MECHANISM DESIGN AND GAME THEORETICAL MODELS FOR INTRUSION DETECTION

HADI OTROK

A THESIS

IN

THE DEPARTMENT

OF

ELECTRICAL AND COMPUTER ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

CONCORDIA UNIVERSITY

MONTRÉAL, QUÉBEC, CANADA

DECEMBER 2008

Your file Votre référence
ISBN: 978-0-494-45674-3
Our file Notre référence
ISBN: 978-0-494-45674-3

ABSTRACT

Mechanism Design and Game Theoretical Models for Intrusion Detection

Hadi Otrok, Ph.D.

Concordia University, 2008

In this thesis, we study the problems related to intrusion detection systems in Mobile Ad hoc Networks (MANETs). Specifically, we are addressing the leader election in the presence of selfish nodes, the tradeoff between security and IDS's resource consumption, and the multi-fragment intrusion detection via sampling. To balance the resource consumption among all the nodes and prolong the lifetime of a MANET, the nodes with the most remaining resources should be elected as the leaders. Selfishness is one of the main problems facing such a model where nodes can behave selfishly during the election or after. To address this issue, we present a solution based on the theory of mechanism design. More specifically, the solution provides nodes with incentives in the form of reputations to encourage nodes in participating honestly in the election process. The amount of incentives is based on the Vickrey-Clarke-Groves (VCG) mechanism to ensure that truth-telling is the dominant strategy of any node. To catch and punish a misbehaving elected leader, checkers are selected randomly to monitor the behavior of a leader. To reduce the false-positive rate, a cooperative game-theoretic model is proposed to analyze the contribution of each checker on the catch decision. A multi-stage catch mechanism is also introduced to reduce

the performance overhead of checkers. Additionally, we propose a series of local election algorithms that lead to globally optimal election results. Note that the leader election model, which is known as *moderate* model is only suitable when the probability of attacks is low. Once the probability of attacks is high, victims should launch their own IDSs. Such a *robust* model is, however, costly with respect to energy, which leads nodes to die fast. Clearly, to reduce the resource consumption of IDSs and yet keep its effectiveness, a critical issue is: When should we shift from moderate to robust mode? Here, we formalize this issue as a nonzero-sum non-cooperative game-theoretical model that takes into consideration the tradeoff between security and IDS resource consumption. Last but not least, we consider the problem of detecting multi-fragments intrusions that are launched from a MANET targeting another network. To generalize our solution, we consider the intrusion to be launched from any type of networks. The detection is accomplished by sampling a subset of the transmitted packets over selected network links or router interfaces. Given a sampling budget, our framework aims at developing a network packet sampling strategy to effectively reduce the success chances of an intruder. Non-cooperative game theory is used to express the problem formally. Finally, empirical results are provided to support our solutions.

Acknowledgments

I would like to express my gratitude to Almighty GOD, the most Beneficent and the most Merciful, for granting me the ability and opportunity to complete this thesis.

I would like to thank my supervisors, Prof. Mourad Debbabi and Prof. Prabir Bhattacharya 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. Working with them was a very valuable experience for me. They deserve all my acknowledgements.

I would like to thank my examiner committee Professors Azzedine Boukerche, Chadi Assi, Joey Paquet and Roch Glitho for giving me the honor by being in my PhD committee. Their time and effort are greatly appreciated.

I would like thank my colleagues Azzam Mourad and Syrine Tlili who shared with me the precious years of my thesis. Also, I would like to thank my friend and colleague Noman Mohammed for his help and support. Moreover, further thanks go to all the members of the Computer Security Laboratory.

Finally, I am very grateful to my mother and wife for their understanding, encouragement and love and to my family members for their endless support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Mobile Ad hoc Networks (MANETs) are non-infrastructure dynamic networks where mobile nodes within the direct radio range can intercommunicate to form a network [6]. Mobile nodes that are not in the direct radio range can intercommunicate through other intermediate mobile nodes. Thus, the intermediate nodes play the role of both routers and hosts at the same time. MANETs have many useful uses in areas with non-infrastructure networks such as military operations. Furthermore, MANETs are relevant in situations where the wired communications have been destroyed or traditional networks are congested, such as in a rescue mission after a natural disaster. Moreover, MANETs could be used in remote areas to deploy telecommunications and internet services, which prompts MANET for commercial use. Thus, laptops, Personal Digital Assistants (PDAs), cell phones and other devices that are not in the transmission range of an access point can still access the internet. MANETs in different geographical areas can be connected with each other and with different types of networks over wired infrastructure-based networks [6].

In MANET, decision-making, key-distribution, routing and forwarding packets are usually decentralized and many of them depend on the cooperative participation of all the nodes [41]. This dependency of MANET on a decentralized paradigm allows an adversary to exploit new types of attacks to disrupt the normal operation of cooperative algorithms used in ad hoc networks [64]. In contrast to wired networks where an attack usually requires physical access to the network, MANET is particularly susceptible to many attacks ranging from passive eavesdropping to active interfering due to their open medium. Figure 1 shows a typical MANET with an external intruder. 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.

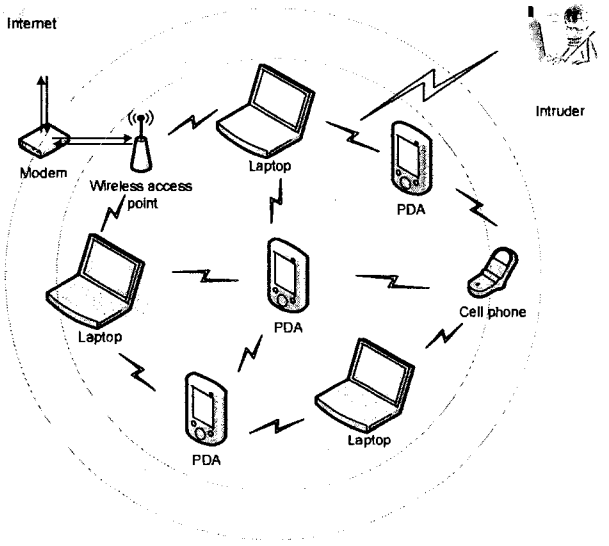

Figure 1: Mobile ad hoc network with external intruder

As the first line of defense, encryption techniques and firewalls are known to be insufficient for securing a MANET against all kinds of attacks. Hence, an Intrusion Detection System (IDS) must be introduced 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 that 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 raise the need for new IDS models that can cope up with the new security needs.

1.1 Motivations

Unlike traditional networks, a MANET has no fixed chokepoints/bottlenecks where an IDS can be deployed [6, 14]. Hence, a node may need to run its own IDS [4, 35] and cooperate with others to ensure security [38, 69]. 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 [41] or based on the connectivity [48]. 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 a 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 the 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 resource 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 the 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 that will balance the resources among nodes, taking into consideration selfishness. Relying on the leaders for providing the IDS service by examining a portion of all nodes' packets is suitable whenever the probability of attacks is low. This is usually known as a *moderate* intrusion detection model. However, a *robust* model where the victim nodes launch their own IDSs will be more desirable when

the probability of attack is high. This critical issue should be addressed and an answer must be given to the following question: When should we shift from moderate to robust mode?

So far, different solutions were proposed for IDS in MANET [6] that take into consideration the unique characteristics of such a network. Proposed models are able to analyze and detect intrusions that are targeting nodes in MANET. Now, the question is: *How to detect and thwart intrusions launched by a node in a MANET targeting another one or another type of networks?* Knowing that different MANETs are connected with each other using wired infrastructure networks. Detecting an intrusion in the network will be done by analyzing the traffic and looking for an unusual activity. Analyzing the traffic could be achieved by either considering the entire traffic or sampling a portion of the traffic searching for intrusions. Analyzing the entire traffic is considered costly since it needs time and consumes a lot of resources such as memory and CPU. On the other hand, analyzing the network using sampling is less costly but it has the problem of missing some intrusions due to its sampling budget constraint. Therefore, finding a strategy that is capable of enhancing the probability of detection using sampling is considered as a challenging problem, especially when we consider the case of smart intruders and cooperative intruders that are capable of sending an intrusion through multiple fragments [75].

In the following subsections, we motivate our work through two different examples.

