Mechanism Design-Based Secure Leader Election Model for Intrusion Detection in MANET

Noman Mohammed, Hadi Otrok, Lingyu Wang, Mourad Debbabi and Prabir Bhattacharya
Computer Security Laboratory
Concordia Institute for Information Systems Engineering
Concordia University, Montreal, Quebec, Canada
Email:{no_moham, h_otrok, wang, debbabi, prabir}@ciise.concordia.ca

***Abstract*—In this paper, we study leader election in the presence of selfish nodes for intrusion detection in mobile ad hoc 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.**

***Index Terms*—Leader election, intrusion detection systems, mechanism design and MANET security.**

I. Introduction

Unlike traditional networks, the Mobile Ad hoc Networks (MANET) have no fixed chokepoints/bottlenecks where Intrusion Detection Systems (IDSs) can be deployed [3], [7]. Hence, a node may need to run its own IDS [14], [1] and cooperate with others to ensure security [15], [26]. 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 [16] or based on the connectivity [19]. 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. Next, we motivate further discussions through a concrete example.

A. Motivating Example

Figure 1 illustrates a MANET composed of ten nodes labeled from N_1 to N_{10}. These nodes are located in 5 one-hop clusters where nodes N_5 and N_9 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 N_5 and N_9 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 [16], nodes N_5 and N_9 will have equal probability, compared to others, in being elected as leaders. The nodes N_5 and N_9 will definitely be elected under the connectivity index-based approach due to their connectivity indices [19]. 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 N_2, N_5 and N_9 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 pre-

Digital Object Identifier 10.1109/TDSC.2009.22 1545-5971/$25.00 © 2009 IEEE

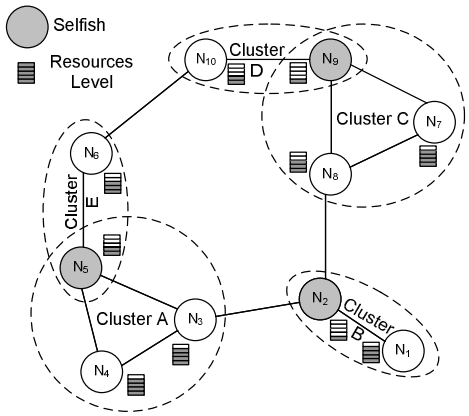

Fig. 1. An Example Scenario of Leader Election in MANET

venting nodes from behaving selfishly. To address the selfish behavior, 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 neighbor 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.

C. Possible Applications of Leader Election Scheme

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 [5] and key distribution [6], [10] 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 [34]). 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.

D. 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. Section VI provides the proof of correctness and security properties of the algorithm. Section VII presents empirical results. Section VIII reviews related work. Finally, Section IX concludes the paper and discusses our future work.

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 $C = \{c_1, c_2, \ldots, c_n\}$ where n is the number of nodes in N. We denote the election process as a function $vt_k(C, i)$ where $vt_k(C, i) = 1$ if a node i votes for a node k; $vt_k(C, i) = 0$, 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 T_{ELECT}. Thus, the model will be repeatable. For example, if $B = 25$ packet/sec and the leader 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):

$$SCF = S(C) = \min \sum_{k \in N} c_k \cdot (\sum_{i \in N} vt_k(C, i) \cdot B) \quad (1)$$

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 Section V.

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 is given in Subsection III-E.

A. Mechanism Design Background

Mechanism design is a sub-field of microeconomics and game theory [21]. Mechanism design uses game theory [25] 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 [31]. The main goal of using mechanism design [17] 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 $i \in \{1, \ldots, n\}$ has a private information, $\theta_i \in \Theta_i$, known as the agent's type. Moreover, it defines a set of strategies A_i for each agent i. The agent can choose any strategy $a_i \in A_i$ to input in the mechanism. According to the inputs $(a_i, \ldots, a_n)$ of all the agents, the mechanism calculates an output $\mathbf{o} = o(a_1, \ldots, a_n)$ and payment vector $\mathbf{p} = (p_1, \ldots, p_n)$ where $p_i = p_i(a_1, \ldots, a_n)$. The preference of each agent from the output is calculated by a valuation function, $v_i(\theta_i, \mathbf{o})$. 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 $u_i = p_i - v_i(\theta_i, \mathbf{o})$. 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 [17], every agent i has a type, θ_i. Each agent gives an input $a_i(\theta_i)$ to the mechanism. The agent chooses the strategy according to its type, where $a_i(\theta_i) = \theta_i$, which is chosen from the strategy set $\Theta = \{\text{Selfish, Normal}\}$. 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 θ_i (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:

$$p_i - v_i(\theta_i^*, \mathbf{o}) = u_i^* \geq u_i = p_i - v_i(\theta_i, \mathbf{o})$$

where θ_i^* denotes non-selfishness and θ_i denotes selfishness. Note that u_i is maximized only when p_i 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 $v_i(\theta_i^*, \mathbf{o})$? 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. A general overview of mechanism design can be found in [17], [21], [28].

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 θ_i. The type θ_i is drawn from each player's available type set Θ_i={*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 i has a utility function [21]:

$$u_i(\theta_i) = p_i - v_i(\theta_i, \mathbf{o}(\theta_i, \theta_{-i})) \quad (2)$$

where,

- θ_{-i} is the type of all the other nodes except i.
- v_i is the valuation of player i of the output $\mathbf{o} \in O$, knowing that O is the set of possible outcomes. In our case, if the node is elected then v_i is the cost of analysis c_i. Otherwise v_i is 0 since the node will not be the leader and hence there will be no cost to run the IDS.

- $p_i \in \Re$ 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, $u_i(\theta_i)$ is what the player usually seeks to maximize. It reflects the amount of benefits gained by player i if he follows a specific type θ_i. 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 $\mathbf{o} = o(\theta_1, \ldots, \theta_n)$ and a payment vector $\mathbf{p} = (p_1, \ldots, p_n)$. 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:
   a) Incentives are used once they are granted.
   b) Misbehaving nodes are catched and punished.
3) Payment design: It is needed to design the amount of incentives that will be given to the nodes based on VCG.

C. Cost of Analysis Function

During the design of the cost of analysis function, the following two problems arise: First, the energy level is considered as private and sensitive information and should not be disclosed publicly. Such a disclosure of 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 i is associated with an energy level, denoted by E_i, and the number of expected alive slots is denoted by nT_i. Based on these requirements, each node i has a power factor $PF_i = E_i/nT_i$. We introduce the set of $l-1$ thresholds $P = \{\rho_1, \ldots, \rho_{l-1}\}$ to categorize the classes as in Equation 3.

