall cost of analysis for protecting the whole network is minimized. Proof: According to proposition 1, each node is assigned a role and the role is decided according to the cost of analysis. Each node sends a Vote message to the node which has the least cost of analysis. Thus, our election scheme minimizes the SCF function depicted in equation 1 through assigning each node to the most cost-efficient leader. Since each node can affect only two-hop away nodes, the locally optimal election results are sufficient to yield the globally optimal result (that is, the minimized SCF function). One exception can occur when a node is added after the election and the new node has the minimum cost of analysis. We don't elect the new node as a leader since it will cause communication overhead (frequent leader change) in the network and could be used maliciously to disrupt the IDS service. The new node has to wait for the expiration of TELECT to participate in the new election.. Security Concerns Our proposed algorithm itself has to be secure along with its algorithmic correctness, which we believe it is hard to achieve especially in a distributed environment. Even though, our algorithm is able to prevent some security flaws such as reply attack and avoid cheating. In the following, we discuss some of the security properties of our algorithm. Algorithm security properties: Since we assume the presence of TESLA and PKI protocols, all the messages are This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

IV. Conclusion and Future Work

The unbalanced resource consumption of IDSs in MANET and the presence of selfish nodes have motivated us to propose an integrated solution for prolonging the lifetime of mobile nodes and for preventing the emergence of selfish nodes. The solution motivated nodes to truthfully elect the most cost efficient nodes that handle the detection duty on behalf of others. Moreover, the sum of the elected leaders is globally optimal. To achieve this goal, incentives are given in the form of reputations to motivate nodes in revealing truthfully their costs of analysis. Reputations are computed using the well known VCG mechanism by which truth-telling is the dominant strategy. We also analyzed the performance of the mechanisms in the presence of selfish and malicious nodes. To implement our mechanism, we devised an election algorithm with reasonable performance overheads. We also provided the algorithmic correctness and security properties of our algorithm. We addressed these issues into two applications: CILE and CDLE. The former does not require any pre clustering whereas CDLE requires nodes to be clustered before running the election mechanism. Simulation results showed that our model is able to prolong the lifetime and balance the overall resource consumptions among all the nodes in the network. Moreover, we are able to decrease the percentage of leaders, single node clusters, maximum cluster size and increase average cluster size. These properties allow us to improve the detection service through distributing the sampling budget over less number of nodes and reduce single nodes to launch their IDS.

VI. Acknowledgments

We thank our thesis advisors, Associate professors and K.Srinivas,T.P.Shekhar,D.Thirupathi for their Support and guidance. Their energy and insight at all levels of constant inspiration

References

[1] T. Anantvalee, J. Wu,"A survey on intrusion detection in mobile Ad-Hoc networks", Wireless/Mobile Network Security, 2006.

[2] L. Anderegg, S. Eidenbenz,"Ad hoc-VCG: A truthful and costefficient routing protocol for mobile Ad-Hoc networks with selfish agents", In proc. of the ACM International Conference on Mobile Computing and Networking (MobiCom), 2003.

[3] F. Anjum, P. Mouchtaris," Security for Wireless Ad Hoc Networks", John Wiley & Sons. Inc., USA, 2007.

[4] S. Basagni,"Distributed and mobility-adaptive clustering for multimedia support in multi-hop wireless networks", In proc. of the IEEE International Vehicular Technology Conference (VTC), 1999.

[5] S. Basagni,"Distributed clustering for ad hoc networks", In proc. of the IEEE International Symposium on Parallel Architectures, Algorithms, and Networks (ISPAN), 1999.

[6] M. Bechler, H. Hof, D. Kraft, F. Pahlke, L. Wolf.,"A clusterbased security architecture for Ad-Hoc networks", In proc. of the IEEE INFOCOM, 2004.

[7] P. Brutch, C. Ko.,"Challenges in intrusion detection for wireless Ad-Hoc networks", In proc. of the IEEE Symposium on Applications and the Internet (SAINT) Workshop, 2003.

[8] S. Buchegger, J. L. Boudec,"Performance analysis of the CONFIDANT protocol (cooperation of nodes - fairness in dynamic Ad-Hoc networks)", In proc. of the ACM MOBIHOC, 2002.