1.1.1 Example 1

Figure 2 illustrates a MANET composed of ten nodes labeled from N_1 to N_{10}. These nodes are located in 5 one-hop clusters where the nodes N_5 and N_9 belong to more than one

cluster and have a limited energy level. We assume that each node has different energy level, which is considered as a private information. At this point, electing the nodes N_5 and N_9 as the leaders is clearly not desirable since losing them will cause a partition in the network and nodes will not be able to communicate with each other. However, with the random election model [41], the nodes N_5 and N_9 will have equal probability, compared to others, in being elected as leaders. The nodes N_5 and N_9 will definitely be elected under the connectivity index-based approach due to their connectivity indices [48]. Moreover, a naive approach for electing nodes with the most remaining energy will also fail since the nodes' energy level is considered as a private information and nodes might reveal fake information if that increases their own benefits. Finally, if the nodes N_2, N_5, and N_9 are selfish and are 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 the normal nodes to launch their IDS and thus die faster.

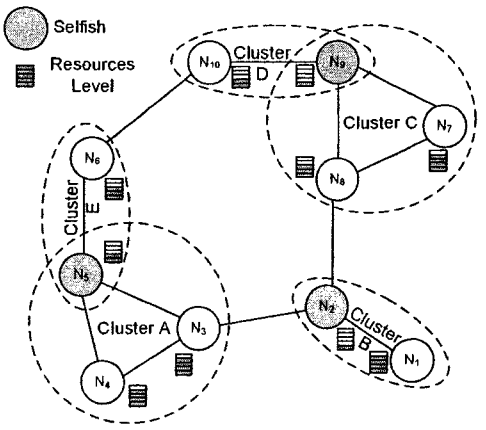

Figure 2: Example scenario of leader election in MANET

1.1.2 Example 2

Figure 3 illustrates a wired network where each node is a MANET or any type of networks. To evade the IDS located at the nodes and reduce the probability of detection, an attacker can follow the following two scenarios. First, an attacker located at node A can launch an attack targeting a victim in node I through dividing it into multi-fragments. This scenario is illustrated in Figure 3.a. To achieve this goal, an intruder selects a path to inject a fragment of the intrusion where another fragment is injected following the same strategy. Second, the multi-intruders divide an attack into multi-fragments where each fragment is launched by an intruder from different node/area. Figure 3.b illustrates this scenario where the intruders A and E inject a fragment of the attack by selecting a path to the victim in node I.

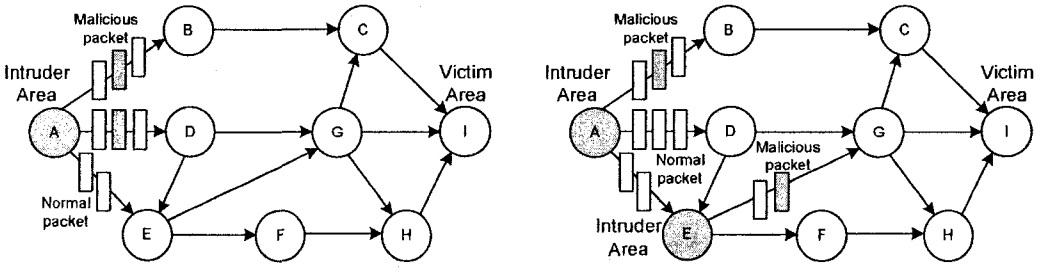

Figure 3: (a) Attacker in A targeting I (b) Multi-attackers (A and E) targeting I

To detect such types of intrusions, the IDS at the nodes has to cooperate, monitor and analyze all the incoming and forwarded traffic, which is considered costly with respect to the resource consumption. Thus, sampling a portion of the traffic would solve this problem but could lead to a security risk due to the network sampling budget constraint. This decreases the probability of detection of the IDS. Therefore, finding a strategy that could

help on detecting such intrusion via sampling is a challenging and an interesting problem. If the IDS is able to analyze the moves of the intruder and to predict them, then the IDS is able to distribute its sampling budget in a way that maximizes the probability of detection.

1.2 Objectives

In this thesis, our goal is to propose a game theoretic framework that studies the tradeoff between security and resource consumption in IDS. More specifically, the following are the objectives that will be addressed throughout this thesis:

- To reduce the resources consumption and prolong the lifetime of the IDS overall the nodes in a MANET by electing optimally a set of leaders in the presence of selfish nodes.
- To consider the tradeoff between the security and the resource consumption by adding new monitors according to the security needs.
- To detect multi-fragment intrusions via sampling.
- To demonstrate the performance of the proposed solutions through simulation and numerical results.

1.3 Proposed Approach

In this thesis, we propose a solution for balancing the resource consumption of IDSs among all nodes in a MANET while preventing nodes from behaving selfishly. To address the

selfishness behavior, we design incentives in the form of reputation to encourage nodes to participate honestly in the election scheme by revealing their analysis-cost. The cost of analysis is designed to protect the nodes' sensitive information (resources level) and ensure the contribution of every node on the election process (fairness). To motivate nodes in behaving normally in every election round, we relate the amount of detection service that each node is entitled to the nodes' reputation value. Thus, the intrusion detection budget will be distributed over the protected nodes according to node's reputation. Besides, this reputation value can also be used to give routing priority and to build a trust environment. The design of incentives is based on a classical mechanism design model, namely, the Vickrey-Clarke-Groves (VCG) [59]. The model guarantees that truth-telling is always the dominant strategy for every node during each election phase. Additionally, to prevent an elected leader from misbehaving after the election, we design a catch-and-punish mechanism to monitor the behavior of a leader with *checker* nodes. To reduce the false-positive rate of checkers, a cooperative decision game is formulated where the checkers are the players. We also introduce a multi-stage catch mechanism for reducing the performance overhead of checkers.

On the other hand, to find the globally optimal cost-efficient leaders, a leader election algorithm is devised to handle the election process, taking into consideration the possibility of cheating and security flaws, such as replay attack. The algorithm decreases the percentage of leaders, single node clusters, maximum cluster size and increases average cluster size. Last but not least, we address these issues in two possible settings, namely, Cluster Independent Leader Election (CILE) and Cluster Dependent Leader Election (CDLE). In

the former, the leaders are elected according to the received votes from the neighbor nodes. The latter scheme elects the 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. In addition, we support the correctness of the proposed methods through analysis and simulation. Empirical results indicate that our scheme can effectively improve the overall lifetime of a MANET.

Running this model in a non-secure environment raises the need for more nodes to launch their IDS according to the attack severity. Thus, more nodes should launch their own IDS according to the nodes' security risk. This will help to prolong the lifetime of nodes and to increase the nodes' security. The question we address here is: *What is the optimal threshold value needed to inform the victim node to launch its own IDS in order to reduce both resource consumption and security risk?* To answer this question, we formalize the tradeoff between the security and IDS resource consumption as a non-cooperative game between a leader-IDS and an attacker with incomplete information about the attacker. This game guides a leader-IDS and an intruder to derive their optimal strategy against each other. For the leader-IDS, the game solution derives the threshold for informing the victim node to launch its own IDS once the probability of attack exceeds the derived threshold. The game will be repeated such that in every election round the leader-IDS will be monitoring via sampling the protected nodes' incoming traffic and deciding according to the game solution whether to inform the victim node to launch its IDS or not. On the other hand, the attacker's strategy will be to attack once the probability of stepping into the robust mode (that is, the

victim node will be running its own IDS) is low. The empirical results indicate that our scheme can effectively reduce the resource consumption of the IDSs without sacrificing security.

To handle the intrusions between different MANETs and other type of networks connected via wired infrastructure networks, we develop a network packet sampling policy to effectively reduce the success chances of an intruder by finding the value of the game using a min-max strategy [65]. Non-cooperative game theory with complete information about the players is used to formally express our problems where the players are: (1) the cooperative intruders or a smart intruder (depends on which scenario we are solving) and (2) the intrusion detection system. This game theoretic model will guide the IDS to have an optimal sampling strategy in order to detect the malicious packets. The strategy for each intruder is the probability of choosing each possible path to send its malicious packet to the victim node. Consequently, the optimal strategy for the IDS is to assign the sampling rates to each link to maximize the probability of detection while not exceeding the total predetermined budget.