TABLE I
PS CALCULATED BY PROPOSED COST FUNCTION

PS(Percentage of sampling)	$Class_4$	$Class_3$	$Class_2$	$Class_1$
After 200 sec	55%	20%	15%	10%
After 600 sec	45%	24%	18%	13%
After 1000 sec	40%	26%	20%	14%

$$CL = \begin{cases} cl_1 & if\ PF < \rho_1 \\ cl_i & if\ \rho_{i-1} \leq PF < \rho_i;\ i \in [2, l-1] \\ cl_l & if\ PF \geq \rho_{l-1} \end{cases} \quad (3)$$

The reputation of node i is denoted by R_i. Every node has a sampling budget based on its reputation. This is indicated by the percentage of sampling, $PS_i = \frac{R_i}{\sum_{i=1}^{N} R_i}$. The c_i notation represents the cost of analysis for a single packet and E_{ids} 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 cost-of-analysis function is formulated as follows:

$$c_i = \begin{cases} \infty & if\ (E_i < E_{ids}) \\ \frac{PS_i}{PF_i} = \frac{\frac{R_i}{\sum_{i=1}^{N} R_i} \times nT_i}{E_i} & otherwise \end{cases} \quad (4)$$

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.

D. 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. Figure 2 shows the abstract model of our reputation system where each node has the following components:

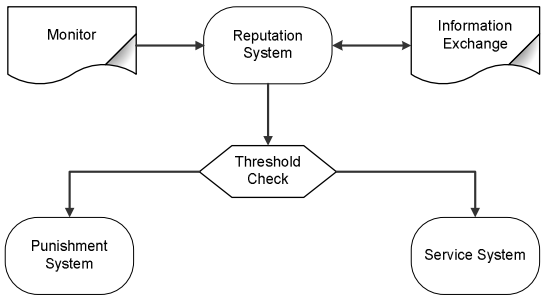

Fig. 2. Reputation System Model

- 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 [29].
- 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 [29] 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 behaviors 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.

E. 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 IDS 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.

Theorem 1: *Using the following design of payment, truth-telling is the dominant strategy:*

$$P_k = \sum_{i \in N} vt_k(C,i) B \rho_k, \text{ where} \tag{5}$$

$$\rho_k = c_k + \frac{1}{\sum_{i \in N} vt_k(C,i)} \times$$

$$[\sum_{j \in N} c_j \sum_{i \in N} vt_j(C|c_k = \infty, i) - \sum_{j \in N} c_j \sum_{i \in N} vt_j(C,i)] \tag{6}$$

Proof: Given any cost vector C, the total cost of node k can be expressed as follows:

$$T_k(C) = c_k \sum_{i \in N} vt_k(C,i)B \tag{7}$$

Using the above equation, our Social Choice Function (SCF) can be denoted as:

$$S(C) = \sum_{k \in N} c_k \sum_{i \in N} vt_k(C,i)B = \sum_{k \in N} T_k(C) \tag{8}$$

where the objective function is the sum of all players' valuations [27]. Here valuation refers to the total cost incurred by a node. According to [18], the strategy-proof payment for minimizing a function should have the following generalized form.

$$P_k = T_k(C) - S(C) + h_k(c^{-k}) \tag{9}$$

where $h_k(c^{-k})$ is an arbitrary function of c^{-k}. When $c_k = \infty$, the node is not elected due to no vote being received from its neighbors. Hence, its utility and payment will be zero. Thus,

$$h_k(c^{-k}) = \sum_{j \in N} c_j \sum_{i \in N} vt_j(C|c_k = \infty, i)B \tag{10}$$

This means,

$$P_k = c_k \sum_{i \in N} vt_k(C,i)B +$$

$$\sum_{j \in N} c_j \sum_{i \in N} vt_j(C|c_k = \infty, i)B - \sum_{j \in N} c_j \sum_{i \in N} vt_j(C,i)B \tag{11}$$

$$= \sum_{i \in N} vt_k(C,i)B\{c_k + \frac{1}{\sum_{i \in N} vt_k(C,i)} \times$$

$$[\sum_{j \in N} c_j \sum_{i \in N} vt_j(C|c_k = \infty, i) - \sum_{j \in N} c_j \sum_{i \in N} vt_j(C,i)]\} \tag{12}$$

$$= \sum_{i \in N} vt_k(C,i)B\rho_k \tag{13}$$

where,

$$\rho_k = c_k + \frac{1}{\sum_{i \in N} vt_k(C,i)} \times$$

$$[\sum_{j \in N} c_j \sum_{i \in N} vt_j(C|c_k = \infty, i) - \sum_{j \in N} c_j \sum_{i \in N} vt_j(C,i)] \tag{14}$$

This concludes the proof since the designed payment is in the generalized form of strategy-proof payment. □

In the above proof, it can be noticed that excluding a node k from election will affect only the two-hop away nodes, since new leaders may need to be elected within the two-hop neighbors of node k.

Example 1: To show how the payment is calculated and used, we consider a MANET with ten nodes as shown in Figure 3. Since our model is repeatable, we present the election process at the 10^{th} round. The reputation at the 9^{th} round is given in the first row of Table II. To elect a new leader in the 10^{th} round, the nodes will first compute their cost of analysis using the cost of analysis function given in Section III-C. The corresponding revealed cost is presented in the second row of Table II. Given the nodes' cost and network topology, node 9 will be the leader among its neighbor since it has the lowest cost of analysis. Equation 5 is used to calculate the payment of node 9, which is in the form of reputation. The payment per packet is $\rho_9 = 2 + \frac{1}{4}(8 \times 1 + 4 \times 3 - 2 \times 4) = 5$. This is because if node 9's cost is ∞ then node 10 would have voted for node 6 and node 7, 8 and 9 would have voted for node 8. Hence the total cost would have been 20 instead of 8. Therefore, the given payment of node 9 is $P_9 = \sum v_9 B\rho_9 = 4 \times 5 \times 5 = 100$ where B= 5 packets/sec is the sampling budget. After election, leader N_9 distributes the IDS sampling budget over the protected nodes N_7, N_8, N_9 and N_{10}, according to their reputation, as follows: $S = \{S_7 = \frac{90 \times 20}{470}, S_8 = \frac{160 \times 20}{470}, S_9 = \frac{110 \times 20}{470}, S_{10} = \frac{110 \times 20}{470}\}$. The details of the election algorithm will be presented in the example of Section V. □

