MECHANISM DESIGN-BASED LEADER ELECTION SCHEME FOR INTRUSION DETECTION IN MANET

NOMAN MOHAMMED

A THESIS

IN

THE DEPARTMENT

OF

CONCORDIA INSTITUTE FOR INFORMATION SYSTEMS ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF APPLIED SCIENCE IN INFORMATION SYSTEMS

SECURITY

CONCORDIA UNIVERSITY

MONTRÉAL, QUÉBEC, CANADA

MAY 2008

Your file Votre référence
ISBN: 978-0-494-42520-6
Our file Notre référence
ISBN: 978-0-494-42520-6

Abstract

Mechanism Design-Based Leader Election Scheme for Intrusion Detection in MANET

Noman Mohammed

We study the leader election in the presence of selfish nodes for intrusion detection systems (IDS) in a mobile ad hoc network (MANET). 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, without incentives for serving others, a node may behave selfishly by lying about its remaining resource and avoiding being elected. We present a solution based on mechanism design theory. More specifically, we design a scheme for electing cluster leaders that have the following two advantages: First, the collection of elected leaders is the optimal in the sense that the overall resource consumption will be balanced among all nodes in the network overtime. Second, the scheme provides the leaders with incentives in the form of reputation so that nodes are encouraged to honestly participate in the election process. The design of such incentives is based on the Vickrey, Clarke, and Groves (VCG) model by which truth-telling is the dominant strategy for each node. Simulation results show that our scheme can effectively prolong the overall lifetime of IDS in MANET and balance the resource consumptions among all the nodes.

Acknowledgments

I would like to express my gratitude to Almighty God for granting me the ability and opportunity to write this thesis.

I would like to thank my faculty advisors, Dr. Mourad Debbabi and Dr. Lingyu Wang for giving me the opportunity to work under their supervision. I am very grateful to them for their valuable suggestions and guidance throughout the preparation of this thesis. I am also very grateful to my committee members Dr. A. Agarwal, Dr. A. Ben Hamza and Dr. B. Zhu for providing valuable advice. I would also like to thank Hadi Otrok who performed a multi role: a friend, a colleague and a guide. Without his enormous support I would not have been able to finish this thesis.

Lastly, I am grateful to my friends for their assistance, to my parents for their endless support and to my wife for her constant encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A mobile ad hoc network (MANET) is a self-configuring network in which each node acts as both a receiving node and a relaying node that transmits packets to other nodes within its transmission range. Unlike other networks, MANET is distributive in nature and therefore it lacks fixed centralized infrastructure. In MANET, decision-making, key-distribution and routing are usually decentralized and many of them depend on the cooperative participation of all nodes [72]. Besides, the topology of the network is always changing since nodes are constantly moving in and out of the network. In addition, these nodes have limited battery power and bandwidth along with computationally limited hardware.

Applications of MANET are diverse [7]. For example the auto configuration capability of MANETs may be useful at war since missions are likely to take place in remote areas where network infrastructures are not available. Furthermore, MANETs are relevant in situations where wired communications have been destroyed or traditional networks are congested, such as in a rescue mission after a natural disaster. MANETs could also be used in remote regions where telecommunications are scarce or there is no internet infrastructure. Finally, laptops, Personal Digital Assistants (PDAs), cell phones and other devices can also form MANETs for daily uses.

1.1 Motivation

MANET is particularly susceptible to many attacks ranging from passive eavesdropping to active interfering due to its unique characteristics. The encryption techniques and firewalls are known to be insufficient for securing MANET against all kinds of attacks. Hence an intrusion detection system (IDS) is needed as a second line of defense for detecting intrusions and consequently triggering an appropriate response. Intrusion detection is the process of monitoring the system or network by looking into the occurring events and searching for intrusions. According to the type of data sources, IDS is typically categorized as *host-based* (HIDS) and *network-based* (NIDS). The former aims to detect intrusions targeting the system by analyzing system audit data, whereas the latter detects intrusions in the network by examining transmitted packets. The IDS techniques for wired infrastructure network which have been developed over the years cannot be directly applied to MANET due to some major differences between the two networks. The unique characteristics of MANET such as mobility and open wireless medium make IDS deployment harder. Unlike traditional networks, the MANET have no fixed chokepoints/bottlenecks where IDS can be deployed [7, 12]. Hence, a node may need to run its own IDS [3, 33] and cooperate with others to ensure security [36, 62]. 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 [39] or based on the connectivity [45]. 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. Misbehaving nodes shall deviate from telling the truth about their resources if that could maximize their own benefits. Moreover, even when all nodes can truthfully reveal their resources level, it remains a challenging issue to elect an optimal collection of leaders to balance the overall resource consumption without flooding the network. This motivated us to work on developing a leader election mechanism which will balance the resources among nodes, taking into consideration selfishness.

1.2 Thesis Contributions

In this thesis, we propose a solution for balancing the resource consumption of IDSs among all nodes while preventing nodes from behaving selfishly. To develop our leader election:

- We propose a cost of analysis function to protect nodes' sensitive information (resources level) and ensure the contribution of every node on the election process.
- We design incentives in the form of reputation to encourage nodes to honestly participate in the election scheme. 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 priority in packet forwarding and to build a trust environment.
- We calculate the payment (reputation) based on a classical mechanism design model, namely, Vickrey, Clarke, and Groves (VCG) [55]. The model guarantees that truth-telling is always the dominant strategy for every node during each election phase.
- We devised a leader election algorithm to elect the globally optimal cost-efficient leaders, taking into consideration the possibility of cheating and security flaws. The algorithm decreases the percentage of leaders, single node clusters, maximum cluster size and increases average cluster size.
- 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.

In summary, the main contribution of this paper is two-fold. First, we identify the important issue of balancing resource consumption of IDSs among nodes, and we provide a solution to this issue through the design and analysis of a new election scheme. Second, we study the threat of selfish nodes to leader election, and we provide a solution by adapting a mechanism design model named VCG.

1.3 Outline of the Thesis

In chapter 2, we discuss several attacks to demonstrate the need for IDS in MANET. We also illustrate the challenges in the deployment of traditional IDS. Then, we present several IDS architectures for MANET and point out leader election as a key concept for cooperative IDS. Moreover, we discuss other applications of leader election along with current solutions and existing problems. We further motivate the problem with an illustrative example. Finally, we provide the basics of game theory and mechanism design. We briefly discuss some of the important concepts such as, dominant strategy, utility function, VCG mechanism, etc. These concepts help to better understand our leader election mechanism since it is based on mechanism design theory. In chapter 4, we formally define our problem statements. We describe our leader election mechanism where the cost of analysis function, reputation model and payment designs are given. Finally we analyze our mechanisms against selfish and malicious nodes. Chapter 5 devises the election algorithm needed to handle the election process. We also provide the proof of correctness and security properties of the algorithm along with performance analysis. In chapter 6, we provide a comprehensive simulation to evaluate the performance of our model. We compare our model with random and connectivity model against different performance matrices. Lastly, chapter 7 provides a summary of our model and proposes some possible continuation based on this thesis.

Chapter 2

Literature Review

In this chapter, the necessity of intrusion detection system is given through presenting some unique attacks targeting a MANET. We then discuss why traditional IDS is not suitable for MANET followed by the proposed IDS architectures, where we point out that leader election is a significant issue for IDS in MANET. Moreover, we show the limitations of the present solutions with an illustrative example. Finally, we briefly explain the basics of mechanism design since our proposed approach for leader election is based on it.

2.1 Attacks Unique to MANET

Security is one of the major challenges in adopting MANETs in different applications [12, 33]. The decentralized nature and dynamic topology of MANETs are unique characteristics that render them more vulnerable to attacks than network architectures. Since all the nodes in a MANET are peers, potential damages caused by a compromised node can be significantly higher than in traditional networks. An intruder that compromises a mobile node can destroy the communication by broadcasting false routing information, providing incorrect link state information, and overflowing other nodes with unnecessary routing traffic. Ultimately, this would lead to a Denial of Service (DoS) attack on the whole network. Moreover, MANET is particularly susceptible to many attacks due to their open medium. On the other hand, the detection and prevention techniques are seriously constrained by the more stringent resources constraints. To better understand the threats in MANET, below we present the taxonomy of some known attacks, which are unique to MANET. Our comparative study of these attacks demonstrates the need for IDS in MANET.

Rushing Attack The *Rushing* attack is an effective Denial of Service (DoS) attack against most (secure) routing protocols [37]. In an on-demand routing protocol, a node wishing to find a route to another node usually broadcasts ROUTE REQUEST packets. In order to limit the performance overhead, existing protocols usually forward only one ROUTE REQUEST from a route discovery, which is taken advantage of by this attack. A relatively weak attacker can either keep the network interfaces of his neighboring nodes full or forward packets faster than others so to increase the probability that routes including him will be chosen by communicating nodes. Existing (secure) routing protocols, such as AODV [11], Adriadne [36], ARAN [69] and SAODV [86], are vulnerable to rushing attack.

Flooding Attack The *Ad Hoc Flooding* attack [33, 39, 81, 84] is launched at the network layer in which the attacker congests the network by flooding it with Route Request (RREQ) packets. The attacker could also choose random IP addresses in case s/he does not know the network topology. After selecting these IP addresses, the attacker will launch massive RREQ messages without taking into consideration the RREQ_RATELIMIT, and without waiting for a route reply (RREP). The routing tables of the attacked nodes will be exhausted, which will deny these nodes from receiving any new route requests.

Packet Dropping Attack The *Packet Dropping* attack can be caused by two kinds of nodes: Malicious [33,39,54] and selfish [57,58,72,83]. The former refers to compromised node, whereas the latter refers to node who wants to save energy by not forwarding others' packets. This attack is due to the nature of MANETs where cooperation between nodes is needed to forward their routing packets. This fact renders MANETs especially vulnerable to such a denial of service attack.

Sybil Attack In *Sybil* attack [87,89,90], an attacker forges multiple, nonexistent identities in the MANET. Manipulating different identities in a network enables a malicious node to control services, which are otherwise accessible only by coordination between several distinct nodes. Roughly speaking, this attack allows a malicious node to have large influence on the services in the network. Sybil Attack is performed when one entity presents multiple identities to one or more other entities.

Invisible Node Attack The authors in [4] give a strict definition of the *Invisible Node* attack: *In any protocol that depends on identification for any functionality, any node that effectively participates in that protocol without revealing its identity is an invisible node and the action and protocol impact is termed an Invisible Node Attack.* This attack is not the Man-in-the-Middle (MITM) attack, but a special type of MITM attack [4, 53]. In MANET, this attack is possible due to the fact that communication is performed by means of broadcasting [29]. A malicious node M that stands between nodes A and B can just silently repeat messages from A to B and messages from B to A, making A and B think they are communicating to each other directly. Cryptographic mechanisms are of no help in preventing this attack.

Wormhole Attack The *Wormhole* attack is a severe attack in MANETs that is particularly challenging to defend against [19, 34, 38, 52, 67, 71]. The wormhole attack can take place without compromising any nodes, authenticity of communication, or confidentiality. During a wormhole attack, a malicious node uses a path outside the network to route messages to another compromised node. Wormhole attacks are hard to discover because the path that is used to pass on information is usually not part of the actual network. Interestingly, a wormhole lowers the time taken for a packet to reach its destination which in itself is not harmful. However, as wormholes fake a route that is shorter than reality, it can confuse routing mechanisms, which rely on the knowledge about distances between nodes. The wormhole attack can be initiated in two modes [47]. In the *hidden mode*, the attackers suppress their identities and remain hidden from others. In the hidden mode the attackers act as two simple transceivers and do not require any cryptographic keys to launch the attack. In the participation mode, the attackers use valid cryptographic keys and participate in the routing as legitimate nodes.