1.4 Thesis Contributions

In this thesis, we propose a game theoretical framework that considers the tradeoff between the security and resources consumption for an IDS. First, we propose a solution for balancing the resource consumption of IDSs among all nodes in MANET in the presence of selfish nodes. Our model is able to:

- Increase the overall lifetime of an IDS in MANET by truthfully electing the most cost-efficient node to handle the detection process on behalf of the whole cluster. This is achieved by balancing the resource consumption for the detection service among all the nodes in MANET.

- Motivate the selfish nodes to truthfully reveal their cost of analysis during a leader election. This is achieved by a reputation system based on the truth-telling mechanism VCG and by binding the reputation of a node to the amount of services the node is entitled to.

- Encourage an elected leader to carry out its responsibility of intrusion detection. This is achieved with a decentralized catch-and-punish mechanism using random *checker* nodes.

- Reduce false-positives, resources consumption and avoid malicious use of the punishment system. This is achieved by having an aggregated function based on checker's reputation and checker's observation followed by mapping this function to its corresponding security level. Cooperative game theory is used to analyze the contribution of each checker on the detection level.

- Adapt the model by adding more monitor nodes (IDS) according to the security needs taking into consideration nodes' resources consumption. This is done by computing the security threshold, by the leader-IDS, to inform the victim node to launch its own IDS.

Second, we propose a traffic sampling strategy based on game theory that considers multi-fragment intrusions launched by an intruder or cooperative intruders. The game is able to:

- Increase the probability of detection by reducing the success chances of an intrusion by finding the value of the game using a min-max strategy.
- Find the optimal sampling strategy of each player.

1.5 Outline of the Thesis

This thesis is organized as follows: In Chapter 2, we present the traditional IDS with its two main components: Data collection and analysis. According to the source of collected data, data collection is categorized into two categories: *Host-based* and *network-based.* On the other hand, data analysis is categorized into two kinds: *Anomaly based* and *signature based.* IDS can be classified according to the false positive and false negative rates. After presenting the traditional IDS, we describe the challenges facing the traditional IDS in MANET. This raises the need for new IDS models that can handle the security challenges that are unique to a MANET. A list of some of the attacks that are unique to a MANET is given. We categorize the IDS models in a MANET into two categories: *Non-cooperative* and *cooperative.* In the former category, nodes are detecting intrusions without cooperation, which reduces the communication overhead but could miss some intrusion. This is because some of the attacks in a MANET requires the cooperation among the nodes. In

the cooperative model, nodes are cooperating with each other to detect attacks and misbehaving nodes. This model is categorized into two kinds: *Non-cooperative* and *cooperative enforcement*. In the former, IDS models are classified into two categories: *Peer-to-peer* and *hierarchical (Cluster head)*. In these types of models, nodes are able to detect the attacks cooperatively without motivating the misbehaving nodes to cooperate. Note that the leader election is the key concept for the cooperative IDS model. Thus, we present the problems of the current leader election models followed by our proposed one. On the other hand, the cooperative enforcement mechanism motivates the misbehaving nodes to cooperate by giving them incentives and punishing the misbehaving ones.

In Chapter 3, we introduce game theory and mechanism design, which is a sub field of game theory. We define game theory and present its applications to intrusion detection. We present the two categories of games: *Non-cooperative* and *cooperative*. In the non-cooperative game, we show how to define a game and to solve it. Note that the non-cooperative games are of two types: *Non-zero-sum* and *zero-sum*. To solve any type of game, the equilibrium point must be calculated, which is done by: *Dominant strategy*, *Nash equilibrium*, *Minimax strategy*, or *mixed strategy*. All these techniques require complete information about the players that would not be available in many cases. To overcome this limitation, Bayesian games are given to solve games with incomplete information about the players. As an example, we illustrate the game between an IDS and an intruder with incomplete information about the intruder. Finally, we present mechanism design and its application to networks followed by the definition of the *social choice function*, *incentive compatible*, *direct revelation principle*, and *Vickrey, Clarke and Groves mechanism (VCG)*.

VCG is used to compute the needed payment to motivate the players to reveal truthfully their private information. Thus, truth telling is the dominant strategy.

In chapter 4, we describe our leader election mechanism where the cost of analysis function, reputation model and payment design are given. The cost of analysis function considers two main properties: Fairness and privacy. Additionally, the reputation model is given to show how reputations are used to motivate the nodes. To motivate the nodes, we relate the reputation value to an IDS sampling service. Thus, the leader will sample to the protected nodes according to their reputation. Moreover, it describes how the misbehaving nodes are caught and punished. The payments of CILE and CDLE are designed using the VCG where truth telling is the dominant strategy among all the nodes. Last but not least, we analyze our mechanisms against selfish and malicious nodes. We provide a "catch and punish mechanism" based on checkers to monitor the behavior of the leader after election. To reduce the false positive rate by the checkers, we use cooperative game theory to analyze the contribution of each checker node on the catch decision. Finally, to reduce the resource consumption of the checkers, we design a gradual catch model where checkers are added according to the behavior of the leader.

In Chapter 5, we devise the election algorithm that is needed to handle the election process. Moreover, we analyze informally the correctness and security properties of the given algorithms along with performance analysis. We show that our leader election model is suitable once the security risk is low. Once the security risk is high, nodes have to launch their monitors. To solve this problem, we propose a tradeoff model that considers the resource consumption and the security. Non-cooperative Bayesian game is used to

formulate the game between a leader and an intruder with incomplete information about the identity of the intruder. The solution of the game guides the leader to know his optimal threshold to inform the victim nodes to launch their monitors. Also, the game guides the intruder to find his optimal attack strategy. Finally, simulation results are given to show the performance of our model compared to others. Furthermore, we show that according to the security needs the victim node launches its own IDS, which reduces the resources consumption.

In Chapter 6, we consider the multi-fragments intrusion detection between a set of MANET or any type of networks that are located in a different geographical area. These networks are connected using the wired infrastructure-based network where in each network there is an IDS that is sampling the incoming and forwarded traffic and cooperatively analyzing these packets searching for intrusions. To evade the IDS, we consider two scenarios. First, an intruder divides an intrusion into multi-fragments where each fragment is injected into the network by selecting a path to a victim. Second, we consider the case of multi-intruders where each intruder injects a fragment of an attack by selecting a path to a victim. To maximize the probability of detection, we use the zero-sum non-cooperative game with complete information about the players to define the games of the two scenarios and to solve them. The solution of the games guides the IDS to know its own optimal sampling strategy and the intruder/multi-intruders to know his/their optimal attack strategy.

Finally, in Chapter 7 we conclude the thesis by providing a summary of the proposed models followed by a discussion of the future work and the list of publications derived from and related to the thesis.

Chapter 2

Intrusion Detection Literature Review

The open nature of a MANET and its dependency on a decentralized paradigm allows an adversary to exploit new types of attacks to disrupt the normal operation of cooperative algorithms used in ad hoc networks [64]. In contrast with the wired networks where an attack usually requires physical access to the network, MANET is particularly susceptible to many attacks ranging from passive eavesdropping to active interfering due to their open medium. 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.

Intrusion prevention systems such as firewalls and encryption techniques are known to be insufficient for securing MANET against all kinds of intrusions that can compromise the confidentiality, integrity or availability of the node or network. Hence, an intrusion detection system (IDS) must be introduced as a second line of defense for detecting intrusions

and consequently triggering an appropriate response. Detecting an intrusion in the network will be done by collecting and analyzing the traffic searching for an unusual activity. Collecting and analyzing the traffic could be achieved by either considering the whole traffic or by sampling a portion of the traffic. Analyzing the whole traffic is considered costly since it needs time and consumes a lot of resources such as memory and CPU. On the other hand, analyzing the network using sampling is less costly but it has the problem of missing some intrusions due to its sampling budget constraint.