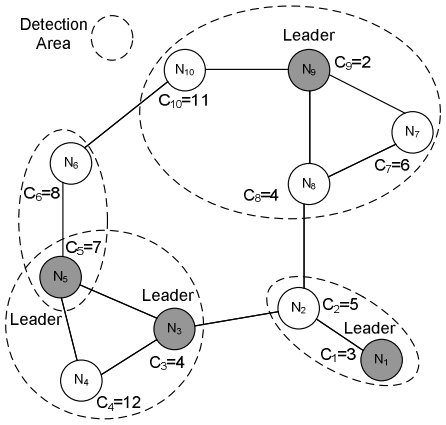

Fig. 3. An example of leader election

TABLE II
LEADER-IDS ELECTION EXAMPLE

Nodes	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}
Reputation 9^{th}	120	140	100	80	130	60	90	160	10	110
Cost of Analysis	3	5	4	12	7	8	6	4	2	11
Reputation 10^{th}	165	140	195	80	170	60	90	160	110	110

F. CDLE Payment Design

In Cluster Dependent Leader Election (CDLE), the whole network is divided into a set of clusters where a set of one-hop neighbor nodes forms a cluster. Here, we use the scheme of [20] to cluster the nodes into one-hop clusters. Each cluster then independently elects a leader among all the nodes to handle the monitoring process based on nodes' analysis-cost. Our objective is to find the most cost-efficient set of leaders that handle the detection process for the whole network. Hence, our social choice function is still as in Equation 1.

To achieve the desired goal, payments are computed using the VCG mechanism where truth-telling is proved to be dominant. Like CILE, CDLE provides payment to the elected node and the payment is based on a per-packet price that depends on the number of votes the elected node gets.

Theorem 2: *Using the following design of payment, truth-telling is the dominant strategy:*

$$P_k = \sum_{i \in N} vt_k(C,i)B\rho_k, \text{ where} \tag{15}$$

$$\rho_k = \min \sum_{j \in -n_k} v_j(\theta_j, o(\theta_j, \theta_{-j})) \quad (16)$$

Proof: According to the standard notation in mechanism design [27], the second best price is the simplest form of VCG mechanism. Here, $\sum_{j \in -n_k} v_j(\theta_j, o(\theta_j, \theta_{-j}))$ denotes the best cost excluding n_k. This is because nodes in the cluster have to select one node from the same cluster to be a leader. Unlike CILE where nodes can vote to its one-hop neighbor and then clusters are formed. □

IV. Security Analysis of the Mechanism

The main objective of our mechanism is to motivate selfish nodes and enforce them to behave normally during and after the election process. Here, we analyze the election mechanism in the presence of selfish and malicious nodes.

A. Presence of Selfish Nodes

A selfish node i will deviate from our mechanism if doing so increases its utility, u_i. Here we consider two type of untruthful revelation, namely, node i might either *under-declare* or *over-declare* the true value c_i of its cost of analysis.

Node i may under-declare its valuation function with a fake value $\hat{c}_i(\hat{c}_i < c_i)$. By under-declaring, node i pretends that it has a cheaper valuation function than reality. Since payments are designed based on VCG, playing by under-declaration will not help the node for two reasons. First, suppose the node i indeed has the lowest cost of analysis c_i, so it will win the election even by declaring its true value. In this case, reporting a lower value $\hat{c}_i$ will not benefit the node because the payment is calculated based on the second best price and does not depend on the value declared by node i. Therefore, the utility of node i remains the same because it will be the difference between the payment and the real value c_i. Second, suppose that the node i does not have the cheapest valuation function but tries to win the election by revealing a lower value $\hat{c}_i$. This will help the node i to win the election but it will also lead to a negative utility function u_i for node i, because the payment it receives will be less than the real cost of analysis. That is, the node i will have to work more than what it has paid for.

On the other hand, the node i might over-declare its valuation by revealing a fake $\hat{c}_i(\hat{c}_i > c_i)$. Following such a strategy would never make a player happier in two cases. First, if the node i indeed has the cheapest valuation function, then following this strategy may prevent the node from being elected, and therefore it will lose the payment. On the other hand, if node i still wins, then its utility remains the same since the payment does not depend on the value it reports. Second, suppose the real valuation function c_i of node i is not the lowest, then reporting a higher value will never help the node to win. Last but not least, the checkers are able to catch and punish the misbehaving leaders by mirroring a portion of its computation from time to time. A caught misbehaving leader will be punished by receiving a negative payment. Thus it discourages any elected node from not carrying out its responsibility. We can thus conclude that our mechanism is truthful and it guarantees a fair election of the most cost-efficient leader.

B. Presence of Malicious Nodes

A malicious node can disrupt our election algorithm by claiming a fake low cost in order to be elected as a leader. Once elected, the node does not provide IDS services, which eases the job of intruders. To catch and punish a misbehaving leader who does not serve others after being elected, we have proposed in [29] a decentralized catch-and-punish mechanism using random *checker* nodes to monitor the behavior of the leader.

Although not repeated here, this scheme can certainly be applied here to thwart malicious nodes by catching and excluding them from the network. Due to the presence of checkers, a malicious node has no incentive to become a leader since it will be caught and punished by the checkers. After a leader is caught misbehaving, it will be punished by receiving a negative reputation and is consequently excluded from future services of the cluster. Thus, our mechanism is still valid even in the presence of a malicious node.

V. 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 [30] 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 [13].

B. 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 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.

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 T_1. 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 */
1. **if** (received *Hello* from all neighbors) **then**
2. Send *Begin-Election* $(ID_k, cost_k)$;
3. **else if**($neighbors(k)$=Ø) **then**
4. Launch IDS.
5. **end if**

On expiration of T_1, each node k checks whether it has received all the hash values from its neighbors. Nodes from whom the *Hello* message have 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 T_2. 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 */
1. **if** $(\forall\ n \in neighbor(k),\ \exists\ i \in n : c_i \leq c_n)$ **then**
2. send $Vote(ID_k,\ ID_i, cost_{j \neq i})$;
3. *leadernode(k)*:= i;
5. **end if**

On expiration of T_2, 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 ID_k of the source node, the ID_i of the proposed leader and second least cost among the neighbors of the source node $cost_{j \neq i}$. Then 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. If node k has the least cost among all its neighbors then it votes for itself and starts timer T_3.

Algorithm 3 (Executed by Elected leader node)
/* 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.