Black Hole Attack In a *Black Hole* attack, a malicious node impersonates a destination node by sending a spoofed route reply packet to a source node that initiates a route discovery. By doing this, a malicious node can deprive the traffic from a source node to a real destination. First, the attack can occur when the malicious node on the path directly attacks the data traffic passing through it. This black hole attack [54] takes place when the intermediate node drops the packets after the agreement to forward the packets. It can happen for various reasons, such as selfishness, packet overload, link broken or maliciousness. Two mechanisms have been introduced in [54] to defend the attack, namely, the watchdog and the pathrater. A second type of black hole attack [40,41,77] is to attack routing control traffic. This happens when a source node broadcasts a RREQ packet in search of destination. A malicious node, without checking in its cache table, sends RREP packet to the source node claiming the fresh route to the destination. If this packet reaches the source node before any other RREP packet, a false route is created. Thus, a malicious node can absorb data packets without forwarding them to the destination.

Sinkhole Attack A *Sinkhole* attack [22, 49, 61] is actually a more sophisticated version of the black hole attack. It works by making a malicious node look especially attractive to the surrounding nodes with respect to routing. A malicious node finds out and uses the loopholes in a routing protocol to attract as much traffic as it can from a particular area, and thus creating a metaphorical sinkhole [46]. Sinkhole attacks can also be used to act as a base for initiating other attacks [51], such as wormhole attack, eavesdropping, data alteration, etc. Since all the traffic typically flows through the compromised node, a wormhole attack would therefore become more effective and easier to achieve.

Stealth Attack In a *Stealth* attack [1, 30, 43], the attacker acts remotely, where even victim is not aware of the attack. Thus, this attack is hard to trace. It allows an attacker to partition a network, reduce its output, hijack and filter traffic to and from the victim nodes. The attack uses three measures [1]: Impersonation, overloading and lying. There are two kinds of stealth attacks, both based on entering false entry and removing valid entries in the routing table of honest node. The first type aims to reduce the output and isolate victim nodes of the network. The second type aims to hijack traffic to and from specific victim nodes in order to perform malicious actions like eavesdropping or packet filtering.

Location-Related Attacks In MANETs, the location information of nodes is very sensitive. Once the location information is obtained, it can be used by position-based routing protocols which are good alternatives of topology-based routing protocols [16]. Adversaries will try to mislead nodes in obtaining wrong location information about others. This shows a need for secure localization techniques [15,24,50,82] which will prevent both the internal and external attackers to misinterpret the location of nodes. On the other hand, these protocols are vulnerable to location information disclosure attack [17,20,26]. Location information should not be available to unauthorized nodes. If an adversary has all the location information, then it becomes easier to launch an attack against that network.

Table 1 summarizes and compares the attacks mentioned above. It indicates whether each attack requires collaboration among attackers, whether it can be detected or prevented by known IDS or cryptographic techniques, and what security properties it violates.

Table 1: Comparison of different attacks in MANET

Attacks	Attackers Collaboration	Detection and Prevention Techniques					Security Properties Violated		
		Anomaly	**Misuse**	**Specification**	**Cryptographic**	**Collaboration**	**Availability**	**Integrity**	**Confidentiality**
Rushing	Not Needed	No	No	No	Yes	Needed	Yes	Yes	Yes
Malicious Flooding	Not Needed	Yes	No	No	Yes	Not Needed	Yes	No	No
Packet Dropping	Not Needed	Yes	No	Yes	No	Needed	Yes	No	No
Sybil	Not Needed	No	No	No	Yes	Needed	Yes	Yes	Yes
Invisible Node	Not Needed	No	No	Yes	No	Needed	Yes	Yes	Yes
Wormhole	Needed	Yes	No	No	Yes	Needed	Yes	Yes	Yes
Backhole	Not Needed	Yes	No	Yes	Yes	Needed	Yes	Yes	Yes
Sinkhole	Not Needed	Yes	No	Yes	Yes	Needed	Yes	Yes	Yes
Stealth	Not Needed	No	No	No	Yes	Needed	Yes	Yes	Yes
Location disclosure	Not Needed	No	No	No	Yes	Not Needed	No	No	Yes

2.2 Intrusion Detection System in MANET

From the above section, we can see that the proactive approaches (i.e., cryptography and authentication) and other techniques (i.e., secure routing [85], [35], [36]) are not sufficient to guarantee security in MANETs. Hence an IDS is needed as a second line of defence.

2.2.1 Traditional IDS Systems

IDS is defined as a system which monitors and analyzes activities of a computer or network system to determine whether there are any abnormal activities that violate the security. In other words, IDS is based on a captured audit data and reasoning about evidence in the data to determine whether the system/network is under attack [59]. Thus, IDS has two components: Data collection and data analysis. Below we describe them briefly.

Data Collection Component

Data collection component is responsible for collecting different types of data, such as key-board input, command-based logs, application-based logs, network traffic, etc. According to the type of the audit data collected, we can classify the IDS into two categories [6].

Host based IDS Host based IDS depends on the operating system audit data to analyze the events resulting from programs or users on the host. It is able to detect abnormal actions such as repeated failed access attempts, changes to system files, etc. by monitoring real time system usage.

Network based IDS Network based IDS runs at the switches, gateways, or routers in order to analyze the captured packets that traverse through the network.

Data Analysis Component

Once the data is collected, data analysis component analyzes the data for possible intrusions. The techniques that the data analysis component uses to analyze the data can be categorized into three kinds [3, 59]: Signature based detection, anomaly based detection and specification based detection.

Anomaly-based IDS The Anomaly detection technique considers an activity as abnormal when it's deviate from the recognized normal usage. It must first be trained using normal data before it can be released in an operative detection mode. The main advantage of this model is that it can detect unknown attacks. On the other hand, its disadvantage is that it has high false positive alarm rate when normal user profiles, operating system, or network behavior vary widely from their normal behavior.

Signature-based (misuse detection) IDS Signature based detection compares the activities of a user with patterns of well-known intrusions for intruders attempting to compromise a system. The main advantage is that it can accurately detect instances of known intrusions. On the other hand, the disadvantage of misuse detection is that anomalous patterns are based on known attacks; therefore new attacks cannot be detected. Besides, it needs a large database of intrusion signatures which need to be updated periodically.

Specification-based IDS A set of constraints is defined by the system to describe the proper execution of a program or protocol. IDS monitors and captures data from the execution and considers it abnormal if it deviates from the defied variables.

Challenges of Traditional IDS

The IDS for wired network which has been developed over the years cannot be directly applied in the wireless network because of some major differences between the two networks. The unique characteristics of MANET such as mobility and open wireless medium make IDS deployment harder. Unlike traditional networks, there are no fixed chokepoints/bottlenecks where firewall or IDS can be deployed. Hence, more nodes need to run their IDS and cooperate with each other to ensure security. It is very inefficient in terms of resource consumption since mobile nodes are energy limited. Also these mobile nodes have computational limitation. Besides, both anomaly and signature based techniques are not suitable for MANET [45]. Anomaly detection needs a long training before deployment. Since the network changes very often in MANET, it is hard to model a normal behavior. On the other hand, signature based technique requires a database of attacks which need to be stored in every mobile host which is unsuitable for these computationally simple nodes. Since the current IDS techniques cannot be applied directly to MANETs, many intrusion detection architectures have been proposed to appeal the characteristics of MANETs. Next section presents different IDS architectures for MANET.

2.2.2 Architectures for IDS in MANET

In this section, we review several IDS architectures that have been proposed to appeal the characteristics of MANETs. In traditional networks, IDSs can be classified into two main categories based on audit data as either host-based or network-based. IDSs can also be divided into anomaly-based systems, signature-based systems, or specification-based systems. In MANETs, IDSs can further be divided into two types based on the presence of cooperation between nodes: Non-cooperative and cooperative IDS. Figure 1 summarizes different IDS architectures in MANET.

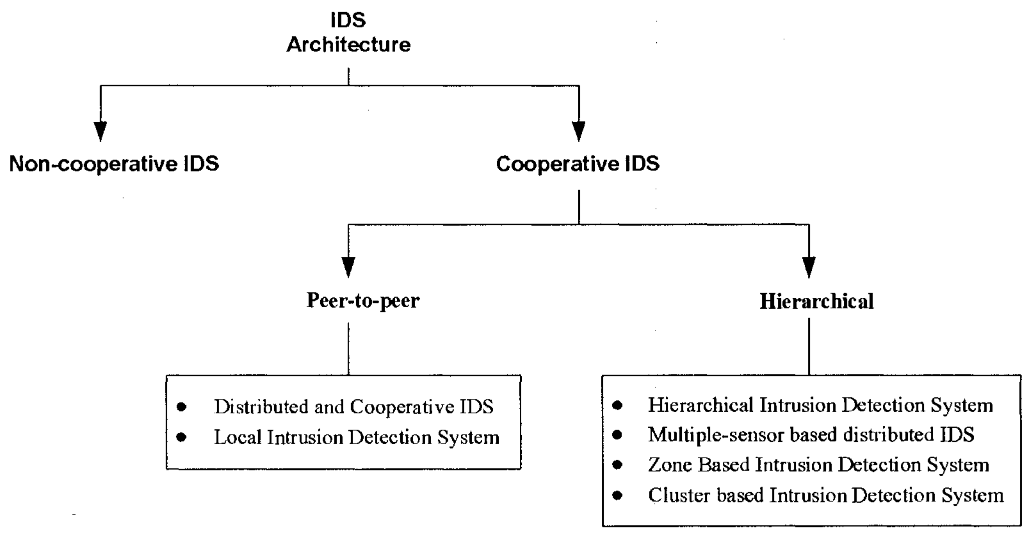

IDS Architecture
Non-cooperative IDS
Cooperative IDS
Peer-to-peer
Hierarchical
Distributed and Cooperative IDS
Local Intrusion Detection System
Hierarchical Intrusion Detection System
Multiple-sensor based distributed IDS
Zone Based Intrusion Detection System
Cluster based Intrusion Detection System

Figure 1: IDS architecture for MANET

Non-cooperative Intrusion Detection Systems

In this architectures, an IDS runs independently on a node to detect intrusions and there is no cooperation among nodes in the network. Each node makes decision based on its own collected information. This architecture is not popular in MANETs since nodes might not be able to detect intrusions based on only its information. In [73, 74], a non-cooperative signature-based intrusion detection architecture has been proposed. The authors argue that message exchange between the nodes increases the complexity and resource consumption in MANET. Misuse technique should be used since anomaly detection technique results high false positive for a network like MANET where the network characteristic varies a lot and lacks a normal behavior. Two polynomial complexity algorithms have been proposed for optimal node selection for running IDS module. The objective is to select a dominating set of nodes which will be able to capture all the traffics of the networks.

Cooperative Intrusion Detection Systems

A more popular category of IDS architecture is the cooperative IDS. In this architecture, nodes cooperate with each other to detect intrusions. This type of architecture is suitable for MANET since some attacks cannot be detected correctly without sharing the information. For example, an attack can be divided into multiple packets and each packet can be sent in different routes. Since data might not be comprehensive, all neighbor IDSs must cooperate in order to detect intrusion. There are two approaches to achieve cooperation: Peer-to-peer and hierarchical approach [7]. In peer-to-peer approach, each node runs independent IDS to detect intrusions and response consequently. Nodes share messages with others and thus cooperation is achieved. On the other hand, different nodes have different responsibilities in hierarchical approach. Nodes are divided into clusters where normal nodes are mostly responsible for local data collection and cluster heads perform the data aggregation, correlation, etc. Below we provide some of the examples of both approaches.