Despite recent efforts on intrusion detection in a MANET [6], the existing solutions suffer from many limitations that are unique to a MANET. Like other services in MANET, the intrusion detection also demands cooperation among all the nodes. Such a cooperative intrusion detection model can yield an unacceptable performance and communication overhead, because all the nodes are usually required to analyze network packets. An apparent way to address this issue is to elect a leader node for handling the intrusion detection service on behalf of the whole cluster. However, the current solutions elect a leader node in a random manner [41] or based on the connectivity index [48] without considering the different energy level of nodes. Nodes with less resources thus will die faster, reducing the overall lifetime of the cluster. Existing solutions also ignore the potential selfish behavior of nodes that are not willing to consume their resources for serving others. At the same time, they benefit from others' services. The study of non-cooperative behavior (*selfishness*) under the cooperation enforcement discipline for solving routing problems has demonstrated many serious consequences of selfishness [61]. Similarly, the presence of misbehaving nodes also brings new challenges to IDS in MANET.

The rest of this chapter is organized as follows: Section 2.1 illustrates the traditional intrusion detection systems. Section 2.2 describes the new challenges of IDS in MANET. Section 2.3 lists the possible attacks targeting a MANET. This list shows the need for new IDS models, which are proposed in Section 2.4. The models are categorized into two categories: Non-cooperative and cooperative. The cooperative models are classified into two categories of mechanisms: Non-cooperative and cooperative enforcement. In the former, the models are divided into two classifications: Peer-to-peer and hierarchical (cluster-head). Moreover, in Section 2.4.3, we present the models that are based on the cooperation enforcement mechanism that can encourage selfish nodes to behave normally. Finally, in Section 2.5, we point out that the leader election is a significant issue for IDS in a MANET. It is the key concept of cooperative IDS model. Furthermore, we show the limitations of the present solutions followed by our proposed leader election model.

2.1 Traditional Intrusion Detection Systems

Intrusions are defined as attempts to compromise the confidentiality, integrity, availability or bypass the security of the system or network. Intrusion detection is the process of monitoring the system or the network by looking to the events occurring and then analyzes these events searching for intrusions. Intrusion Detection System (IDS) is defined as an automated system, which monitors and analyzes the activities of a computer or network system to determine whether there are any abnormal activities that violate the security. In other words, the IDS is based on a captured audit data and reasoning about evidence in the data

to determine whether the system/network is under attack [64]. Thus, intrusion detection system has two main components: Data collection and data analysis.

2.1.1 Data Collection

Detecting an unusual activity will be done through monitoring the system/network and collecting events. The sources of audit data can be a keyboard input, command-based logs, application-based logs, and network traffic. According to the type of the audit data collected, we can classify the IDS into two categories [6, 7]: Host-based and Network-based.

Host-based IDS (HIDS)

It 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, and monitoring real-time system usage. Host-based IDS does not depend on the network bandwidth, and is usually used in small networks, where each host dedicates its processing power to achieve the task of system monitoring. Notice that, running such type of intrusion detection can slow down the hosts and decline its performance. Host-based IDS is not affected by the use of end-to-end cryptography.

Network-based IDS (NIDS)

Generally, it runs at the switches, gateways, or routers in a wired network in order to analyze the captured packets that traverse through the network hardware interfaces. On

the other hand, a Mobile Ad hoc Network (MANET) does not have such types of network elements, where the IDS can collect audit data for the entire network. In the wired network, network traffic is monitored on the wired network segment, while in an ad hoc network, nodes can only monitor the network within their observable radio range. In contrary to the firewall, the network-based intrusion detection can analyze not only the header but the entire packet. They are able to look at the payload within a packet, in order to know which host application has been accessed, and to raise alerts when an adversary tries to compromise such application. Network-based, in wired network, can run as a black box to monitor the entire network. Figure 4 illustrates a traditional network IDS with data collection and analysis components. The NIDS is placed after the firewall to work as a second line of defense by collecting and analyzing the traffic looking for intrusions.

One of the drawbacks of the NIDS is its inability to detect all types of intrusions especially those that are launched using the evasion tools [75]. Such intrusions can be designed using multiple packet fragments that can be forwarded using multiple routes. This requires the NIDS sensors to collect and analyze all the traffic to detect such types of intrusions, which consumes a lot of resources. In the following section, we elaborate more about the analysis techniques. In this thesis, we will focus on the NIDS.

2.1.2 Data Analysis

Once the data is collected, the data analysis component analyzes the data for possible intrusions. Detecting an intrusion in the network will be done by analyzing the collected traffic

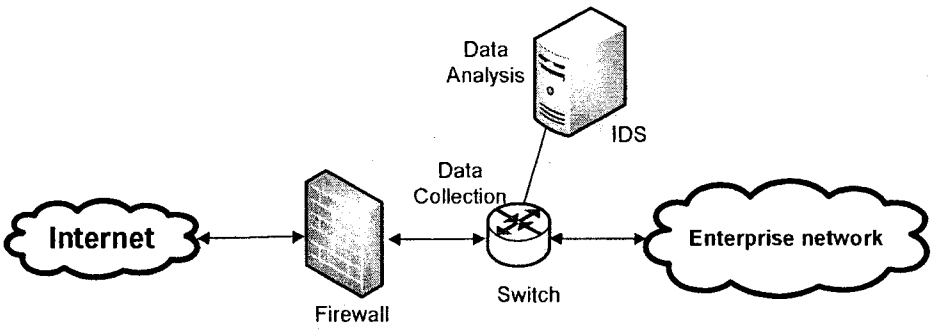

Figure 4: Traditional network intrusion detection system

looking for an unusual activity. Analyzing the traffic could be achieved by either considering the whole collected traffic or by sampling a portion of the traffic looking for intrusions. Analyzing the whole traffic in real-time is considered costly in terms of resources consumption. On the other hand, analyzing a portion of the traffic (sampling) is considered less costly but could lead to security risks by missing some intrusions due to the sampling budget constraint. Especially, the intrusions that are formed using multi-fragments could be missed since a specific number of fragments are required to detect such types of intrusions.

In [51], the authors have considered the problem of detecting intruding packets in a network by means of a network packet sampling. Since packet sampling and an examination in real-time could be expensive, the network operator has to devise an effective sampling scheme to detect the intruding packets injected into the network by an adversary. They take into consideration the scenario where the adversary has significant information about the network and can either pick paths to minimize chances of detection or could pick suitable network ingress-point if only shortest path routing was allowed. They have formulated the problem in a game-theoretic framework and the solution to this problem corresponded

to a max-flow problem from which the stable operating points were obtained. However, their approach is not practical when a practiced intruder or cooperative intruders divide the attack over multiple packets and transmits them through possibly different routes. We particularly take into account these special problems, and use the same network model as in [51]. We then build our game-theoretic models and formulate the sampling problem. We solve the games using the min-max approach to find the optimal sampling strategy for the IDS in order to detect these intrusion packets that are launched either by a smart intruder or by cooperative intruders.

The techniques that the data analysis component uses to analyze the data can be categorized into two kinds [4, 64]: Anomaly-based and signature-based detection.

Anomaly-based IDS

It flags the observed activities that abnormally deviate from the recognized normal usage as anomalies. It must first be trained using the normal data before it can be released in an operative detection mode. For example, the normal profile of a user may have the averaged frequencies of some system commands that are used during login sessions. If for a monitored session, the frequency is changed to a higher or a lower value, as compared to the average then an alarm will be raised. The main advantage of this model is that it can detect the unknown attacks. On the other hand, its disadvantage is that it has high false positive alarm rate when the normal user profiles, operating system, or network behavior vary widely from their normal behavior.

Signature-based or misuse detection IDS

It compares the activities of a user with patterns of well-known intrusions (signatures) for intruders attempting to compromise a system. It contains an internal table of anomalous patterns. This table represents the system traces that correspond to known attacks. If an activity matches a pattern in the table, an alarm will be raised. For example, a misuse rule for guessing a password can be "there are more than three failed login trials within one minute". The main advantage is that it can exactly and proficiently detect instances of known intrusions. On the other hand, the disadvantage of the misuse detection is that the anomalous patterns are based on known attacks; therefore, new attacks cannot be detected. Misuse detection needs the maintenance of a large central database of intrusion signatures whenever a new intrusion is discovered. Moreover, misuse detection can be fooled by a smart intruder. For example, an intrusion can be a mix between a normal activity and a real intrusion, which leads to a result that does not match any of the predefined patterns.