On expiration of T_3, 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 T_{ELECT}, based on the relative reputations of the ordinary nodes. We enforce re-election every period T_{ELECT} since it is unfair and unsafe for one node to be a leader forever. Even if the topology remains same after T_{ELECT} 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 10^{th} 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 *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 N_1, N_3 and N_5 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 of Table II. □

C. Adding a new node

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 neighbors of the new

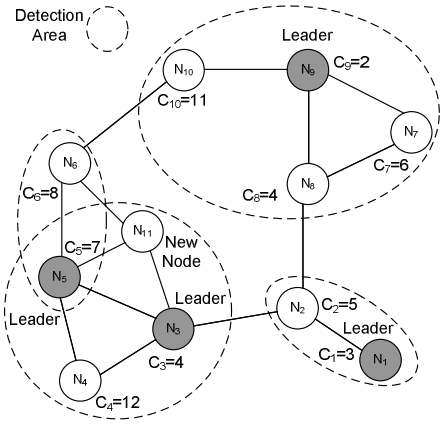

Fig. 4. A MANET after adding a new node

node, reply with a *Status* message. If the neighbor node k is a leader node, then the *Status* message contains its cost. On the other hand, if node k is an ordinary node, the *Status* message contains the ID of its leader node as in Algorithm 4.

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

On receiving the *Status* messages from the neighbors, the new node n sends *Join* to the leader node. If two of its neighbors are leaders with the same cost, then the new node can send *Join* to any of the nodes depending on its physical location (i.e; signal strength). We assume that an ordinary node have no interest to be a leader during the service time since it will not receive any payment from others. The algorithm does not make the new node as a leader for others before the new election (i.e., to reduce performance overhead). Detailed analysis is presented in Section VI. If the new node has the least cost, it can either send *Join* to the leader node or launches its own IDS. After getting the *Join* message, the leader node adds the new node to its service list and divides its budget according to nodes' reputation. We do not give any new payment to the leader as the leader node has the same budget. A problem can arise from keeping the same sampling budget for every added node. It causes the voting nodes to have less IDS service compared to what they have payed for at the election time. Thus, less sampling is offered to the voting nodes, which will ease the job of an attacker. An attacker can take an advantage from this technique only if the network is static. On the other hand, in a dynamic network, which is the case of MANET, nodes are dynamically added and removed from the network due to mobility. As a result, the average value of the budget will remain the same. Thus, the security of nodes will not be effected.

Finally, the leader node sends an *Acknowledge* message, that includes its payment, to the new node so that the new node can update its reputation table. Note that new nodes can still use their reputation value for having detection service.

Example 3: Let us consider a new node that will be added to the network in Figure 3. The resulting network is shown in Figure 4. The new node 11 is connected with node 3, 5 and 6. The cost of node 11 is 6. Node 11 sends a *Hello* message to all its neighbors. All the nodes reply with the *Status* message as in Algorithm 4. Node 11 sends *Join* message to leader node 3 as it has the least cost. Finally, leader node 3 adds node 11 in its serving list. □

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 announce a new election and updates its reputation table. On the other hand, if node n is an ordinary node then its leader node update 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. $update_{reputation}(k)$;
4. send $Begin - Election$ as in Algorithm 1;
5. **end if;**
6. **if** ($leader(k)$ = TRUE) **then**
7. **if** ($n \epsilon\ service(k)$) **then**
8. $update_{service()}$;
9. **end if;**
10. **end if;**

Example 4: Here, we consider the removal either of an ordinary node or a leader node. Considering the network in Figure 3, let us assume that node 7 has left the network or died. In other words, the links between the node 7 and others have been broken. Immediately, node 8 and 9 will be aware of the failure. On receiving the *Dead(7)* message, nodes 8 and 9 check whether node 7 is their leader or it's being served by them following the steps of Algorithm 5. As node 7 is an ordinary node, node 8 does nothing. In case of node 9, it updates its serving list. Assume now that the links of node 9 have been broken as shown in Figure 5. Then the neighboring nodes 7, 8 and 10 will discover that node 9 is their leader using Algorithm 5. Immediately, they will go for a new election by sending a *Begin-Election* message as in Algorithm 1. Thus, node 8 will become the new leader due to its lowest cost. In the case of node 10, it will launch its own IDS since it has no neighboring leader node. It cannot even join node 6, since node 6 is an ordinary node and is being served by node 5. Therefore, it has to wait for the expiration of T_{ELECT} for a new election. The resulting network is shown in Figure 5. □

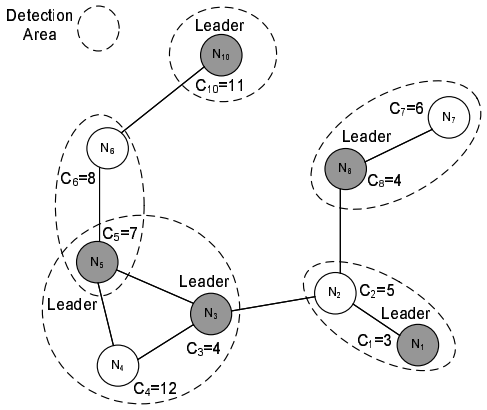

Fig. 5. A MANET after adjustment

E. 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 4^{th} step, the leader node signs the $Acknowledge$ message and others verify. Hence each normal node signs 3 messages and verifies $3|Ng_i| + 1$ messages where Ng_i is the number of neighboring nodes. On the other hand, the leader node signs 4 messages and verifies $3|Ng_i|$ messages. Note that each node must find the least cost node which requires $O(log(Ng_i))$. Therefore, each node approximately performs $O(Ng_i)$ verifications, $O(1)$ signatures and $O(log(Ng_i))$ to calculate the least cost node. Thus the computation overhead for each node is $O(Ng_i)$ + $O(1)$ + $O(log(Ng_i)) \approx O(Ng_i)$. Since our algorithm involves more verification than signing, nodes can use the public key cryptosystem of [13] 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 4^{th} step. Hence, the total communication overhead of our election algorithm is $3|Ng_i|+ 1 \approx O(Ng_i)$, where $|Ng_i|$ is the number of neighboring nodes.

3) Storage Overhead: According to the algorithm, each node maintains a *reputation-table*, *neighbors* list and two variables: *Leadernode* and *leader*. The leader node keeps an extra *service-table*. Hence, each normal node needs $|N_i|$ + $|Ng_i|$ + 2 storage and the leader node needs $|N_i|$ + $|Ng_i|$ + $|V_i|$ + 2. Knowing that $|N_i|$ is the number of nodes in the network, $|V_i|$ is the number of votes the leader node received where $|N_i| > |Ng_i| > |V_i|$. Therefore, the total storage for each node is in the order of $O(N_i)$.

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 in [20].