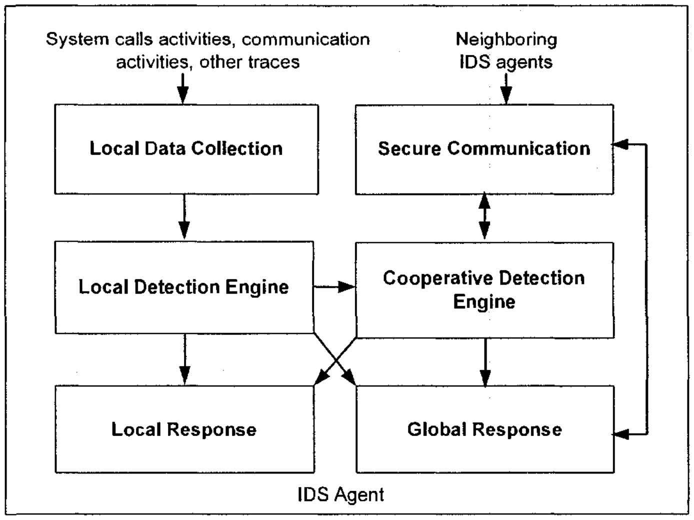


Figure 2: DCIDS architecture for MANET

Distributed and Cooperative Intrusion Detection System (DCIDS) This is an example of peer-to-peer approaches, which is proposed by Zhang and Lee [88]. In this model, each node independently runs an IDS agent for locally collecting, analyzing data and responding them accordingly. At the same time, nodes can cooperate with each other globally to detect an intrusion which has weak local evidence. The internal of an IDS agent can be structured into six different modules as shown in Figure 2. *Local Data Collection* collects the real-time audit data (i.e., system calls activities, communication activities, and other traces). *Local Detection Engine* analyzes the collected data to find the evidence of anomalies. *Local Response* alerts the local user when an anomaly is detected by local detection engine with strong evidence. *Global Response* initiates the response to an intrusion depending on the type of the intrusion and the type of network protocol and applications. *Cooperative Detection Engine* is used when the certainty of the evidence for detection of an anomaly is weak. The IDS agent requests the cooperation of neighboring IDS agents by this engine.

Finally, *Secure Communication* is needed to communicate with other agents securely.

Local Intrusion Detection System (LIDS) Local Intrusion Detection System (LIDS) is a peer-to-peer cooperative architecture implemented on each node to collect data and detect attacks independently [2]. It can also be extended for global intrusion detection by cooperating with other LIDS. In this architecture, two types of data are exchanged among LIDS: Security data (to obtain information form collaborating nodes) and intrusion alerts (to inform others of locally detected intrusion). The authors proposed to use SNMP (Simple Network Management Protocol) data located in MIBs (Management Information Source) as an audit data source for LIDS.

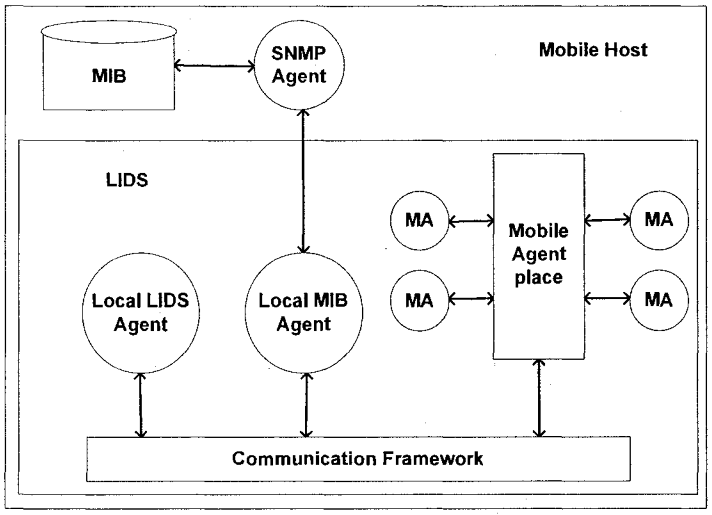


Figure 3: LIDS architecture for MANET

The LIDS architecture consists of several components that are shown in Figure 3. The *Local LIDS Agent* is responsible for local intrusion detection and response. In addition, it responses to intrusion alert sent from other agents to protect itself from such an intrusion.

Local MIB Agent is in charge of providing a means of collecting MIB variables for local LIDS agent or mobile agents. Local MIB agent will be an interface with SNMP agent if SNMP runs on the node. Otherwise, a SNMP based agent should be developed to permit update and retrieval of the MIB variables used by intrusion detection. *Communication Framework* is used to facilitate internal and external communication with LIDS. *Mobile Agent* is to collect and process data on other nodes by distributing their LID. Then, their results are sent to LIDS or another node for more investigations. *Mobile Agent Place* is responsible for a security control of mobile agents.

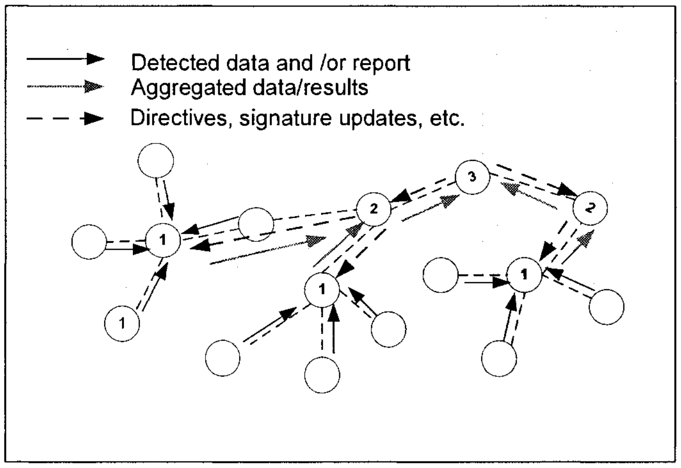

Detected data and /or report
Aggregated data/results
Directives, signature updates, etc.
1
2
3

Figure 4: Hierarchical IDS architecture for MANET

Hierarchical Intrusion Detection System Since the nature of MANETs is distributed and requires cooperation of other nodes, hierarchical IDS architecture [25] has been proposed for a hierarchical network divided into clusters as shown in Figure 4. In this architecture, each IDS running on every node is locally responsible to monitor, determine intrusions and decide on detected intrusions. In addition, a cluster-head like other nodes is responsible for its node as well as globally for its cluster to collect data, detect intrusions and globally response to network intrusions. MANET is structured in more than two levels. The first level cluster-heads are nodes labeled “1” which are called leaf nodes while the second level cluster-heads are nodes labeled “2” and so on. To form the hierarchy, leader election algorithm is used in each cluster to elect a cluster-head. The authors did not elaborate the election procedure but suggested some criteria for selecting cluster-heads. The selection should be done based on connectivity, proximity, resistance to compromise, processing power, storage capacity, energy lever, etc.

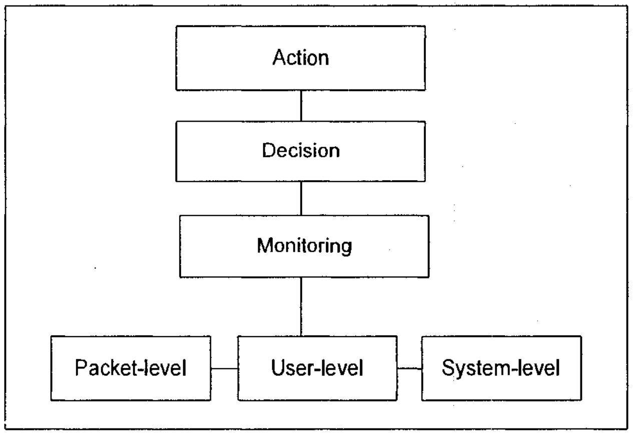


Figure 5: Multiple-sensor based IDS architecture for MANET

Multiple-sensor based distributed Intrusion Detection System Kachirski and Guha proposed an intrusion detection model using multiple sensors [45]. It follows the hierarchical approach. The task of intrusion detection has been divided into three different modules and represented by a mobile agent, which is shown in Figure 5. Each node is responsible to host different IDS agent to balance the workload of the system. *Monitoring Agent* is for host monitoring and network monitoring. Every node of network is monitored by the host monitoring agent. The agent monitors system and application level activities. On the other hand, certain nodes have network monitoring agent. This is done to preserve the total computational and battery power of mobile hosts. Network monitoring agents are responsible for collecting and analyzing network level packets. *Decision Agent* works at host level for local intrusions. Since certain nodes collect and analyze network level information, only these nodes have network level decision agent to make collective decision about network level intrusions. *Action Agent* is hosted by every node. Based on the monitoring, each node is able to take appropriate response of a given intrusion. The network is divided into multiple clusters where there exists a cluster head for each cluster. The cluster-heads host network level monitoring and decision agents on behalf of the cluster. In this architecture, the cluster head is selected based on the connectivity of each node. A detailed algorithm for electing a cluster-head is given in [45].

Zone Based Intrusion Detection System (ZBIDS) ZBIDS follows the hierarchical approach [75]. The whole network is divided into non overlapping zones, where nodes are of two types: Interzone or intrazone nodes. Nodes that have physical connections to the nodes of other zones are known as interzone nodes, where as nodes without any connection are intrazone nodes. The formation and the maintenance of the zones need geographic partitioning or clustering algorithms. Global Positioning System (GPS) or other methods are needed to find the physical location of each node to determine its zone identity by mapping its physical location to the predefined zone map. The interzone nodes are responsible for global aggregation and correlation while intrazone nodes are responsible for local data analysis.

Cluster based Intrusion Detection System This architecture is similar to Kachirski and Guha's architecture [45], proposed by Hang and Lee [39]. The intrusion detection system is anomaly based and nodes compute many different features of the network. Since it takes significant computational cost to obtain traffic-related features, the network is divided into clusters. The cluster-heads help to compute the traffic-related features and the citizen nodes compute and transmit the non-traffic related features to the cluster-heads. Thus the overall cluster-wise feature collection cost is reduced since a lot of features are related to traffic-related. The authors propose a random election procedure for electing cluster-heads. This election takes place after every fixed interval to elect a new leader. In addition, a cluster recovery protocol also has been proposed to adjust lost nodes (nodes without leader) due to mobility.

Besides, there are some other IDS models which are based on cooperation enforcement [13,14,57]. These techniques mainly focus on the selfish behavior of nodes regarding forwarding others packets. Selfish nodes can degrade the performance of the network by dropping others packet. To motivate the selfish nodes in routing, CONFIDANT [13] 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 [57] 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.

2.3 Leader Election in IDS

In this section, we present the application of leader election in IDS along with other possible applications in MANET. We then describe the current techniques and their limitations with an illustrative example.

2.3.1 Applications of Leader Election

Leader election is a significant issue of MANET and it is the key concept of Cooperative IDS. The main difference between the two approaches of cooperative IDS is the organization of the nodes. In peep-to-peer approach, each node needs to communicate with other nodes to achieve the cooperation. This results in higher communication overhead as the number of nodes increases. To reduce this overhead, nodes can be organized into hierarchy by forming clusters and electing a leader for that cluster to distribute the responsibilities.

This is the key concept of hierarchy approaches [25, 39, 45] of cooperative IDS. Leader election is a significant issue of MANET and has been addressed in different research work [8, 48, 76, 78].

Like Intrusion Detection System (IDS), leader election is needed for routing [9] and key distribution [10, 27, 68] in MANET. In key management, a central key distributor is needed to update the keys of the nodes. In routing, 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. Hence, the performance of the network significantly depends on the efficient clustering and leader election algorithm.