2.1.3 IDS Classification

Intrusion detection systems can be classified according to the false positive and negative rates. False positives (false alarms) are generated whenever the IDS considers normal data or traffic as intrusions. On the other hand, false negatives are the ratio of malicious activities where the IDS fails to detect. Additionally, resource consumption, such as memory, CPU and energy, is considered as one of the problems facing an IDS in the traditional and ad hoc networks. Thus, an efficient IDS must have low false positive and negative rates and consume less resources.

2.2 IDS Challenges in MANET

The decentralized nature, mobility, open wireless medium and dynamic topology of MANETs are unique characteristics that allows an attacker to exploit more vulnerabilities than traditional network architectures. MANET has no fixed chokepoints/bottlenecks where the firewall or IDS can be deployed. Moreover, the potential damages caused by a compromised node can be significantly higher than in traditional networks since all the nodes in a MANET are peers. 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 attack on the whole network. On the other hand, the detection and prevention techniques are seriously constrained by the more stringent resources constraints. All these limitations show that the traditional IDS cannot be used to provide the needed security for a MANET. To understand better the threat in a MANET, we need to identify the common attacks, which are presented in Section 2.3.

2.3 Unique Attacks to MANET

Attacks in MANET are usually classified into two categories: *Passive* and *active* [64]. In the former, an adversary can passively eavesdrop on the data without disrupting the network. In the latter, an adversary can launch an attack through the replication of data, modification of data, deletion of exchanged data, violating confidentiality, integrity, or authentication, inject erroneous messages, denial of service, and broadcasting false routing

information. The attackers external to a cluster of nodes in a MANET may launch attacks to prevent network services from working properly. The first line of defense, encryption and authentication, can usually prevent the attackers who are not members of the MANET from disrupting the traffic. However, the encryption and authentication can do very little against attacks launched by nodes who are legitimate members of the network. Such malicious nodes can cooperate with each other to compromise the network and prevent their attacks from being detected.

The following is the attack taxonomy for some of the known attacks against MANET [41]:

- *Blackhole*: A malicious node impersonates a destination node by sending a spoofed route reply packet to a source node that initiates a route discovery. In this way, a malicious node can withhold the traffic from a source node to a destination. It takes place when the intermediate node drops the packets after the agreement to forward them [58]. This can be due to several reasons, such as selfishness, packet overload, link broken or maliciousness. Watchdog and Pathrater mechanism is introduced to defend against such type of attack [58]. Moreover, blackhole attack [42, 43, 88] can be used to attack the routing control traffic. This happens when a source node broadcasts a Route Request (RREQ) packet searching for a destination. A malicious node, without checking its routing table, sends a Route Reply (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. In this way, a malicious node can drop data packets without forwarding them to the destination.

- *Sinkhole*: It is a more complex version of the blackhole attack [25, 53, 68]. 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 [49]. Moreover, it can be used to initiate other type of attacks [56], such as wormhole and eavesdropping. A wormhole attack would become more effective and easier to achieve since all the traffic will flow through the compromised node.

- *Speeding up*: Some routing messages, in several routing protocols, have the property that says "first in first accepted". Here, an attacker can simply send a malicious control routing message in order to block legitimate messages that arrive later.

- *Rushing*: It is an effective denial of service attack against most routing protocols [39]; such as AODV [12], Adriadne [38], ARAN [77] and SAODV [99]. In an on-demand routing protocol, a node finds a route to another one by broadcasting a ROUTE REQUEST packet. In order to limit the performance overhead, existing protocols usually forward only one ROUTE REQUEST for a route discovery, which leads to such an attack. A naive attacker can either keep the network interfaces of his neighboring nodes busy or forward packets faster than others, which increases the probability of being selected by the communicating nodes.

- *Wormhole*: It is a severe attack in a MANET that requires at least two intruders [21,36,40,57,74,76]. It can take place without compromising any nodes, authenticity of communication, or confidentiality. A tunnel is created between two intruders in order to disrupt the routing protocol by tunneling the routing control messages, i.e.,

prevent the discovery of any routes unless through the wormhole.

It is hard to discover since the path that is used to pass on the 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 [50]. In the hidden mode, the attackers act as two simple transceivers without requiring any cryptographic keys to launch the attack. In the participation mode, the attackers use valid cryptographic keys to participate in the routing as legitimate nodes.

- *Dropping of packets*: A malicious node drops packets, conditionally or randomly, that are supposed to be forwarded to other nodes. Malicious nodes are classified into two categories: Active [35, 41, 58] and passive (selfish) [61, 62, 81, 96]. The former are defined as the compromised nodes that are controlled by an attacker. On the other hand, the latter are selfish nodes that wants to save energy by not forwarding others' packets. This attack is due to the nature of MANETs where the cooperation between nodes is needed to forward their routing packets.

- *Malicious flooding*: Flood the whole network or some victim nodes with large amounts of data or control packets [35,41,93,97]. It 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 he does not know

the network topology. After selecting these IP addresses, the attacker launches 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 deny these nodes from receiving any new route requests.

- *Location-Related*: The location information of the nodes is considered as a sensitive matter since it can be used maliciously. Once the location information is discovered, it can be used by position-based routing protocols, which are good alternatives of topology-based routing protocols [18]. The adversaries will try to mislead the nodes in obtaining fake location information about others. This shows the need for secure localization techniques [17, 28, 54, 94] that prevent both the internal and external attackers from misinterpreting the location of nodes. On the other hand, these protocols are vulnerable to the location information disclosure attack [19, 22, 30]. Thus, the 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 the network.

- *Network partition*: An attacker partitions the network into k sub-networks, and nodes in different sub-networks cannot communicate with each other due to a route path problem introduced by the attacker [6].

- *Sleep Deprivation*: An attacker can exhaust the battery power of other nodes by requesting their services over and over again so they cannot go into an idle or power preserving the state [6].

- *Denial-of-Service (DoS)*: A large number of route requests, generated by an adversary, could be directed to a node such that the victim cannot send or receive normal packets from others [6].

- *Cache poisoning*: An adversary can compromise the information in the routing table through modifying its content, deleting information from it, or by injecting fake information to it [6].

2.4 IDS Models in MANET

From the list of attacks, we can see that the proactive approaches (i.e., cryptography and authentication) and other techniques (i.e., secure routing [37, 38, 98]) are not sufficient to guarantee security in MANETs. Moreover, the traditional IDS cannot be adapted to a MANET. Hence, new distributed and cooperative IDS models are designed to handle the security needs of MANET. The proposed models are classified into different categories as shown in Figure 5.

2.4.1 Non-Cooperative IDS Model

In this model, each node runs independently its own IDS without cooperating with the others to detect intrusions. Each node decides based on its own collected information. This architecture has many problems that effect its spread. This is because many of the attacks require a cooperation scheme in order to detect such intrusions. The well-known model that belongs to this architecture is the Watchdog-Pathrater, which is described below. This