VI. 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 nodes 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 [29]. After receiving *Hello* and *Begin-Election* messages from all the neighbor nodes within $(T_1 + T_2)$ time, nodes are sorted according to their cost of analysis. By executing Algorithm 2, each node sets its variable *leadernode(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 T_{ELECT} to participate in the new election. □

B. 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 signed

by the source node and verified by others. Thus, *integrity* is provided and the possibility of altering the Vote messages is prevented. Moreover, the source *authentication* is granted since PKI allows the recipient to verify the identity of the sender through its signature. Additionally, the *freshness* of messages is provided through TESLA that synchronizes the clocks among the nodes and consequently avoids reply attacks. Finally, to avoid nodes from not continuing the execution of the algorithm after discovering their loss, a *fairness* property must be given to avoid such a flaw. This will be granted through excluding the nonparticipating nodes from having the cluster's services.

The algorithm is cheat-proof: We claim that our algorithm is cheat-proof because a node, which does not have the least cost of analysis among its neighbors cannot be elected as a leader. To prevent a node from revealing its cost after observing others, we design our cost revaluation procedure in two rounds: First, each node computes the hash of its cost where all the nodes use the same hash function. Then, nodes broadcast the hash value using the *Hello* message. Second, upon receiving the hash values from all the nodes, each node reveals its cost of analysis. Since the hash values are already available, every node verifies the cost of analysis of the other nodes. In this way, we are able to prevent cheating by declining the revelation of the announced cost of analysis value or changing it later on.

VII. Simulation results

In this section, we evaluate the performance of our model (CILE) with respect to random and connectivity models. We simulate the schemes using Network Simulator 2 (NS2).

A. Performance Metrics

The main objective of our simulation results is to study the effect of node selection for IDS on the life of all nodes. To show the negative impact of selfish node, we conducted two experiments: *Time taken for the first node to die and percentage of packet analysis.* Besides, we use the following metrics to evaluate our algorithm against others: *Percentage of alive nodes, energy level of nodes, percentage of leader node, average cluster size, maximum cluster size and number of single node clusters.* Our experiments have been conducted in both static and dynamic networks. For a static network, we compare our algorithm with both random and connectivity models, while for dynamic network, we only compare with connectivity model since we believe that the random model will perform almost the same as in static one. Our experimental results have a 95% confidence and a 5% precision.

B. Simulation Environment

To implement the models, we modify the energy model to measure the effect of running IDS. Initially, we randomly assign 60 to 100 joules to each node. We assume that the energy required for running the IDS for one time slot as 10 joules. We ignore the energy required to live and transmit packets to capture the silent aspect of the problem. We set the transmission radius of each node to 200 meters. Two nodes are considered as neighbors if their Euclidean distance is less than or equal to 200 meters.

TABLE III
Simulation Parameters

Parameter	Value
Simulation Time	2000 seconds
Simulation Area	500×500 m
Number of Nodes	$20, 30, 40, 50$
Transmission Range	200 m
Movement Model	Random Waypoint Model
Maximum Speed	15 meters/sec
Pause Time	200 s
Traffic Type	CBR/UDP
Packet Rate	4 packets/sec
T_{ELECT}	20 sec

Besides, we deploy different number of nodes, which varies from 20 to 50 in an area of 500×500 square meters. It helps us to measure the performance of the nodes from sparse networks to dense networks. Table III summarizes our simulation parameters.

C. Experimental Results

Nodes can behave selfishly before and after the election. A node shows selfishness before election by refusing to be a leader. On the other hand, selfishness after election is considered when nodes misbehave by not carrying out the detection service after being a leader. Both kinds of selfishness have a serious impact on the normal nodes. To show the seriousness and impact of selfishness before election on resource consumption, Figure 6.(a) depicts the impact of selfish nodes on the life of normal nodes. The result indicates that the normal nodes will carry out more duty of intrusion detection and die faster when there are more selfish nodes. Figure 6.(b) shows the impact of selfishness after election on security. We consider the presence of 20% of selfish nodes out of 10 nodes. As selfish nodes do not exhaust energy to run the IDS service, it will live longer than the normal nodes. Thus, the more the time goes, the more the chances that the selfish node will be the leader node. Hence, the percentage of packet analysis decreases with time, which is shown in Figure 6.(b). This is a severe security concern since fewer packets are analyzed.

In Figure 6.(c), we compare our model with the other two models to show the percentage of alive nodes with respect to time. We simulate our model in a network of 10 mobile nodes as shown in Figure 3 with the presence of 20% of selfish nodes. We consider nodes 4 and 7 to be selfish and study their impact on our model, random and connectivity models with no mobility. The nodes repetitively elect a set of leaders every T_{ELECT} seconds. The election is based on the proposed scheme. The experiment indicates that our model results in a higher percentage of alive nodes, in contrast to other models. On the other hand, the random model elects leaders without considering the energy level and leads nodes with low energy to die fast. Finally, the connectivity model elects leaders based on their number of connections. In the case of static scenarios, the model elects the same node repeatedly, which causes the normal nodes to die very fast. In our model, the node that has the least cost of analysis becomes the leader. In this way, all the nodes can keep a balance of their energy level with time.

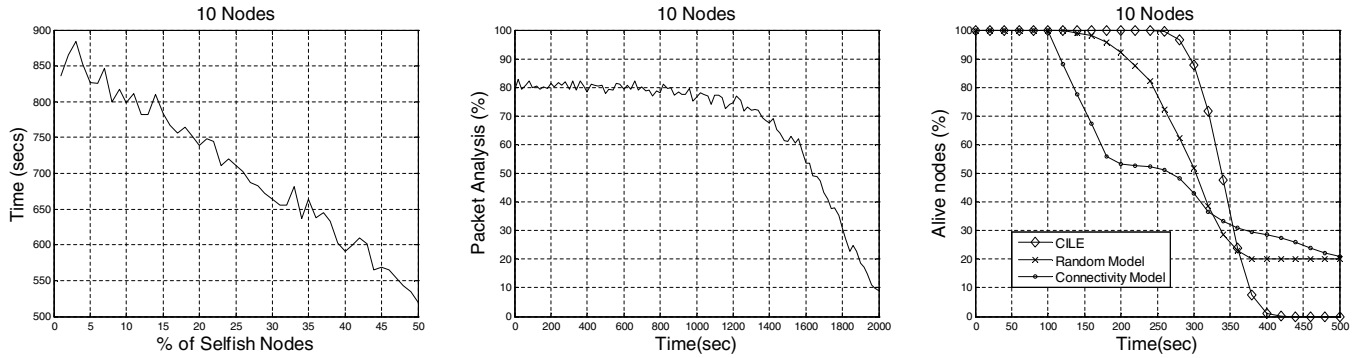

Fig. 6. (a) Time for Normal Node to Die (b) Percentage of Packet Analysis (c) Percentage of Alive Nodes