2.3.2 Problems of Current Techniques

Many clustering and leader election algorithms have been proposed. These algorithms can be classified into two categories [76]: Cluster-first or leader-first. In cluster-first approach [48], a cluster is formed then nodes belonging to that cluster elect a leader node. In the leader-first approach [8], a set of leader nodes is elected first then the other nodes are assigned to different leader nodes. Whether it is cluster-first or leader-first, the election of leader node is done in three ways: randomly, based on connectivity (node's degree) and based on a node's weight. Here, the weight normally refers to the remaining energy of a node [79]. In random election [39] model, a node is elected randomly as a leader by a group/cluster of Nodes. If there are N nodes in a cluster, then the probability of any particular node to be leader is $1/N$. Random model does not consider the remaining resources of nodes or the presence of selfish nodes for electing leader node. In connectivity model [45], nodes with highest degree (connection to other nodes) are elected as leader nodes for the network. Again, the solution ignores both the difference in remaining resources and the selfishness issue. We motivate further discussions through a concrete example.

Figure 6 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 are 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 [39], nodes N_5 and N_9 will have equal probability, compared to others, in being elected as leaders. Whereas nodes N_5 and N_9 will definitely be elected under the connectivity index-based approach due to their connectivity indices [45]. 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 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 than others.

We pointed out the problems of random model and connectivity model. We believe that weight based leader election should be the proper method for election. But unfortunately, the information regarding the remaining energy is private to a node and thus not verifiable. Previous methods assume that each node is associated with a weight [9] or there exist a

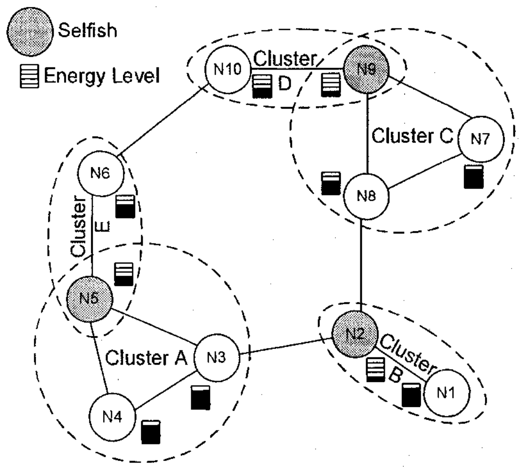


Figure 6: An example scenario of leader election in MANET

trusted authority to certify each node's metric (weight) which is used to elect a leader. We consider these assumptions as quite strong for MANET. Since nodes might behave selfishly, they might lie about their resource level to avoid being leader if there is no other mechanism to motivate them. Thus, we address the selfish problem and propose a solution which is able to balance the resources of nodes considering the selfish behavior of nodes.

2.4 Mechanism Design Background

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, rational players select to deliver untruthful information about their preferences, if that leads to individually better outcome. The main goal of using mechanism design [23] is to address this problem by designing incentives for players (nodes) to provide truthful information about their preferences over different outcomes. In the rest of the chapter, we discuss the important concepts of game theory and mechanism design which are important to better understand our mechanism. A more general overview of mechanism design can be found in [23,42,55,64].

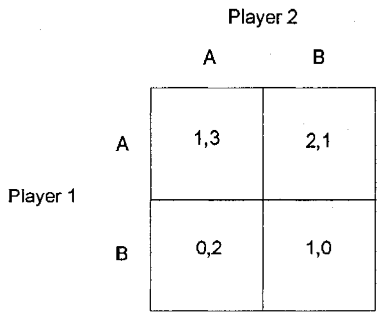
Mechanism design is a sub-field of microeconomics and game theory [55]. Mechanism design uses game theory [60] tools to achieve the desired goals. The main difference between game theory and mechanism design is that the former can be used to study what could happen when independent players act selfishly. 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 applications of mechanism design are diverse and it has been used successfully in electronic market design, resource allocation problems, task scheduling problems, etc. In the following, we present the basic definitions and concepts of game theory that are used in mechanism design. Then, we introduce mechanism design which is the fundamental concept of our proposed model.

2.4.1 Game Theory

Games are defined by a number of players that interact by forming a coalition or possibly threaten each other by taking actions under uncertainty. Each player finally receive some benefits, payoffs, or loss some rewards or payoffs. Thus, a game is an interaction model, where each player has its own strategies and possible payoffs. Game theory is the tool that studies the relationship and interaction, which are established and broken in the course of cooperation and competition between the nodes.

Basic Definitions

A normal game consists of a finite set of players $P = \{1, 2, \ldots, n\}$, a strategy set s_1, for each player and a set of outcomes, O. Below, we explain different elements of a game through an example where two players have two strategies each. Figure 7 shows the payoffs for each strategy profile of the game, where the columns represent the strategies of player 2 and the rows represent the strategies of player 1. The intersection between the row and the column represents the payoff of player 1 and 2. Thus the utility of player 1 and player 2 are 1 and 0 respectively when both the player choose strategy B.



		Player 2	
		A	B
Player 1	A	1,3	2,1
	B	0,2	1,0

Figure 7: Example of a two player game

Strategy Each player can select his strategy from a strategy set s_i. For example, the strategy set for each player is $S_1 = S_2 = \{A, B\}$. Strategies can be two types: pure and mixed. Pure strategies are deterministic while a mixed strategy is the probability distribution over the set of pure strategies. A strategy profile $s = \{s_1, s_2, \ldots, s_n\}$ is the vector of strategies. It is the set of all the strategies chosen by the players, where as $s_{-i} = \{s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n\}$ denotes the strategies of all the players except player

i. Strategy profile is the outcome, $o(s) \in O$, of the game. For example, the possible outcomes of the game are (A, A), (A, B), (B, A), and (B, B). Each player chooses its strategy in such a way that its preferred outcome occurs and the preference over the outcome of the game is expressed by the utility function.

Utility Function Each player has preferences over the outcomes of the game. Preferences over outcomes are represented through a utility function, u_i. Utility function of a player i can be considered as a transformation of the outcome to a real number. It is expressed mathematically as:

$$u_i : O \rightarrow \Re$$

A player prefers outcome, o_1 over outcome, o_2 if $u_i(o_1) > u_i(o_2)$. A rational player always wants to maximize its utility. Thus, it chooses a strategy which will increase its expected utility given the preferences of the outcomes, the structure of the game and the belief of others strategies.

Solution Concepts

Game theory uses different techniques to determine the outcome of the game. These outcomes are the stable or the equilibrium points of the game. The most well known equilibrium concept is known as Nash equilibrium. It states that each player plays its best strategy to maximize the utility, given the strategies of the other players.

Nash Equilibrium A strategy profile $s^* = \{s_1^*, s_2^*, \ldots, s_n^*\}$ is Nash equilibrium if this strategy profile maximizes the utility of every player, i. Formally

$$\forall i \; u_i(o(s_i^*, s_{-i}^*)) \geq u_i(o(\acute{s}_i, s_{-i}^*)), \forall \acute{s}_i$$

Nash equilibrium is the point where no player can take advantage of the other player's strategy to improve his own position. Given the above example, if player 2 predicts that player 1 will choose strategy A then choosing a strategy other than strategy A will lead to a loss for player 2. Although Nash equilibrium is a fundamental concept of game theory but the assumptions of this technique is too strong for some scenarios. Nash equilibrium assumes that every player has the perfect information about the other players and all the players must reach the same Nash equilibrium point.

Dominant Strategy Strategy s_i^* is the dominant strategy for player i, if it maximizes the expected utility against all possible strategies of the other players.

$$u_i(s_i^*, s_{-i}) \geq u_i(\acute{s}_i, s_{-i}), \forall s_{-i}, \forall \acute{s}_i \neq s_i^*$$

A strategy is dominant if it is the best strategy for the player against any strategies of other players. In dominant strategy equilibrium, every player has a dominant strategy. It is a very robust solution concept since it does not require a player to have information of other players. In the case of mechanism design, a social choice function implementation is much more desirable by dominant strategy than Nash implementation. It is because Nash implementation requires each agent to be well informed about others which is not practical in the case of mechanism design. An ideal mechanism provides each player with a dominant strategy and thus implements a solution for a multi-agent optimization problem.

2.4.2 Mechanism Design

A standard mechanism design model is defined by a set of N players. Each player holds a private information θ_i, which is called the type of the player i. The type θ_i is drawn from each player's available type set Θ_i and it describes how each player values all possible outcomes. Each player i has a utility $u_i(\theta_i, o)$ for given outcome $o \in O$ and type θ_i, where O is the set of possible outcomes. The goal of mechanism design is to implement a system wide solution which is defined as social choice function (SCF).

Social Choice Function A social choice function is a function $f : \Theta_1 \times \ldots \times \Theta_n \rightarrow O$, which gives an output $f(\theta) \in O$, given the types of the players, $\theta = (\theta_1, \ldots, \theta_n)$.

The mechanism design problem is to define some rules that are used to determine the solution of the social choice function despite players own self interest.

Mechanism A mechanism, $M = (S_1, \ldots, S_n, g(.))$ defines a set of strategies for each player and an outcome rule, $g : S_1 \times \ldots \times S_n \rightarrow O$ such that, $g(s)$ is the outcome implemented by the mechanism for the strategy profile, $s = (s_1, \ldots, s_n)$.

In other words, a mechanism defines strategies for the nodes and rules to compute the outcome based on players strategies. A mechanism, M implements a social choice function, $f(.)$ if the outcome $g(.)$ of the mechanism is based on *equilibrium players' strategies* and also a solution for the social choice function for all $(\theta_1, \ldots, \theta_n) \in \Theta_1 \times \ldots \times \Theta_n$.

Mechanism Implementation A mechanism, $M = (S_1, \ldots, S_n, g(.))$ implements a social choice function, $f(.)$ if $g((S_1^*(\theta_1), \ldots, S_n^*(\theta_n)) = f(\theta)$, where the strategy profile, $s^* = (s_1^*, \ldots, s_n^*)$ is the equilibrium point of the mechanism.

Here, the equilibrium strategy profile can be based on any equilibrium concept, such as Nash, Bayesian-Nash, dominant, etc. Generally, dominant strategy implementation is preferred since it has less assumption than other concepts regarding players.

The Revelation Principle

It is hard to identify which social choice functions are implementable since in theory, we need to consider all possible mechanisms. The *revelation principle* states that we can restrict ourselves to a small class of mechanism which is known as *direct revelation mechanism.* This is the simplest type of mechanism where each player is required to reveal its type as its strategy.

Direct Revelation Mechanism In direct revelation mechanism, the strategy of a player i is $S_i = \Theta_i$ and $g(\theta) = f(\theta)$ for all $\theta \epsilon \Theta_1 \times \ldots \times \Theta_n$.

In other words, the strategy of a player is to report its type to the mechanism. The revelation principle also states that we can further restrict to such direct revelation mechanism where truth telling is the optimal strategy of the players. The social choice functions implemented by such mechanisms are known *incentive compatible.*

Incentive Compatible An incentive compatible direct revelation mechanism implements a social choice function, $f(.)$, if the mechanism, $M = (\Theta_1, \dots, \Theta_n, g(.))$ has an equilibrium $(s_1^*, \dots, s_n^*)$, where $s_i(\theta_i) = \theta_i$ for all $\theta_i \in \Theta_i$ and for all player i.

In other words, the incentive compatible direct revelation mechanism has an equilibrium point where each player reports its type truthfully. If the concept of equilibrium is dominant strategy, then this mechanism is said to be strategy proof. Thus, the *revelation principle* restricts our attention to only incentive compatible direct mechanism to determine which social choice function are possible to implement and which are not.