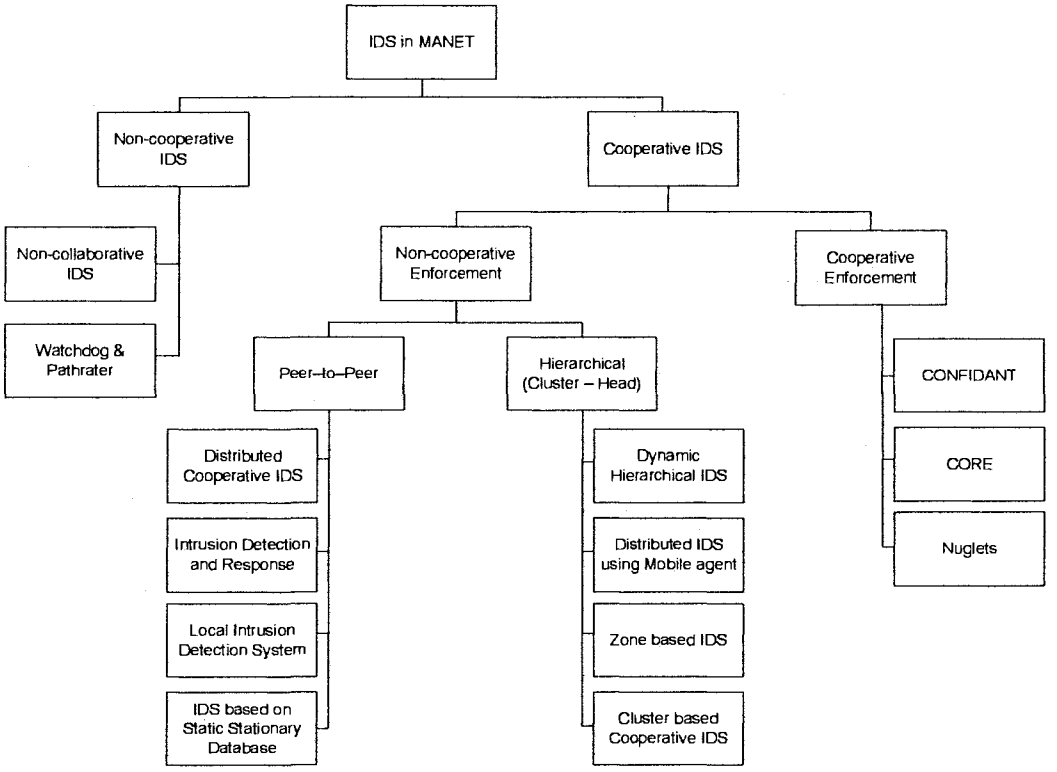

Figure 5: IDS categories in MANET

model is able to detect the selfish and malicious nodes.

Non-collaborative Intrusion Detection

In [84,85], a non-cooperative signature-based intrusion detection model has been proposed. The model is designed non-cooperatively since the message exchange between the nodes can increase the complexity and resource consumption in a MANET. Misuse technique is used since anomaly detection technique results high false positive rate in MANET where the network characteristic varies a lot and lacks a normal behavior. To enhance the efficiency of the model, a smaller subset of nodes are required to monitors the traffic which is

considered as a more practical solution.

Watchdog-Pathrater Approach

Watchdog and pathrater [58] are two techniques used to improve the throughput in MANET in the presence of misbehaving nodes. Where such type of nodes agree to forward packets but fail to do so for many reasons. A node may misbehave because it is:

- *Overloaded*: A node is facing problems; such as, buffer space or network bandwidth to forward packets.
- *Selfish*: A node is unwilling to consume its resources such as the battery life, CPU or bandwidth. Note that a selfish node expects other nodes to forward its traffic.
- *Malicious*: It is done by dropping packets.
- *Broken*: A node can face a software/network problem that prevents it from forwarding packets.

The watchdog goal is to identify the compromised nodes in the network. When a node sends packets to other nodes, the watchdog runs in a promiscuous mode to ensure that the next node forwards these packets. If the next node started to drop packets then it is considered as a misbehaving node and the watchdog counter will be incremented by one. If this counter comes to a value that is greater than a predefined threshold then the node is considered as a misbehaving node. On the other hand, the goal of the pathrater is to notify the routing protocols from using misbehaving nodes. The pathrater runs on each node in

order to choose a reliable route. Each node will rate its neighbor nodes in order to calculate a path metric by calculating the average node ratings in the path. One of the advantages of the watchdog is it can detect, in the case of Dynamic source Routing (DSR), a misbehavior not only at the link level but also on the routing level. The weaknesses of the watchdog can be through its inability to detect the misbehaving nodes in the presence of:

- *Ambiguous collision*: It is a routing problem that prevents node A from overhearing the transmission from node B.
- *Receiver collision*: Node A can tell whether the next neighbor node B forwards the packet or not. Also, it cannot check whether the next neighbor to B that is C has forwarded the packet or not.
- *Limited transmission power*: A node transmission power can be strong enough to be received by the previous node and weak to the next node.
- *False misbehavior*: It occurs when a node falsely notes other nodes as misbehaving.
- *Partial dropping*: A node can drop packets in a way, less than a predefined threshold, that could not be detected by the watchdog.

This model does not include any punishment procedure that can enforce nodes to behave normally. Hence, the misbehaving nodes can continue operating in the network and benefit from other normal nodes' services. Cooperation enforcement mechanism is proposed as a solution for such a problem, which is given in Section 2.4.3

2.4.2 Cooperative IDS Model

Cooperative IDS model is the second category of IDS in MANET where the nodes co-operate with each other in order to detect intrusions. Such a model is more suitable for a MANET since some of the attacks require information exchange among the nodes to detect efficiently such type of attacks. For example, an attack can be divided into multiple packets where each packet can be transmitted through different routes. There are two approaches to achieve cooperation: *Peer-to-peer* and *hierarchical (Cluster-head)* [6]. In the former, each node runs independently its own IDS to detect intrusions. Nodes share messages with others and thus cooperation is achieved. In the latter approach, each node has different responsibilities. 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.

Distributed and Cooperative Intrusion Detection System (DCIDS)

Zhang and Lee [100] describe a cooperative distributed intrusion detection model, where every node in the network participates in detecting and responding for intrusions. Moreover, every mobile node runs an IDS locally to perform local data collection and anomaly detection. Cooperative detection and global response can be activated when a node raises an anomaly with weak local evidence. This model takes into consideration two types of attacks: (1) Abnormal change in routing tables. (2) Intrusion detection in other layers. The internal structure of an IDS agent can be structured into six different modules as shown in Figure 6.

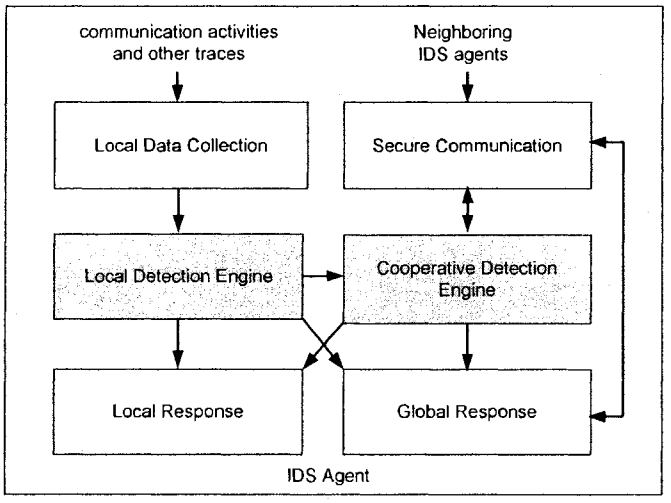

Figure 6: DCIDS model

- *Local data collection*: It collects local audit traces, communication activities and other traces that will be transferred to the local detection engine.

- *Local detection engine*: It uses the data passed by the local data collection to detect the local anomalies.

- *Local response*: It runs actions locally and performs the tasks such as alerting the local mobile node.

- *Global response*: It cooperate actions among all the neighbors in order to initiate a cooperative response, such as initiating a re-authentication process.

- *Cooperative detection engine*: It is triggered whenever there is an inconclusive evidence of an intrusion.

- *Secure communication*: It provides a secure communication channel among all mobile nodes. It is used by the cooperative detection engine and global response.

The advantage of this model is in its design since it is modeled based on the anomaly detection technique, since misuse technique uses a central database, which does not exist in an ad hoc network. On the other hand, this model suffers from performance penalties, resources consumption and false alarm rates.

Intrusion Detection and Response Model

Intrusion Detection and Response Model (IDRM) was proposed in [13] to enhance the security of the Ad hoc On Demand Distance Vector (AODV) routing protocol. This model is an extension of the above one (i.e., the DCIDS). Each node runs its own IDRM and use neighborhood information in order to detect misbehaving nodes. Once the misbehavior count exceeds a predefined threshold, it sends this information to other nodes in order to take a global response. When the other nodes receive this information, they check their local malicious count, for this malicious node, whether it is greater than a threshold value. If this happened, it sends a special packet called MAL to the entire network reporting an intrusion. If another node suspects that the node is compromised, it sends another special packet called REMAL reporting an intrusion node is detected. If two or more nodes report the same result, a special packet called PURGE will be transmitted to the entire network in order to drop the node from the network. In this case, all the nodes that are using this node for routing their packets will find a new routing path. This model describes several types of attacks such as: Distributed false route request, DoS and Masquerade. The main

disadvantages of this model are: Resources consumption and traffic overhead.