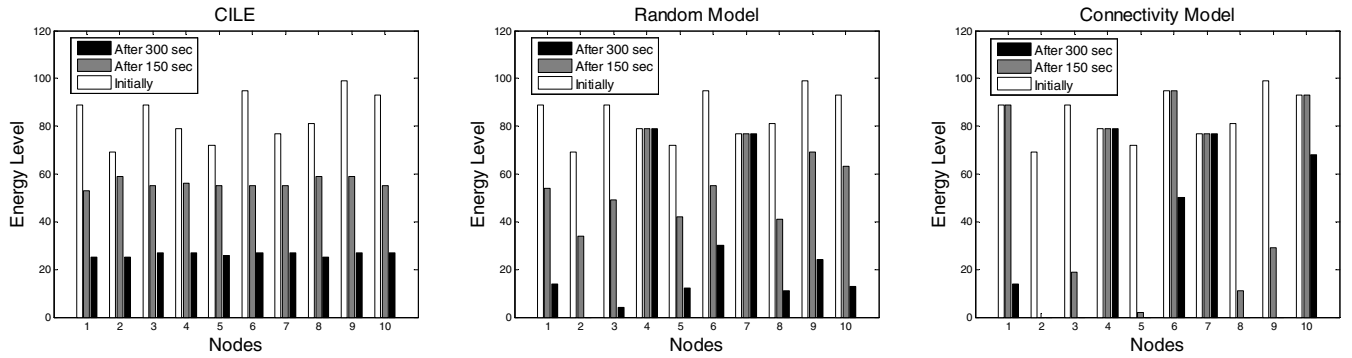

Fig. 7. (a) Energy Level of Our Model (b) Energy Level of Random Model (c) Energy Level of Connectivity Model

Hence, all the nodes will live long and die at the same time which is clearly shown in Figure 7. Figure 7.(a) indicates that our model is able to balance the resource consumption among all nodes. On the other hand, the random (Figure 7.(b)) and connectivity (Figure 7.(c)) models result in unbalanced energy consumption and several dead nodes.

Now, we evaluate the performance of our algorithm in a dynamic network for different number of nodes from 20 to 50. The simulation parameters are mentioned in Table III. We compare our model only with the connectivity model since we believe that the expected performance of the random model will be close to the one given with low mobility (static network). Figure 8 shows that more nodes are alive in our model compared to the connectivity one. As the number of nodes increases, the life of nodes also increases since there are more nodes to act as leaders. Thus, the detection service is distributed among the nodes which prolongs the live time of the nodes in MANET.

Last but not least, we compare some of the cluster characteristics of our model with those of the connectivity model. Figure 9.(a) shows the percentage of the leader nodes. The percentage of leaders for our model is less as compared to those of the connectivity model that saves the energy of nodes. Figure 9.(b) compares the average cluster size of both the models for different number of nodes. Our model has a higher average cluster size than the other one. This proves that our model is able to uniformly distribute the load of the leaders. Figure 10.(a) illustrates the size of the maximum cluster. The maximum cluster size for both models is increasing with the number of nodes. For our model, the maximum cluster size is less and thus avoid many problems; such as, message collisions, transmission delays and etc. This could also improves the detection probability since more number of packets is analyzed per node compared to the other model. Moreover, our model is able to reduce the number of single node clusters as the density of nodes is increasing. This shown in Figure 10.(b).

From these experiments, we can conclude that our model is able to balance the IDS resource consumption in the presence of selfish nodes. Moreover, it is able to reduce single node clusters and also the maximum cluster size. Besides, it achieves more uniform clusters with less leader nodes. Finally, these properties improve the efficiency of the IDS on detecting intrusions since the sampling budget is distributed over less number of nodes compared to the other model.

VIII. Related Work

This section reviews related work on intrusion detection in MANET, the application of mechanism design to networks and application of leader election scheme to routing and key distribution.

A. Intrusion Detection Systems in MANET

The difference between wired infrastructure networks and mobile ad hoc networks raises the need for new IDS models that can handle new security challenges [23]. Due to the security needs in MANET, a cooperative intrusion detection model has been proposed in [35], where every node participates in running its IDS in order to collect and identify possible intrusions. If an anomaly is detected with a weak