Quasi-Linear Preferences

One common assumption in mechanism design is that players have quasi-linear utility function. A general form of a quasi-linear function for a player i with type θ_i is:

$$u_i(o, \theta_i) = v_i(x, \theta_i) - p_i$$

where x is an element of a discrete choice set K defined by the outcome o and p_i is the payment of the player. The preference of each player from the output is calculated by a valuation function, $v_i(x, \theta_i)$. This is a quantification in terms of a real number to evaluate the value of x. Thus the utility of a node is calculated as the combination of output measured by valuation function and the payment defined by the mechanism. As an example, we can consider an auction of an item. The outcome of the auction is the determination of a winner and a payment which the winner has to pay for the item. If the value of item to the winner is 20 then it has a positive utility as long as the payment is less then 20. 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? The VCG mechanism answers this question by giving the nodes a fixed payment independent of the nodes' valuation.

Vickrey-Clarke-Groves (VCG) Mechanisms

VCG mechanism is a well known mechanism which was proposed by Vickrey [80], Clarke [21] and Groves [31]. VCG is an efficient and strategy-proof direct revelation mechanism. It is used to solve problems in which players have quasi-linear preferences. In VCG mechanism, each player gives its type, $\theta_i = s_i(\theta_i)$ as an input to the mechanism which may not be the true type. Based on the input vector, $\theta = (\theta_i, \ldots, \theta_n)$ the mechanism computes an output in terms of a choice rule, k and a payment rule, p_i .The choice rule of VCG is:

$$k^*(\theta) = \arg\max \sum_{x \in K} v_i(x, \theta_i)$$

The selection of k^* is such that it maximizes the total valuation over all the players. Thus, VCG mechanism produces efficient allocation. The payment rule of VCG is:

$$p_i(\theta) = h_i(\theta_{-i}) - \sum_{j \neq i} v_j(k^*, \theta_j)$$

where $h_i : \theta_{-i} \rightarrow \Re$ is an arbitrary function which does not depend on the type of player i. Therefore, the payment of a player is not influenced by its reported type.

Mechanism Design Applications

Mechanism design has been extensively used in microeconomics for modeling solutions for various economical problems such as auctions. It has been used in computer science [63] for solving least cost path and task scheduling problems using algorithmic mechanism design. Distributed mechanism design based on VCG is first introduced in [28] to compute the lowest cost routes for all source-destination pairs and payments for transit nodes on all the routes. An extension of this work is given in [70] where the authors considered consequences that may appear after the computation phase is finished. Currently in MANET, mechanism design is mainly used for routing purposes. 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 [5]. In [18], 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. Leader-election in IDS is significantly different from the above problems. We address selfishness in IDS leader election, which is a real problem that has not been addressed by previous approaches.

2.5 Summary

In this chapter, we discussed different types of attacks which are unique to MANET and mentioned that proactive approaches are not enough to prevent these attacks. Intrusion detection system is needed as a second line of defense to defend the network. We identified the challenges of IDS and elaborated the new techniques of IDS architecture for MANET. We illustrated that leader election is a key concept for cooperative IDS and for other applications of MANET. We discussed the problems of the current solutions of leader election and motivated it with an example. To solve the problems of the current solutions, we use mechanism design in our proposed method. Finally, we reviewed the basics of game theory and mechanism design which are needed for the rest of the thesis.

Chapter 3

Leader Election Mechanisms

In this chapter, we define formally the problem statement. Then, we describe our leader election mechanism where the cost of analysis function, reputation model and payment designs are given. Finally, we analyze our mechanisms against selfish and malicious nodes.

3.1 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 total 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):

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

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 this chapter. Second, we need to design an election algorithm that can provably minimize the above SCF while not incurring too much of performance overhead. This is addressed in chapter 5.

3.2 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 it reveals the true cost of analysis. In Section 3.3 a detailed analysis is given. We assume that each player i has a utility function [55]:

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

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.

3.2.1 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 follows:

$$CL = \begin{cases} cl_1 & if\ PF < \rho_1 \\ cl_i & if\ \rho_{i-1} \le PF < \rho_i;\ i \in [2, l-1] \\ cl_l & if\ PF \ge \rho_{l-1} \end{cases} \tag{3}$$

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:

```
Cost-of-Analysis Function
/* Nodes execute this function to calculate their cost*/
1. if ($E_i < E_{ids}$) then
2.     $c_i = \infty$
3. else
4.     $c_i = \frac{PS_i}{PF_i} = \frac{\frac{R_i}{\sum_{i=1}^{N} R_i} \times nT_i}{E_i}$
5. end if
```

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 2 shows the PS for 20 nodes divided equally in 4 energy classes where nodes in class 4 have the most resources. Table 2 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

Table 2: 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%

able to balance the energy of the nodes and gives a fair budget to all nodes.

3.2.2 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 8 shows the abstract model of our reputation system where each node has the following components:

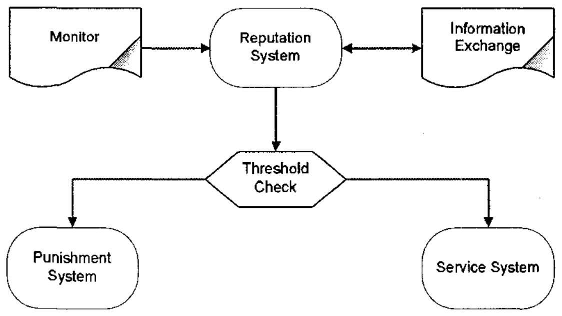


Figure 8: Reputation system model

Monitor: 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 [65].

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 [65] 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.

3.2.3 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:*

$$P_k = \sum_{i \in N} vt_k(C, i) B \rho_k, \text{ where} \tag{4}$$

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

$$[\sum_{j\in N} c_j \sum_{i\in N} vt_j(C|c_k=\infty, i) - \sum_{j\in N} c_j \sum_{i\in N} vt_j(C,i)] \tag{5}$$

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

$$T_k(C) = c_k \sum_{i\in N} vt_k(C,i)B \tag{6}$$

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

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

where the objective function is the sum of all players' valuations [63]. Here valuation refers to the total cost incurred by a node. According to [44], the strategy-proof payment for minimizing a function should have the following generalized form.

$$P_k = T_k(C) - S(C) + h_k(c^{-k}) \tag{8}$$

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,

$$h_k(c^{-k}) = \sum_{j\in N} c_j \sum_{i\in N} vt_j(C|c_k=\infty, i)B \tag{9}$$

This means,

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

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

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

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

$$= \sum_{i \in N} vt_k(C,i)B\rho_k \tag{12}$$

where,

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

$$[\sum_{j \in N} c_j \sum_{i \in N} vt_j(C|c_k = \infty, i) - \sum_{j \in N} c_j \sum_{i \in N} vt_j(C,i)] \tag{13}$$

This concludes the proof since the designed payment is in the generalized form of strategy-proof payment. $\square$

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. A detailed example is given in section 4.2.

3.2.4 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. Any cluster formation algorithm [48] can be used to divide the network into clusters. Each cluster then independently elects a leader node among them to handle the monitoring process based on the cost of analysis. Our objective is to find the most cost-efficient node that handles the detection process for the particular cluster. Hence our social choice function (Equation 1) can be simplified for a particular cluster in CDLE as follows:

$$SCF(for\ CDLE) = \min \sum_{k \in n} v_k(\theta_k, o(\theta_k, \theta_{-k})) \tag{14}$$

To achieve the desired goal, payments are computed using the VCG mechanism where truth-telling is proved to be dominant. Like CILE, our mechanism 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:*

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

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

Proof: According to the standard notation in mechanism design [63], 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 price excluding n_k. □

3.3 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.

3.3.1 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.

3.3.2 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 [65] 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.

3.4 Summary

In this chapter, we presented our leader election mechanism for truthfully electing the leader nodes. We formulated our model using the standard mechanism design notations. To achieve the design goal, we designed the cost of analysis function which is the private information of the nodes. To reveal the private information truthfully, we provided payment in the form of reputation. We described how this payment can be used to motivate nodes and to build a trusted environment. We also designed the payments for the two models which are strategy proof. Finally, we analyze our mechanism against selfish and malicious nodes. In the following chapter, we present the leader election algorithms to implement our mechanism.

Chapter 4

Leader Election Algorithms

To run the election mechanism given in Chapter 4, 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.

4.1 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 [66] 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 [32].

4.2 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.

4.2.1 New Election

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 4.4.2.

```
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($neighbors(k)$=∅) 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 and terminates the algorithm.

```
Algorithm 2 (Executed by every node)
/* Each node votes for one node among the neighbors */
1. if ($\forall\ n \in neighbor(k) : c_i < c_n$)  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.

On expiration of T_3, the elected node i calculates its payment using equation 4 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

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.
```

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: To illustrate the election scheme, Figure 9 shows a MANET with ten nodes. 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 3.

Table 3: 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

To elect a new leader in the 10^{th} round, every node sends *Hello* and then a *Begin-Election* message according to Algorithm 1. Nodes reveal their cost of analysis to the

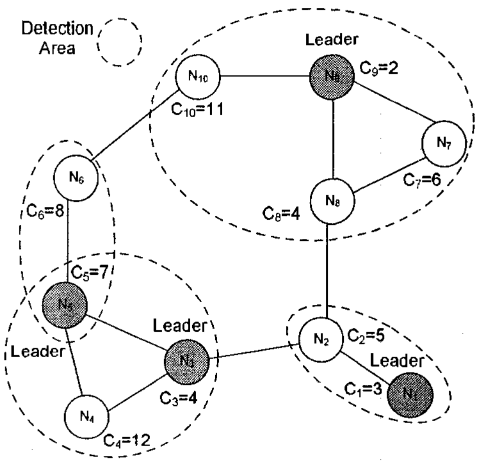


Figure 9: A MANET after leader election

mechanism based on their type (*Selfish* or *Normal*). The corresponding cost of analysis is given in the second row of Table 3. Then, node 7, 8, 9 and 10 vote for node 9 to be the leader as it has the least cost of analysis. Similarly, node 6 votes for node 5; node 3, 4 and 5 vote for node 3; node 1 and 2 vote for node 1. After getting the vote, leader node 1, 3, 5 and 9 will calculate their payment using equation 4. For node 9, the payment per packet is $\rho_9 = 2 + \frac{1}{4}(8 \times 1 + 4 \times 3 - 2 \times 4) = 5$ 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 total payment of node 9 is $P_9 = \sum v_9 B \rho_9 = 4\times5\times5 = 100$, where B= 5 packets/sec is the sampling budget. After election, leader N_9 distributes the total IDS sampling budget over the protected nodes N_7, N_8, N_9 and N_{10}, according to their reputation, as follows: $S = \{S_7 = \frac{90\times20}{470}, S_8 = \frac{160\times20}{470}, S_9 = \frac{110\times20}{470}, S_{10} = \frac{110\times20}{470}\}$. Similarly, the payment for elected leaders N_1, N_3 and N_5 will be 45, 95 and 40 respectively. Finally, the leader nodes will send *Acknowledge* message to all neighbors and run their IDS. Upon receiving the *Acknowledge*, all the neighboring nodes increase the reputation of the elected leaders, as shown in the third row of Table 3.

4.2.2 Adding New Nodes

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 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).

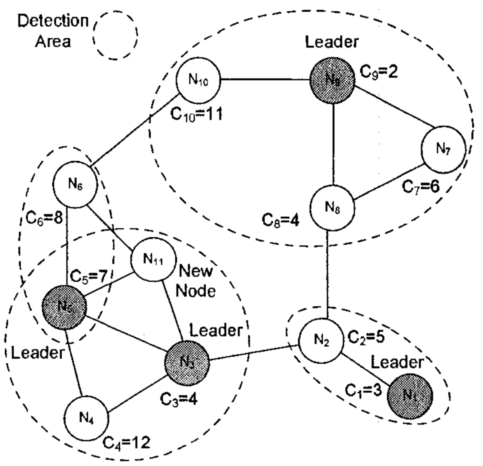


Figure 10: A MANET after adding a new node

Detailed analysis is presented in Section 4.4. 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.

Example: Let us consider a new node that will be added to the network in Figure 9. The resulting network after adding the new node is shown in Figure 10. 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. Node 11 sends *Join* to leader node 3 as it has the least cost. Finally, leader node 3 adds node 11 in its serving list.

4.2.3 Removing Nodes

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.