Local Intrusion Detection System (LIDS)

The LIDS is distributed in nature that uses mobile agents to collect and process data at remote hosts [2]. LIDS uses the data stored in the Management Information Base (MIB) as the source audit data. Note that, MIB is the database used by Simple Network Management Protocol (SNMP) to collect information of different objects. Mobile agents will be used to transfer SNMP requests to remote hosts and to process data at the source to reduce communication overheads. LIDS can use either misuse or anomaly for intrusion detection process. Moreover, LIDS uses Intrusion Detection Exchange Format (IDMEF) and Intrusion Detection Exchange protocol (IDXP) to ensure that the IDS running on a broad range of platforms can still interact and exchange intrusion information. LIDS disadvantage is that it does not take into consideration the false information broadcasted by compromised nodes. The LIDS architecture consists of several components that are shown in Figure 7.

- *Local LIDS Agent*: It 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*: It provides 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.

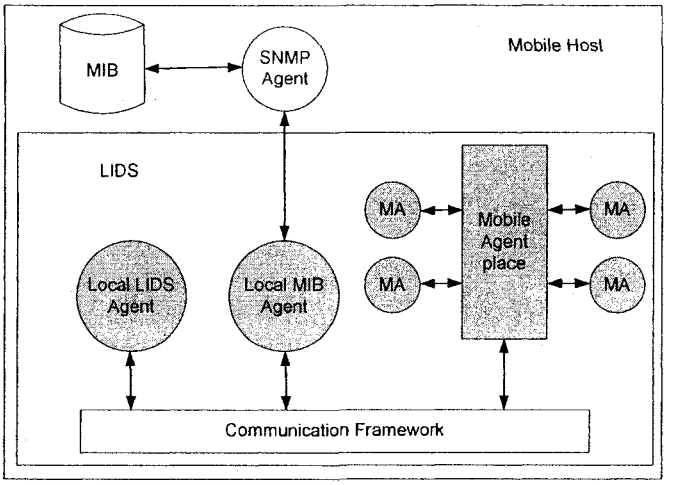

Figure 7: LIDS model

- *Communication Framework*: It is used to facilitate internal and external communication with LIDS.

- *Mobile Agent (MA)*: It collects and processes data on other nodes. Then, the collected results are sent to LIDS or another node for more investigations.

- *Mobile Agent Place*: It is responsible for a security control of mobile agents.

Intrusion Detection Architecture Based on a Static Stationary Database

This distributed and cooperative IDS model is proposed in [80] to decide when and how a MANET is being compromised. The architecture of this model is divided into two parts:

- *Mobile IDS agent*: It resides in each mobile node to cooperatively decide whether the network is under attack or not. Each agent has five parts: (1) Local Audit Trail is responsible for collecting network traffic and system audit data. (2) Local Intrusion

Database (LID) is a local database that stores all necessary information needed by the IDS; such as known signature attacks. (3) Anomaly Detection Module (ADM) is responsible for detecting a different type of anomaly. (4) Misuse Detection Module (MDM) identifies known signature attacks that are defined in the LID. The ADM and MDM are connected directly to the LID to determine if an intrusion is taking place. (5) Secure Communication module provides a secure channel to exchange information between different IDS agents. The ADM and MDM use this channel to cooperatively detect and respond for intrusions.

- *Stationary Secure Database (SSD)*: It contains known signatures and stores patterns of each user's normal activity. Moreover, it acts as a trusted party that contains the latest misuse signatures and patterns of normal user's activity. The SSD could be updated by the administrator of the network. In addition, the LID of the node is connected to the SSD in order to update its signatures and patterns. Note that, SSD is so difficult to be deployed in an ad hoc network environment with the absence of a central management point.

Dynamic Hierarchical Intrusion Detection System

Since the nature of MANETs is distributed and dynamic, it raises the need for dynamic hierarchical IDS architecture which is proposed by [82]. It divides the network into clusters with cluster-heads and different role for each node, as shown in Figure 8. Each node has the capability of monitoring, analyzing, responding to detected intrusions (if there is enough evidence) and forwarding collected data to cluster-heads. A cluster-head like other nodes

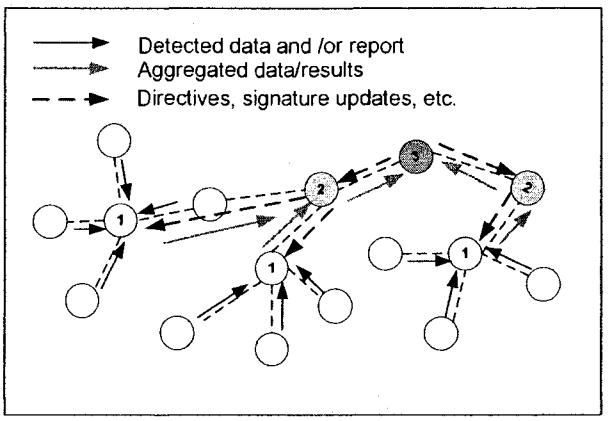

Figure 8: Hierarchical IDS model

is responsible for local and global detection. Each cluster-head, in this model, has different role according to its level. 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 (one-hop members), resistance to compromise, processing power, storage capacity, energy level, etc. This model suffers from selfishness where nodes might not participate to be elected as cluster-head.

Distributed Intrusion Detection Using Mobile Agents

The main contribution of this model is the efficient distribution of mobile agents with a specific IDS task according to their functionality in the MANET [48]. Moreover, it restricts computation intensive analysis of overall network security state to a few nodes. The

election of these nodes is done dynamically in order to do not allow the overall network security depends on any particular node. The network is divided to several clusters with an elected cluster head for each cluster. The selected cluster heads collect all packets within the communication range and analyze the packets for known patterns of attacks. 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 [48]. The proposed IDS is built on a mobile agent framework, which is shown in Figure 9. There are three major agents:

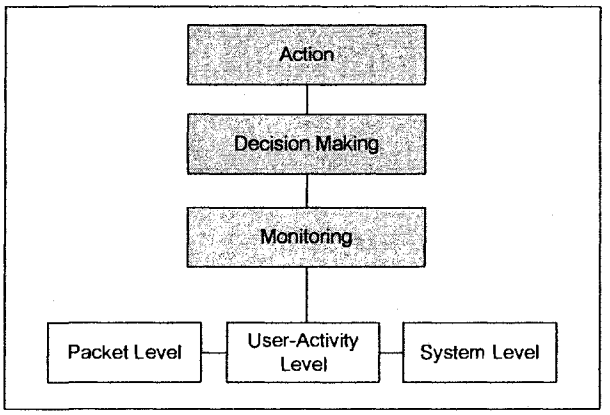

Figure 9: Multiple-sensor based IDS architecture for MANET

- *Monitoring agent*: It is categorized into three different sensors:
  - User activity level and system level: They are located at the local detection agents, which are located at each node in the network. These agents search for malicious activities on the host node, such as unusual process memory allocation, abnormal CPU activities, or user operations such as invalid login. If an anomaly is detected then the monitoring agent reports to the decision agent in

order to terminate the suspicious process.

  - Packet monitoring level: If more conclusive evidence is gathered about the malicious node, which is detected by the local detection agent, from the packet monitoring level, then an action is undertaken by the network agent on that node. To preserve the total computational and battery power of mobile hosts, few nodes will run their packet monitoring agent.

- *Decision making agent*: It contains a state machine for all the nodes within the cluster it belongs to. It is responsible for deciding whether a node is compromised from the information gathered by the monitoring agent at each node.

- *Action agent*: When an intrusion is detected, the decision agent sends a command that an action must be taken.

Zone Based Intrusion Detection System (ZBIDS)

ZBIDS divides the whole network into non overlapping zones, where nodes are categorized into two types: Interzone or intrazone [86]. Nodes with physical connections to other zone's nodes are known as interzone nodes, whereas zone internal nodes are known as 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. Each node has an IDS similar to the one proposed in Section 2.4.2. The interzone nodes are responsible for global aggregation and

correlation while intrazone nodes are responsible for local data analysis.