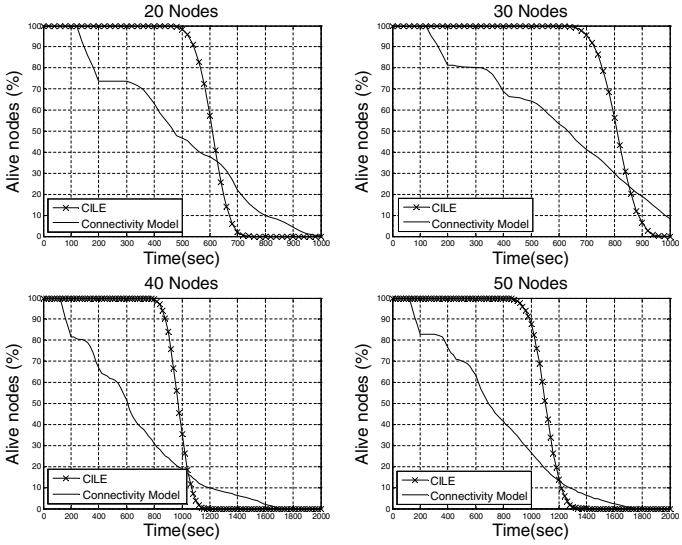

Fig. 8. Percentage of Alive Nodes

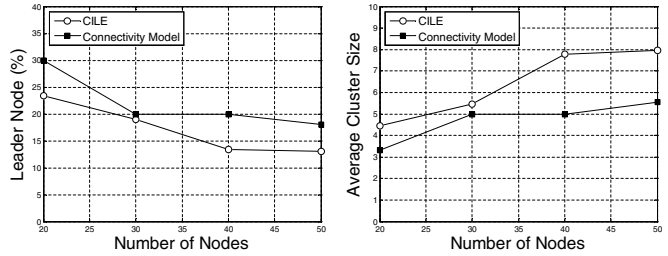

Fig. 9. (a) Percentage of Leader Node (b) Average Cluster Size

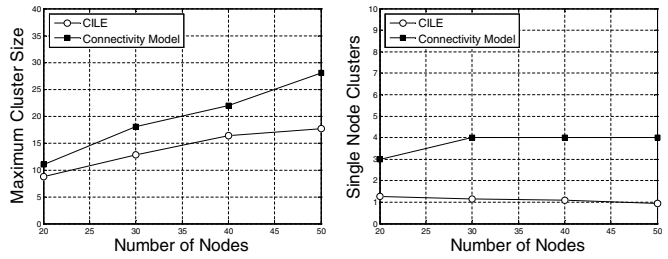

Fig. 10. (a) Size of Maximum Cluster (b) Number of Single Node Clusters

evidence, then a global detection process is initiated for further investigation about the intrusion through a secure channel. An extension of this model was proposed in [16], where a set of intrusions can be identified with their corresponding sources. Moreover, the authors address the problem of run-time resource constraints through modeling a repeatable and random leader election framework. An elected leader is responsible for detecting intrusions for a predefined period of time. Unlike our work, the random election scheme does not consider the remaining resources of nodes or the presence of selfish nodes. In [19], a modular IDS system based on mobile agents is proposed and the authors point out the impact of limited computational and battery power on the network monitoring tasks. Again, the solution ignores both the difference in remaining resources and the selfishness issue. To motivate the selfish nodes in routing, CONFIDANT [8] proposes a reputation system where each node keeps track of the misbehaving nodes. The reputation system is built on the negative evaluations rather than positive impression. Whenever a specific threshold is exceeded, an appropriate action is taken against the node. Therefore, nodes are motivated to participate by punishing the misbehaving ones through giving a negative reputation. As a consequence of such a design, a malicious node can broadcast a negative impression about a node in order to be punished. On the other hand, CORE [22] is proposed as a cooperative enforcement mechanism based on monitoring and reputation systems. The goal of this model is to detect selfish nodes and enforce them to cooperate. Each node keeps track of other nodes cooperation using reputation as a metric. CORE ensures that misbehaving nodes are punished by gradually excluding them from communication services. In this model, the reputation is calculated based on data monitored by local nodes and information provided by other nodes involved in each operation. In contrast to such passive approaches, our solution proactively encourage nodes to behave honestly through computing reputations based on mechanism design. Moreover, it is able to punish misbehaving leaders through a cooperative punishment system based on cooperative game theory [29]. In addition to this, a non-cooperative game is designed to help the leader IDS to increase the probability of detection by distributing the node's sampling over the most critical links.

B. Application of Mechanism Design

As a sub-field of microeconomics and game theory, mechanism design has received extensive studies in microeconomics for modeling economical activities, such as auctions [21]. Nisan and Ronen applies mechanism design for solving the least-cost path and task scheduling problem [27]. Distributed mechanism design based on VCG is first introduced in a direct extension of Border Gateway Protocol (BGP) for computing the lowest-cost routes [11]. Moreover, in [12] the authors outlined the basics of distributed mechanism design and reviewed the results done on multi-cast cost sharing and inter-domain routing. Mechanism design has been used for routing purposes in MANETs, such as a truthful adhoc-VCG mechanism for finding the most cost-efficient route in the presence of selfish nodes [2]. In [9], the authors provide an incentive compatible auction scheme to enable packet forwarding services in MANETs using VCG; a continuous auction process is used to determine the distribution of bandwidth and incentives are given as monetary rewards. To our best knowledge, this work is among the first efforts in applying mechanism design theory to address the security issues in MANETs, in particular, the leader-election for intrusion detection. This paper is the extension of [24] where we presented the leader election mechanism in a static environment without addressing different performance overhead.

C. Leader Election applications

Distributed algorithms for clustering and leader election have been addressed in different research work [20], [4], [33], [32]. These algorithms can be classified into two categories [32]: Cluster-first or leader-first. In the cluster-first approach [20], a cluster is formed and then the nodes belonging to that cluster elect a leader node. In the leader-first approach [4], a set of leader nodes is elected first then the other nodes are assigned to different leader nodes. Some of the methods

assume there exist a weight associated with each node [5] or there exist a trusted authority [33] to certify each node's metric (weight) which is used to elect a leader. We consider these assumptions as quite strong for MANET. Our model is able to run in a clustered and non-clustered networks where we are able to perform better results with respect to different performance metrics.

IX. 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.

References

[1] T. Anantvalee and J. Wu. A survey on intrusion detection in mobile ad hoc networks. *Wireless/Mobile Network Security*, 2006.

[2] L. Anderegg and S. Eidenbenz. Ad hoc-VCG: A truthful and cost-efficient 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 and 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, and L. Wolf. A cluster-based security architecture for ad hoc networks. In *proc. of the IEEE INFOCOM*, 2004.

[7] P. Brutch and 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 and 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.

[9] K. Chen and K. Nahrstedt. iPass: An incentive compatible auction scheme to enable packet forwarding service in MANET. In *proc. of the International Conference on Distributed Computing Systems*, 2004.

[10] B. DeCleene, L. Dondeti, S. Griffin, T. Hardjono, D. Kiwior, J. Kurose, D. Towsley, S. Vasudevan, and C. Zhang. Secure group communications for wireless networks. In *proc. of the IEEE Military Communications Conference (MILCOM)*, 2001.

[11] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP based mechanism for lowest-cost routing. In *proc. of the ACM symposium on Principles of distributed computing (PODC)*, 2002.

[12] J. Feigenbaum and S. Shenker. Distributed algorithmic mechanism design: Recent results and future directions. In *proc. of the AMM International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM)*, 2002.

[13] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In *proc. of the Cryptographic Hardware and Embedded Systems (CHES)*, 2004.

[14] S. Gwalani, K. Srinivasan, G. Vigna, E. M. Beding-Royer, and R. Kemmerer. An intrusion detection tool for AODV-based ad hoc wireless networks. In *proc. of the IEEE Computer Security Applications Conference (CSAC)*, 2004.

[15] Y. Hu, A. Perrig, and D. B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *proc. of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, 2002.

[16] Y. Huang and W. Lee. A cooperative intrusion detection system for ad hoc networks. In *proc. of the ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2003.

[17] L. Hurwicz and S. Reiter. *Designing Economic Mechanisms*. Cambridge University Press, 1^{st} edition, 2008.

[18] J.Green and J.Laffont. *Incentives in Public Decision-Making*. Springer Netherlands, USA, 1996.

[19] O. Kachirski and R. Guha. Efficient intrusion detection using multiple sensors in wireless ad hoc networks. In *proc. of the IEEE Hawaii International Conference on System Sciences (HICSS)*, 2003.

[20] P. Krishna, N. H. Vaidya, M. Chatterjee, and D. K. Pradhan. A cluster-based approach for routing in dynamic networks. In *proc. of the ACM SIGCOMM Computer Communication Review*, 1997.

[21] A. Mas-Colell, M. Whinston, and J. Green. *Microeconomic Theory*. Oxford University Press, New York, 1995.