```
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 ($n \in service(k)$) then
8.       $update_{service()}$;
9.    end if;
10. end if;
```

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 updates its *service-table.*

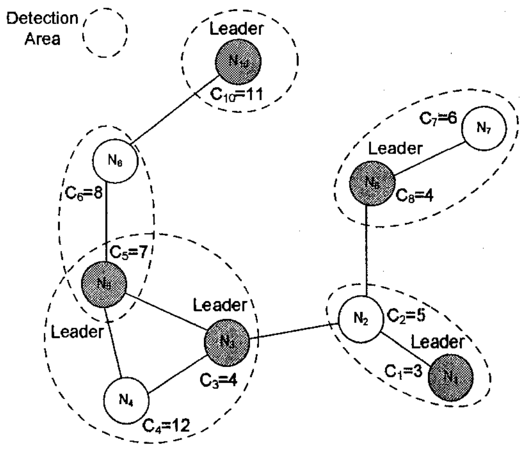


Figure 11: A MANET after adjustment

Example: Here, we consider the removal either of an ordinary node or a leader node. Considering the network in Figure 9, let us assume that node 7 has left the network or died. Immediately, node 8 and 9 will be aware of the failure. On receiving the *Dead(7)* message, node 8 and 9 check whether node 7 is their leader or it's being served by them. As node 7 is an ordinary node, node 8 does nothing. In case of node 9, it updates its serving list as in Algorithm 4. Assume now that the links of node 9 have been broken as shown in Figure 11. Then the neighboring nodes 7, 8 and 10 will discover that node 9 is their leader. Immediately, they will go for a new election, and node 8 will become the new leader. 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, because node 6 is an ordinary node and it is served by node 5. It has to wait for the expiration of T_{ELECT} for the new election.

4.3 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.

4.3.1 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 [32] to verify a signature in $0.43s$. Since leader election will take place after a certain interval, this computational overhead is tolerable.

4.3.2 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 $3|N_i| + 1 \approx O(N_i)$, where $|N_i|$ is the number of nodes in the network.

4.3.3 Storage Overhead

According to the algorithm, each node maintains a *reputation-table*, *neighbors* list and two variables: *Leadernode* and *leader* (see section 4.2). 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 [48].

4.4 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.

4.4.1 Algorithmic Correctness

Here, we prove that our algorithm achieves our objectives mentioned in section 4.1.

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. This can be proved from Algorithm 2. 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 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 new election round after T_{ELECT} to participate in the election process. □

4.4.2 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. In the following, we discuss some of the security properties of our algorithm.

The algorithm provides basic security requirements. Since we assume the presence of TESLA and PKI protocols, all the messages are signed by the source node and verified by others. This provides the integrity of every message. Unauthorized nodes can not do any modification of the messages. The recipient nodes can verify the signature of the sender node. Since the private key belongs only to the sender node, thus source authentication is also achieved by protecting the message integrity. Besides, each message includes the time. Since loose synchronized clocks are available between the nodes, the recipient nodes can verify wheatear the message is reply or not. Thus our algorithm is also safeguarded against reply attack.

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.

4.5 Summary

In this chapter, we designed our election algorithm to implement our mechanism. The algorithm provides the details about how the nodes compute their payments and how they elect a set of leader node from the whole network. Besides, we also extended the solution to reconfigure the network in the case of addition or removal of nodes from the network. We calculated different overheads of our algorithm and showed that our algorithm is correct and meet our objectives. Finally we discussed different security properties of the algorithm. In the next chapter, we validate the performance of our algorithm through simulation.

Chapter 5

Simulation Results

In this section, we evaluate the performance of our model with respect to random and connectivity models. We simulate the schemes using Network Simulator 2 (NS2) [56] and MATLAB.

5.1 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. Each point in the graph is the average result of 100 simulation run.

5.2 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 4: 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 4 summarizes our simulation parameters.

5.3 Experiments

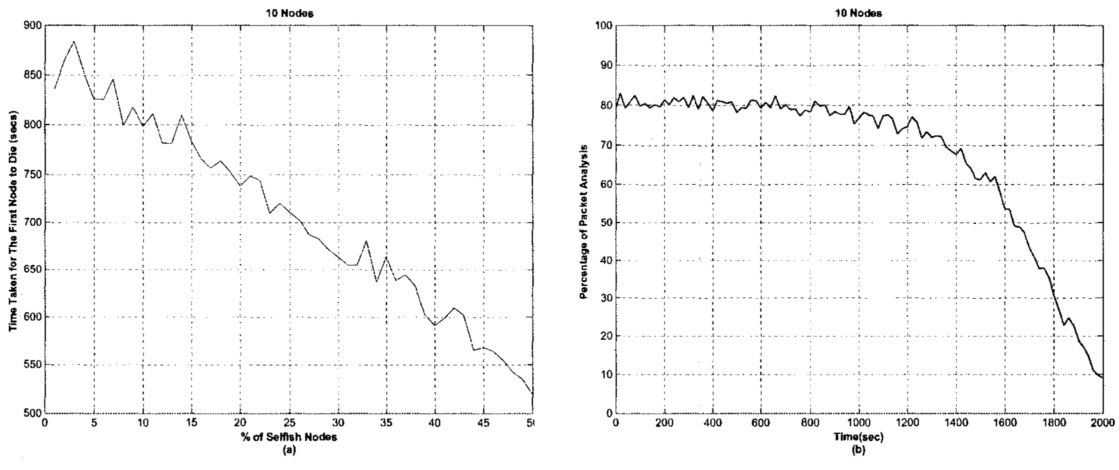


Figure 12: Effect of selfish nodes on the other nodes

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 12.(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 12.(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 12.(b). This is a severe security concern since fewer packets are analyzed.

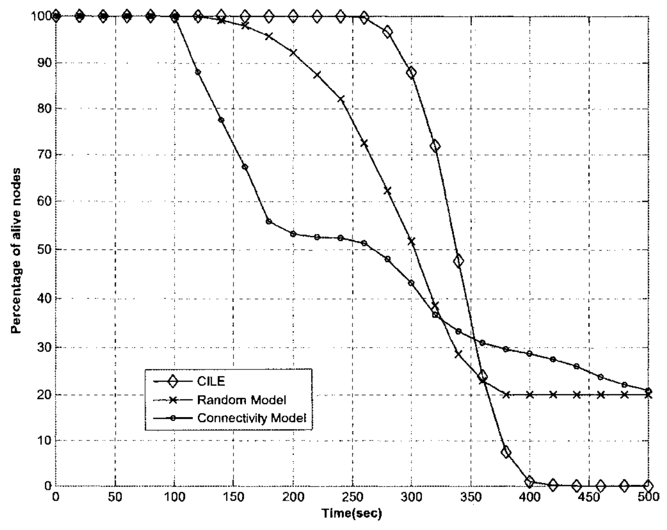


Figure 13: Percentage of alive nodes

In Figure 13, 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 9 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. Hence, all the nodes will live long and die at the same time which is clearly shown in Figure 13.

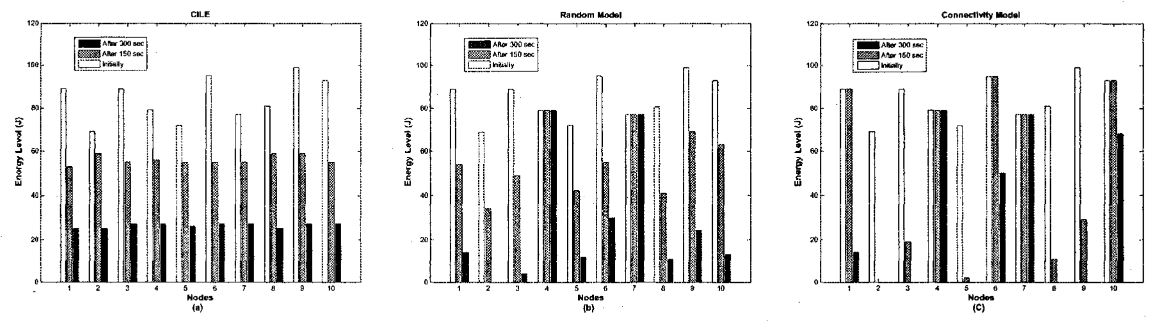


Figure 14: Energy level of the nodes

Figure 14.(a) indicates that our model is able to balance the resource consumption among all nodes. On the other hand, the random (Figure 14.(b)) and connectivity (Figure 14.(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 4. 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 15 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 16.(a) shows the percentage of the leader nodes. The percentage of leaders for our model is less as compared to those of the connectivity

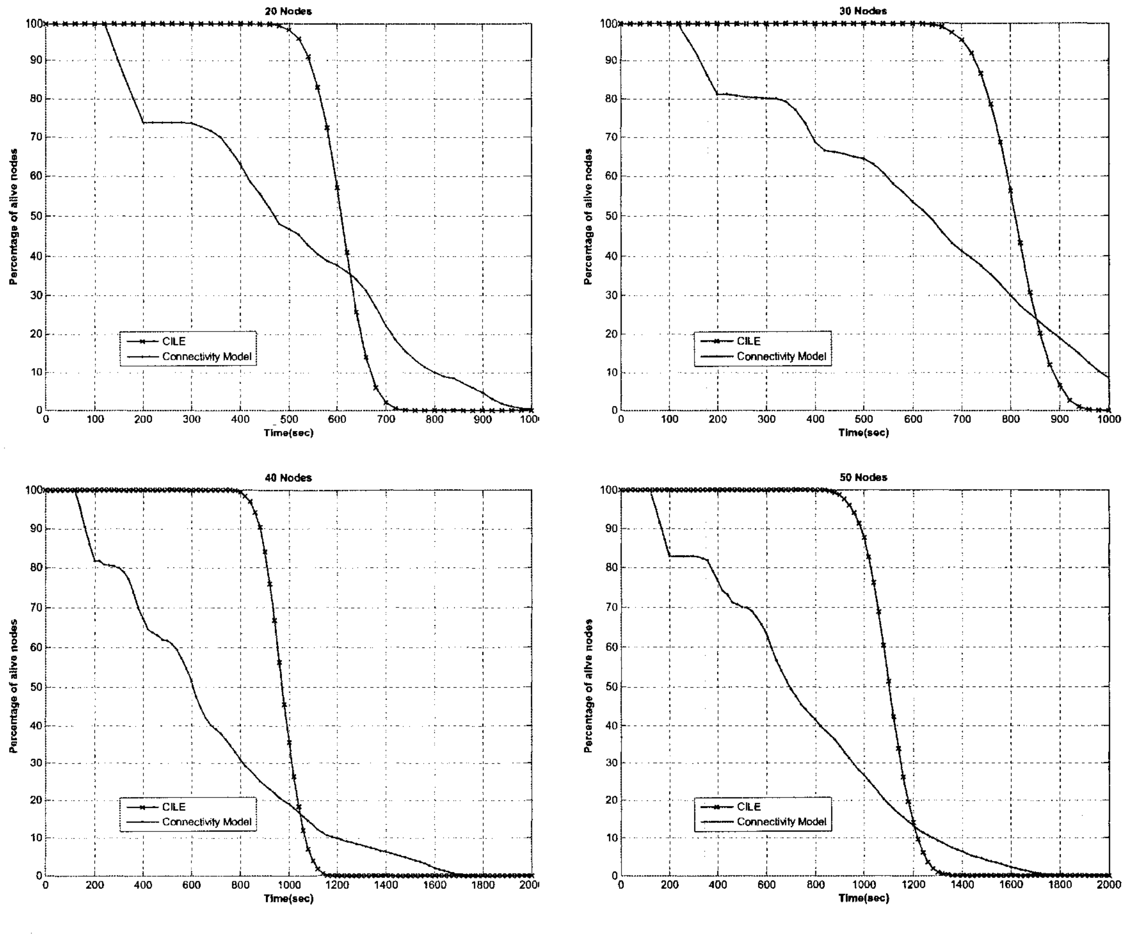


Figure 15: Percentage of alive nodes in dynamic network

model that saves the energy of nodes. Figure 16.(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 16.(c) 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

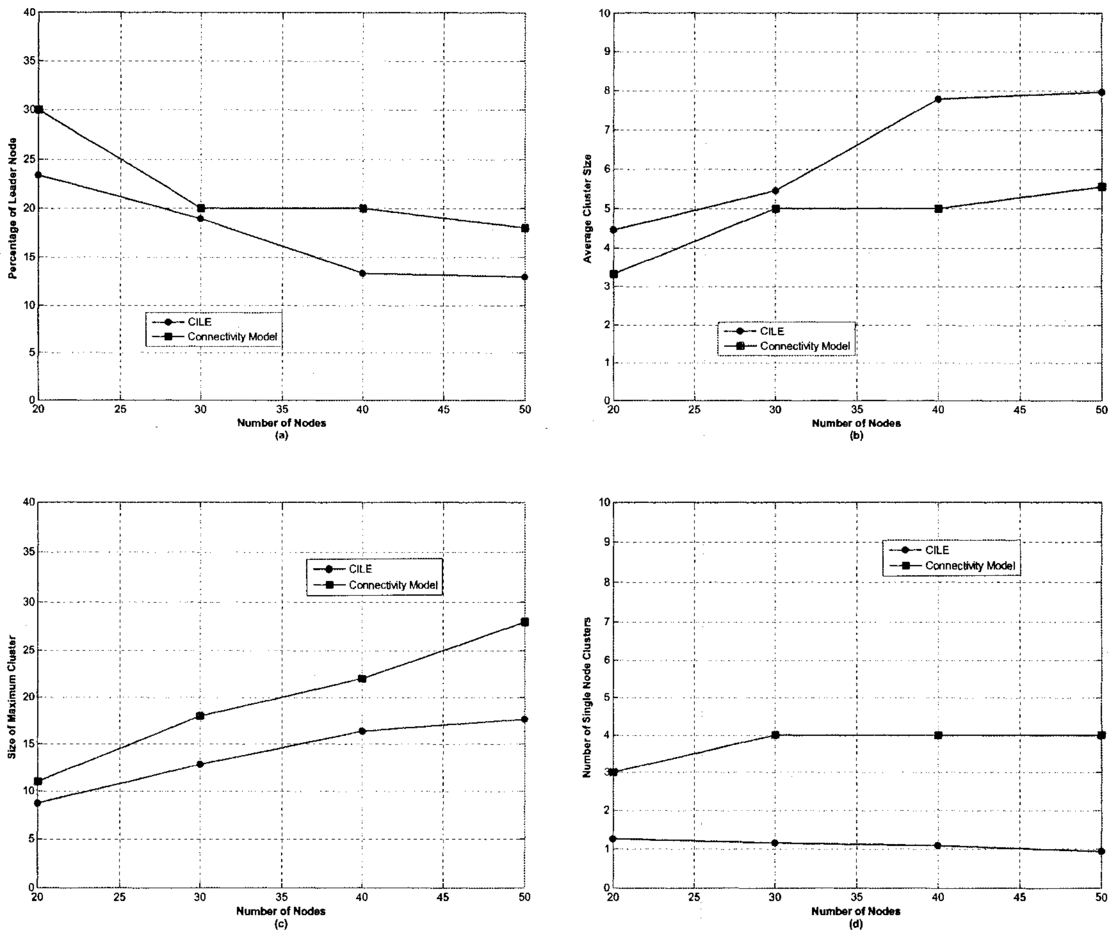


Figure 16: Comparison of cluster characteristics

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 16.(d). 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.

5.4 Summary

In this chapter, we compared our model with the other two models. We simulated the models both in static and dynamic network for different number of nodes. At first, we showed the negative impact of selfish nodes on the network. Selfish nodes decrease the life time of other nodes and also increase the security vulnerabilities of the network. The other models ignore completely the presence of selfish nodes.

We observed that our model can prolong the life time of the network and at the same time can balance the energy level of the nodes. Unlike other models, it prevents the nodes from dying faster. We also discovered some other advantages of our model. Our model reduces the number of leader nodes with uniform cluster size. Besides, it has less number of single node clusters while the maximum cluster size is smaller than connectivity model.

Chapter 6

Conclusion

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 more uniformly.

6.1 List of Publications

The following are the list of publications resulted from the thesis:

- **N. Mohammed**, H. Otrok, L. Wang, M. Debbabi, and P. Bhattacharya. Mechanism Design-Based Secure Leader Election Model for Intrusion Detection in MANET. Submitted to IEEE Transactions on Dependable and Secure Computing, 2008.
- H. Otrok, **N. Mohammed**, L. Wang, M. Debbabi, and P. Bhattacharya. A Cooperative Leader Election Mechanism for Intrusion Detection in Mobile Ad hoc Networks. Submitted to IEEE Communications Magazine.
- 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, Special Issue onăAlgorithmic and Theoretical Aspects of Wireless Ad Hoc and Sensor Networks, Vol. 31, No. 4, 2008, pages 708-721.
- **N. Mohammed**, H. Otrok, L. Wang, M. Debbabi, and P. Bhattacharya. A Mechanism Design-Based Multi-Leader Election Scheme for Intrusion Detection in MANET.

Proceedings of IEEE Wireless Communications & Networking Conference (WCNC), Las Vegas, Nevada, USA, March 2008.

- H. Otrok, **N. Mohammed**, L. Wang, M. Debbabi, and P. Bhattacharya. An Efficient and Truthful Leader IDS Election Mechanism for MANET. Proceedings of IEEE International conference on Wireless and Mobile Computing, Networking and Communications (WiMob), New York, USA, October 2007.