[22] P. Michiardi and R. Molva. Analysis of coalition formaton and cooperation strategies in mobile adhoc networks. *Journal of Ad hoc Networks*, 3(2):193 – 219, 2005.

[23] A. Mishra, K. Nadkarni, and A. Patcha. Intrusion detection in wireless ad hoc networks. *IEEE Wireless Communications*, 11(1):48 – 60, 2004.

[24] N. Mohammed, H. Otrok, L. Wang, M. Debbabi, and P. Bhattacharya. A mechanism design-based multi-leader election scheme for intrusion detection in manet. In *proc. of the IEEE Wireless Communications & Networking Conference (WCNC)*, 2008.

[25] P. Morris. *Introduction to Game Theory*. Springer, 1^{st} edition, 1994.

[26] P. Ning and K. Sun. How to misuse AODV: A case study of insider attacks against mobile ad-hoc routing protocols. In *proc. of the IEEE Information Assurance Workshop*, 2003.

[27] N. Nisan and A. Ronen. Algorithmic mechanism design. In *Games and Economic Behavior*, pages 129–140, 1999.

[28] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 1^{st} edition, 2007.

[29] H. Otrok, N. Mohammed, L. Wang, M. Debbabi, and P. Bhattacharya. A game-theoretic intrusion detection model for mobile ad-hoc networks. *Journal of Computer Communications*, 31(4):708 – 721, 2008.

[30] A. Perrig, R. Canetti, D. Tygar, and D. Song. The TESLA broadcast authentication protocol. *RSA Cryptobytes*, 5(2):2 – 13, 2002.

[31] J. Shneidman and D. Parkes. Specification faithfulness in networks with rational nodes. In *proc. of the ACM Symposium on Principles of Distributed Computing*, 2004.

[32] K. Sun, P. Peng, P. Ning, and C. Wang. Secure distributed cluster formation in wireless sensor networks. In *proc. of the IEEE Computer Security Applications Conference (ACSAC)*, 2006.

[33] S. Vasudevan, B. DeCleene, N. Immerman, J. Kurose, and D. Towsley. Leader election algorithms for wireless ad hoc networks. In *proc. of the IEEE DARPA Information Survivability Conference and Exposition (DISCEX III)*, 2003.

[34] S. Vasudevan, J. Kurose, and D. Towsley. Design and analysis of a leader election algorithm for mobile ad hoc networks. In *proc. of the IEEE International Conference on Network Protocols (ICNP)*, 2004.

[35] Y. Zhang and W. Lee. Intrusion detection in wireless ad-hoc networks. In *proc. of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2000.

Noman Mohammed received the M.A.Sc. degree in Information Systems Security from Concordia University, Canada in 2008 and the B.S. degree in Computer Science from North South University, Bangladesh in 2005. He is currently pursuing his Ph.D. in Computer Science at Concordia University with the Alexander Graham Bell Canada Graduate Scholarship from the Natural Sciences and Engineering Research Council of Canada (NSERC). His research interests include data privacy, economics of network security and secure distributed computing.

Hadi Otrok holds a Ph.D. in Electrical and Computer Engineering (ECE) from Concordia University, Montreal, Canada. During his Ph.D., he worked on network security, more specifically, on intrusion detection systems in Mobile Ad hoc Networks. He used game theory and mechanism design to formulate and solve intrusion detection problems. He received his Masters degree from Lebanese American University (LAU) where he worked on security testing and evaluation of cryptographic algorithms. Currently, he holds a Post-Doctoral position at Ecole de Technologie Superieure (University of Quebec). He is working on secure resource allocation for virtual private networks. His research interests are mainly on network security, application security, and middleware security. He is serving as technical program committee member for different international conferences and reviewer for prestigious international journals.

Lingyu Wang is an Assistant Professor of the Concordia Institute for Information Systems Engineering (CIISE) at Concordia University, Canada. He received his Ph.D. degree in Information Technology from George Mason University, USA. His current research interests include database security, data privacy, vulnerability analysis, intrusion detection, and security metrics. His research has been supported in part by the Discovery Grants from the Natural Sciences and Engineering Research Council of Canada (NSERC) and by Fonds de recherche sur la nature et les technologies (FQRNT).

Mourad Debbabi received the Ph.D. and M.Sc. degrees in computer science from Paris-XI Orsay, University, France. He is currently a Full Professor and the Director of the Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Quebec, Canada. He holds the Concordia Research Chair Tier I in Information Systems Security. He is also the Vice-President of the National Cyber Forensics Training Alliance (NCFTA Canada). He is the founder and one of the leaders of the Computer Security Laboratory (CSL) at Concordia University. He is the Specification Lead of four Standard JAIN (Java Intelligent Networks) Java Specification Requests (JSRs) dedicated to the elaboration of standard specifications for presence and instant messaging. In the past, he served as Senior Scientist at the Panasonic Information and Network Technologies Laboratory, Princeton, New Jersey, USA; Associate Professor at the Computer Science Department of Laval University, Quebec, Canada; Senior Scientist at General Electric Research Center, New York, USA; Research Associate at the Computer Science Department of Stanford University, California, USA; and Permanent Researcher at the Bull Corporate Research Center, Paris, France. He published more than 150 research papers in journals and conferences on computer security, formal semantics, Java security and acceleration, cryptographic protocols, malicious code detection, programming languages, type theory and specification and verification of safety-critical systems. He supervised to successful completion more than 50 graduate students at M.Sc. and Ph.D. levels.

Prabir Bhattacharya (SM'92, F'02) received the D.Phil. degree in 1979 from the University of Oxford, U.K and did his undergraduate studies at the University of Delhi, India. He is currently a Full Professor at the Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Quebec, Canada where he holds a Canada Research Chair, Tier 1. During 1986-99, he served at the Department of Computer Science and Engineering, University of Nebraska, Lincoln, USA where he was a Full Professor from 1994. During 1999-2004, he worked at the Panasonic Information Technologies Laboratory in Princeton, NJ, USA as a Principal Scientist and a Project Leader. He is a Fellow of the IEEE, the International Association for Pattern Recognition and the Institute for Mathematics and Its Applications, UK. During 2006-07 he served as the Associate Editor-in-Chief of the IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics). He is currently an Associate Editor of six journals including the IEEE Transactions on SMC-B. He was a Distinguished Visitor of the IEEE Computer Society during 1996-1999. In 2008 he received an Outstanding Service award from the IEEE Systems, Man and Cybernetics Society. He has authored or co-authored about 236 publications including 100 journal papers, and co-authored three books; also he holds four US Patents.