6.2 Future Work

Our current model can reduce the overall resource consumption of IDSs in a network but this model is suitable when the probability of attack is low. As a future work, we are planning to consider the tradeoffs between security and IDS resources consumption through adding more leaders to the network once the risk of attack is high. Thus, present model can be considered as *moderate* intrusion detection systems since a portion of the nodes' packets are monitored and analyzed. Leaders should be added optimally to the network according to the security. Hence, an optimal strategy needs to be formulated to dynamically adjust the number of IDS nodes according to the resource constraints and potential threats. Besides, we proposed an abstract model of the reputation system. Further investigation is needed to ensure the security of the proposed reputation system.

Bibliography

[1] M. J. Akobsson, S. Wetzel, and B. Yener. Stealth attacks on ad hoc wireless networks. *Proceedings of IEEE Vehicular Technology Conference, Vol.3, 6-9 Oct. 2003.*

[2] P. Albers, O. Camp, J. Percher, B. Jouga, L. Me, and R. Puttini. Security in ad hoc networks: a general intrusion detection architecture enhancing trust based approaches. *Proceedings of Intl. Workshop on Wireless Information Systems, April 2002.*

[3] T. Anantvalee and J. Wu. A survey on intrusion detection in mobile ad hoc networks. *Wireless/ Mobile Network Security, Springer, pp. 170-196, 2006.*

[4] T. R. Andel and A. Yasnisac. The invisible node attack revisited. *Proceedings of IEEE SoutheastCon , pp. 686-691, March 2007.*

[5] L. Anderegg and S. Eidenbenz. Ad hoc-VCG: A truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents. *Proceedings of the ACM MobiCom'03, San Diego, California, 2003.*

[6] F. Anjum, D. Subhadrabandhu, and S. Sarkar. Signature based intrusion detection for

wireless ad-hoc networks: A comparative study of various routing protocols. *Proceedings of IEEE VTC, Wireless Security Symposium, Florida, October 2003.*

[7] Farooq Anjum and Petros Mouchtaris. *Security for Wireless Ad Hoc Networks.* John Wiley & Sons. Inc., USA, 2007.

[8] S. Basagni. Distributed and mobility-adaptive clustering for multimedia support in multi-hop wireless networks. *Proceedings of the IEEE* 50^{th} *International Vehicular Technology Conference (VTC), September 1999.*

[9] S. Basagni. Distributed clustering for ad hoc networks. *Proceedings of Fourth International Symposium on Parallel Architectures, Algorithms, and Networks, 2009.*

[10] M. Bechler, H. Hof, D. Kraft, F. Pahlke, and L. Wolf. A cluster-based security architecture for ad hoc networks. *Proceedings of IEEE INFOCOM, 2004.*

[11] E. Belding-Royer and S. Das. Ad hoc on-demand distance vector (AODV) routing. *Network Working Group , Request for Comments 3561, 2003.*

[12] P. Brutch and C. Ko. Challenges in intrusion detection for wireless ad-hoc networks. *Proceedings of Symposium on Applications and the Internet Workshop, January 2003.*

[13] S. Buchegger and J. Le Boudec. Performance analysis of the CONFIDANT protocol (cooperation of nodes - fairness in dynamic ad-hoc networks). *Proceedings of the 3rd ACM Intl. Symposium on Mobile Ad Hoc Networking and Computing, June 2002.*

[14] L. Buttyan and J. Hubaux. Nuglets: A virtual currency to simulate cooperation in self-organized ad hoc networks. *Technical Report DSC/2001/001, Swiss Federal Institute of Technology-Lausanne, 2001.*

[15] S. Capkun and J. Hubaux. Secure positioning of wireless devices with application to sensor networks. *Proceedings of IEEE INFOCOM, 2005.*

[16] S. Carter and A. Yasinsac. Secure position aided ad hoc routing protocol. *Proceedings of Intl. Conference on Communications and Computer Networks, November 2002.*

[17] S. Chapkin, B. Bako, F. Kargl, and E. Schoch. Location tracking attack in ad hoc networks based on topology information. *Proceedings of IEEE International Workshop on Wireless and Sensor Networks Security, Vancouver, Canada, 2006.*

[18] K. Chen and K. Nahrstedt. iPass: an incentive compatible auction scheme to enable packet forwarding service in MANET. *Proceedings of the IEEE ICDCS'04, 2004.*

[19] H. S. Chiu and K. S. Lui. Delphi: Wormhole detection mechanism for ad hoc wireless networks. *Proceedings of Intl. Symposium on Wireless Pervasive Computing, 2006.*

[20] H. Choi, T. F. La Porta, and P. McDaniel. Privacy preserving communication in MANETs. *Proceedings of IEEE Communications Society Conference on Sensor, Mesh, and Ad Hoc Communications and Networks,San Diego, CA, June 2007.*

[21] E. H. Clarke. Multipart pricing of public goods. *Public Choice, 11:17-33, 1971.*

[22] B. J. Culpepper and H. C. Tseng. Sinkhole intrusion indicators in DSR MANETs. *Proceedings of First International Conference on Broadband Networks, 2004.*

[23] R. Dash, N. Jennings, and D. Parkes. Computational mechanism design: A call to arms. *IEEE Intelligent Systems*, pages 40–47, 2003.

[24] W. Du, L. Fang, and P. Ning. LAD: Localization anomaly detection for wireless sensor networks. *Proceedings of the IEEE International Parallel & Distributed Processing Symposium (IPDPS '05), April 2005.*

[25] D. Sterne el at. A general cooperative intrusion detection architecture for MANETs. *Proceedings of IEEE Workshop on Information Assurance, March 2005.*

[26] Alberto Escudero-Pascual, Thijs Holleboom, and Simone Fischer-Hübner. Privacy of location data in mobile networks. *Proceedings of the 7th Nordic Workshop on Secure IT Systems (Nordsec 2002), Karlstad University, 7-8 November 2002.*

[27] B. DeCleene et al. Secure group communications for wireless networks. *Proceedings of MILCOM, VA, October, 2001.*

[28] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP based mechanism for lowest-cost routing. *Proceedings of the ACM annual symposium on Principles of distributed computing, 2002.*

[29] D. Glynos, P. Kotzanikolaou, and C. Douligeris. Preventing impersonation attacks in MANET with multi-factor authentication. *Proceedings of International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, April 2005.*

[30] A. Grilo, M. Macedo, P. Sebastiao, and M. Nunes. Stealth optimized fisheye state routing in mobile ad-hoc networks using directional antennas. *Proceeding of IEEE Vehicular Technology Conference, Vol.4, pp. 2590-2596 , 30 May-1 June 2005.*

[31] Theodore Groves. Incentives in teams. *Econometrica, 41:617-631, 1973.*

[32] N. Gura, A. Patel, and A. Wander. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. *In Proceedings of the 2004 Workshop on Cryptographic Hardware and Embedded Systems (CHES), 2004.*

[33] S. Gwalani, K. Srinivasan, G. Vigna, E. M. Beding-Royer, and R. Kemmerer. An intrusion detection tool for AODV-based ad hoc wireless networks. *Proceedings of the Annual Computer Security Applications Conference, Tucson, AZ, December 2004.*

[34] L. Hu and D. Evans. Using directional antennas to prevent wormhole attacks. *Network and Distributed System Security Symposium, 2004.*

[35] Y. Hu, D. B. Johnson, and A. Perrig. SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks. *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applica- tions (WMCSA'02), pp. 3-13, June 2002.*

[36] Y. Hu, A. Perrig, and D. B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. *Proceedings of 8th Annual International Conference on Mobile Computing and Networking (MobiCom'02), pp. 12-23, September 2002.*

[37] Y. Hu, A. Perrig, and D. B. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. *Proceedings of the ACM Workshop on Wireless Security (WiSe), San Diego, California, USA, 2003.*

[38] Y.C. Hu, Adrian Perrig, and David B. Johnson. Wormhole attacks in wireless networks. *IEEE Journal on Selected Areas in Communications, February 2006.*

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

[40] Y. A. Huang, W. Fan, W. Lee, and P. S. Yu. Cross-feature analysis for detecting ad-hoc routing anomalies. *Proceedings of 23rd International Conference on Distributed Computing Systems (ICDCSŠ03), pp. 478-487, May 2003.*

[41] Y. A. Huang and W. Lee. Attack analysis and detection for ad hoc routing protocols. *Proceedings of 7th International Symposium on Recent Advances in Intrusion Detection (RAID'04), pp. 125-145, September 2004.*

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

[43] M. Jakobsson, X. Wang, and S. Wetzel. Stealth attacks on vehicular wireless networks. *Proceedings of IEEE Vehicular Technology Conference, 2004.*

[44] J.Green and J.Laffont. *Incentives in public decision-making.* Springer Netherlands, USA, 1996.

[45] O. Kachirski and R. Guha. Efficient intrusion detection using multiple sensors in wireless ad hoc networks. *Proceedings of Annual Hawaii International Conference on System Sciences (HICSS'03), January 2003.*

[46] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. *Proceedings of First IEEE International Workshop on Sensor Network Protocols and Applications, 2002.*

[47] M. Khabbazian, H. Mercier, and V. K. Bhargava. Wormhole attack in wireless ad hoc networks: Analysis and countermeasure. *Proceedings of the Global Telecommunications Conference, San Francisco, CA, USA, 2006.*

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

[49] I. Krontiris, T. Dimitriou, T. Giannetsos, and M. Mpasoukos. Intrusion detection of sinkhole attacks in wireless sensor networks. *Proceedings of 3rd International Workshop on Algorithmic Aspects of Wireless Sensor Networks, Poland, 2007.*

[50] L. Lazos and R. Poovendran. SeRLoc: Robust localization for wireless sensor networks. *Proceedings of ACM Sensor Networks, 2005.*

[51] C. E. Loo, M. Y. Ng, C. Leckie, and Marimuthu Palaniswami. Intrusion detection for routing attacks in sensor networks. *International Journal of Distributed Sensor Networks, December 2006.*

[52] R. Maheshwari, J. Gao, and S. R. Das. Detecting wormhole attacks in wireless networks using connectivity information. *Proceedings of 26th IEEE International Conference on Computer Communications, pp. 107-115, May 2007.*

[53] J. Marshall, V. Thakur, and A. Yasinsac. Identifying flaws in the secure routing protocol. *Proceedings of IEEE Performance, Computing, and Communications Conference, pp. 167-174, 2003.*

[54] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misehavior in mobile ad hoc networks. *Proceedings of Six Annual International Conference in Mobile Computing and Networking, pp. 225-265, Boston, MA, August 2000.*

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

[56] The VINT project. The network simulator ns-2. *http://www.isi.edu/nsnam/ns.*

[57] P. Michiardi and R. Molva. Analysis of coalition formaton and cooperation strategies in mobile adhoc networks. *Journal of Ad hoc Networks, Elsevier, 2005.*

[58] P. Michiardi and R. Molva. Prevention of denial of service attack and selfishness in mobile ad hoc networks. *RR-02-063, Institute Eurécom, France, 2002.*

[59] A. Mishra, K. Nadkarni, and A. Patcha. Intrusion detection in wireless ad hoc networks. *Proceeding of IEEE Wireless Communications, February 2004.*

[60] P. Morris. *Introduction to Game Theory.* Springer, 1st edition, 1994.

[61] E. C. H. Ngai, J. Liu, and M. R. Lyu. On the intruder detection for sinkhole attack in wireless sensor networks. *Proceedings of IEEE International Conference on Communications, Vol. 8, pp. 3383-3389, June 2006.*

[62] Peng Ning and Kun Sun. How to misuse AODV: A case study of insider attacks against mobile ad-hoc routing protocols. *Proceedings of the 4th Annual IEEE Information Assurance Workshop, pages 60-67, West Point, June 2003.*

[63] N. Nisan and A. Ronen. Algorithmic mechanism design. *Proceedings of STOC, 1999.*

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

[65] 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.*

[66] A. Perrig, R. Canetti, D. Tygar, and D. Song. The TESLA broadcast authentication protocol. *Cryptobytes (RSA Laboratories, Summer/Fall 2002), 5(2):2-13, 2002.*

[67] R. Poovendran and L. Lazos. A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks. *Wireless Networks, pp. 27-59, Feb. 2007.*

[68] Abderrezak Rachedi, Abderrahim Benslimane, Lei Guang, and Chadi Assi. A confident community to secure mobile ad hoc networks. *Proceedings of IEEE International Conference on Communications, 2007.*

[69] K. Sanzgiri, B. Dahill, B.N. Levine, C. Shields, and E. Belding-Royer. A secure routing protocol for ad hoc networks. *Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP '02), November 2002.*

[70] J. Shneidman and D. Parkes. Specification faithfulness in networks with rational nodes. *Proceedings of the ACM PODC'04, 2004.*

[71] N. Song, L. Qian, and X. Li. Wormhole attacks detection in wireless ad hoc networks: A statistical analysis approach. *Journal of Network and Computer Applications, Vol. 30, Issue 1, January 2007.*

[72] V. Srinivasan, P. Nuggehalli, C. F. Chiasserini, and R. R. Rao. Cooperation in wireless ad hoc networks. *Proceedings of IEEE INFOCOM'03, USA, 2003.*

[73] D. Subhadrabandhu, S. Sarkar, and F. Anjum. Efficacy of misuse detection in adhoc networks: Part I. *IEEE Journal for selected Areas in Communication, special issue on Security in Wireless Networks, 24(2), 274-290, 2006.*

[74] D. Subhadrabandhu, S. Sarkar, and F. Anjum. RIDA: Robust intrusion detection in ad hoc networks. *Proceedings of IFIP Networking Conference, Canada, May, 2005.*

[75] B. Sun, K.Wu, and U. W. Pooch. Alert aggregation in mobile ad hoc networks. *Proceedings of ACM Workshop on Wireless Security (WiSe'03) in conjuction with the 9th Annual International Conference on Mobile Computing and Networking, 2003.*

[76] Kun Sun, Pai Peng, Peng Ning, and C. Wang. Secure distributed cluster formation in wireless sensor networks. *Proceedings of the* $22^{n}d$ *Computer Security Applications Conference (ACSAC '06), Dec 2006.*

[77] L. Tamilselvan and V. Sankaranarayanan. Prevention of blackhole attack in MANET. *Proceedings of International Conference on Wireless Broadband and Ultra Wideband Communications, August 2007.*

[78] S. Vasudevan, B. DeCleene, N. Immerman, J. Kurose, and D. Towsley. Leader election algorithms for wireless ad hoc networks. *In the 3rd DARPA Information Survivability Conference and Exposition (DISCEX III), April 2003.*

[79] S. Vasudevan, J. Kurose, and D. Towsley. Design and analysis of a leader election algorithm for mobile ad hoc networks. *Proceedings of the* 12^{th} *IEEE International Conference on Network Protocols (ICNP), 2004.*

[80] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance, 16:8-37, 1961.*

[81] Ronghua Wang, Wenliang Du, and Peng Ning. Containing denial-of-service attacks in broadcast authentication in sensor networks. *Proceedings of Eighth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), 2007.*

[82] Yawen Wei, Zhen Yu, and Yong Guan. Location verification algorithms for wireless sensor networks. *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS 2007), Toronto, Canada, June 25-29, 2007.*

[83] X. Wu and D. K. Y. Yau. Mitigating denial-of-service attacks in MANET by distributed packet filtering: A game-theoretic approach. *Proceedings of the ACM symposium on Information, Computer and Communications Security, 2007.*

[84] P. Yi, Z. Dai, S. Zhang, and Y. Zhong. A new routing attack in mobile ad hoc networks. *International Journal of Information Technology, Vol. 11 No. 2, 2005.*

[85] M. G. Zapata. Secure ad hoc on-demand distance vector (SAODV)routing. *Proceedings of ACM Mobile Computing and Communication Review (MC2R), July 2002.*

[86] M. G. Zapata and N. Asokan. Securing ad hoc routing protocols. *Proceedings of the ACM Workshop on Wireless Security (WiSe 2002), September 2002.*

[87] Qinghua Zhang, Pan Wang, Douglas S. Reeves, and Peng Ning. Defending sybil attacks in sensor networks. *Proceedings of the International Workshop on Security in Distributed Computing Systems (SDCS-2005), June 2005.*

[88] Y. Zhang, W. Lee, and Y. Huang. Intrusion detection techniques for mobile wireless networks. *ACM/Kluwer Wireless Networks Journal, September 2003.*

[89] H. Zhou, M. W. Muttka, and L. M. Ni. Multiple-key cryptography-based distributed certificate authority in mobile ad-hoc networks. *Proceedings of IEEE Global Communications Conference (GLOBECOM), Vol. 3, pp. 1681-1685, 2005.*

[90] L. Zhou and Z. J. Haas. Securing ad hoc networks. *Proceedings of IEEE Network Magazine, vol 13, issue 6, pp. 24-30, 1999.*