Cooperative Intrusion Detection System

This model [41] is an extension of the model proposed in [100], where a set of intrusions can be identified with their corresponding sources. The intrusion detection system is anomaly based where nodes compute different features of the network. Since it takes significant computational cost to obtain traffic related features, the network is divided into clusters with cluster head for each cluster. The cluster head helps to compute the traffic related features and the member nodes compute and transmit the non-traffic related features to the cluster head. Thus, the overall cluster-wise feature collection cost is reduced since a lot of features are traffic related. The authors propose a random election procedure for electing cluster heads. An elected cluster head is responsible for detecting intrusions for a predefined period of time. In addition, a cluster recovery protocol has been proposed to adjust lost nodes (nodes without leader) due to mobility reasons. This model does not consider the selfish behavior of nodes, which could significantly reduce the performance of the model.

2.4.3 Cooperation Enforcement Mechanisms

There are some other IDS models that are based on cooperation enforcement [15, 16, 61]. These techniques mainly focus on the selfish behavior in routing (i.e., forwarding others packets). Selfish nodes can degrade the performance of the network by dropping others' packet. These proposed techniques motivate selfish nodes through providing incentives and

punishing them in case of misbehavior.

CONFIDANT Mechanism

Watchdog-Pathrater model, which has been presented before, cannot motivate selfish nodes to participate in MANET. Thus, new models are needed to solve such a problem. To motivate the selfish nodes in routing, CONFIDANT (Cooperation Of Nodes: Fairness In Dynamic Ad hoc NeTwork) [15] proposes a reputation system where each node keeps track of the misbehaving nodes through the use of watchdog. The reputation is used to evaluate the routing and forwarding behavior of the node according to the used network protocol. When a node detects a misbehaving node, it sends a warning message, called ALARM, to others. Once a node receives such a message, it evaluates how trustworthy the message is based on the source's reputation. Thus, the reputation value reflects who to trust. 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 a misbehaving node such as excluding it from the routing paths. 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 (false ALARM) about a node in order to be punished.

CORE Mechanism

CORE [61] 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. Negative reputation is not shared with other nodes to overcome the problem of CONFIDANT, which leads to DoS attack. Thus, only positive information is shared among nodes. CORE has three main components:

- *Monitoring*: It uses the watchdog technique to detect a misbehaving node. CORE proposes a watchdog sampling frequency whenever the traffic density is heavy. This will reduce the rate at which packets are monitored.

- *Reputation Manager*: Reputation is calculated and updated through direct observations provided by the local monitor or indirect observations provided by other entities in the network. Once the reputation manager receives a previous direct observations from the monitoring component, it saves them in the form of a reputation table. This table can be accessed by the punishment system, which uses it to provide or deny service to nodes. Repeated game theory is used to evaluate the reputation of the node. Each node observes the previous m opponent's move where cooperation is rated as $+1$ and defection as -1. Then the average of the moves is calculated to evaluate the reputation.

- *Punishment*: The punishment component bases its actions according to the information provided by the reputation manager. If the reputation is ≥ 0 then the node will

cooperate otherwise defect.

Nuglets Mechanism

This model uses virtual currency known as "Nuglet" as incentives to motivate nodes in MANET to cooperate and discourage them from overloading the network [16]. Two models are proposed: Packet Purse and Packet Trade. In the Packet Purse model, each packet is loaded with nuglets by the source and each forwarding node takes some of the cash loaded for its forwarding service. The packet will be delivered to the destination, if the source has correctly calculated the needed amount of nuglets. The advantage of this model is that it discourages nodes from overloading the network. On the other hand, the drawback of this model is that it requires the source to know exactly the required number of hops the packet needs to be delivered to the destination. To overcome this problem, Packet Trade model proposes that each packet does not carry nuglets but is traded by the relay nodes for cash. Each relaying node buys the packet from the previous node on the path and sells it to the next one for more cash in order to gain some cash. Thus, the destination will pay for the packet and not the source, which will be an advantage over the Packet Purse model. On the other hand, this will lead to malicious flooding attack since packet generation is not charged.

2.5 Applications of Leader Election

Leader election is a significant issue of MANET and the key concept of cooperative IDS model. 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 to 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 [41, 48, 82] of cooperative IDS. Leader election is a significant issue of MANET and has been addressed in different research proposals [9, 52, 87, 89].

Like Intrusion Detection System (IDS), leader election is needed for routing [10] and key distribution [11, 27] 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 energy saving mode. Hence, the performance of the network significantly depends on the efficient clustering and leader election algorithms.

2.5.1 Problems of Leader Election Techniques

Many clustering and leader election algorithms have been proposed. These algorithms can be classified into two categories [87]: Cluster-first or leader-first. In cluster-first approach [52], a cluster is formed then nodes belonging to that cluster elect a leader node. In the

leader-first approach [9], 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 randomly, based on connectivity (node's degree) or based on a node's weight. Here, the weight normally refers to the remaining energy of a node [90]. In random election [41] 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 a 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 [48], nodes with highest node's 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 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 [10] or there exist a 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. Nodes might behave selfishly by lying about their resources level to avoid being elected as a 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.

2.5.2 Proposed Leader Election Model

Our proposed solution elects leaders based on their energy level by following the cooperation enforcement and game theory concepts. Therefore, our model has the following features:

- Motivate selfish nodes to cooperate during the election process.
- Calculate the reputation of the nodes based on mechanism design.
- Punish the misbehaving leader through a cooperative decision model based on cooperative game theory.

2.6 Summary

In this chapter, we discussed the traditional intrusion detection systems and their components and classifications. We showed the challenges of traditional model in MANET, which raises the need for designing new models that can handle the unique attacks targeting MANET. To make the chapter self-contained, we listed the possible attacks targeting ad-hoc networks and showing that proactive approaches are not sufficient to prevent such attacks. Thus, intrusion detection system is needed as a second line of defense to defend ad-hoc networks. We categorized IDS models in MANET into two main categories: Non-cooperative and cooperative. The cooperative models are classified into two classifications: Non-cooperative and cooperative enforcement mechanisms. 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. Finally, we showed the most prominent cooperation enforcement mechanisms that are used to motivate selfish nodes to participate in MANET. Our proposed leader election approach calculates the reputation of the nodes based on mechanism design and punishes a misbehaving node through a cooperative punishment system based on cooperative game theory.

Chapter 3

Game Theory and Mechanism Design

Game theory has been successfully applied to many disciplines including economics, political science, and computer science [8,65]. It has been used to formally solve some network problems, where different players have different strategies for network usage. It is a powerful tool in studying the relationships that are established and broken in the course of cooperation and competition. It usually considers a multi-player decision problem where multiple players with different objectives can compete and interact with each other by forming a coalition or possibly threaten each other taking actions under uncertainty. Finally, players will either receive some benefits, payoffs, or loss some rewards or payoffs.

Mechanism design is a sub-field of microeconomics and game theory [59]. Mechanism design uses the game theoretic [65] 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/maliciously as an example. 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.

To make the thesis self contained, we discuss in this chapter the basics of game theory and mechanism design, which are important to better understand our models. A more general overview of game theory can be found in [8, 65] while for mechanism design can be found in [26, 45, 59, 71].

This chapter is organized as follows: Section 3.1 gives a definition for game theory and its classifications. Section 3.1.1 provides the related work for applying Game theory to IDS. Section 3.1.2 shows how games are defined followed by, in Section 3.1.3, how to solve the game in the case of non-cooperative for both categories: *Non-zero sum* and *zero-sum*. Moreover, *dominant strategy, Nash equilibrium and minimax strategy* with illustrative examples are given followed by an example of how to model the IDS game. Additionally, we solve the IDS game for the case of incomplete information about the intruder using Bayesian game. Section 3.1.4 presents *cooperative game theory*. Section 3.2 illustrates the basics of mechanism design, where the *quasi-linear utility function, incentive compatible mechanism and revelation principle* are given. Finally, the *VCG mechanism*, which is an incentive compatible direct revelation mechanism, is presented with an example.

3.1 Game Theory

Game theory is a formal way to analyze *interactions* among a *group* of *rational* players that behave *strategically*. Where,

- *Group*: The decision makers in any game are known as players and the set of players is referred to as a group.
- *Interaction*: The decision each player makes in a group has an effect on at least one other player.
- *Strategy*: Each player has a set of actions to choose from, which is referred to as the strategy of the player. A strategy is a complete decision rule that defines the action an agent will take in each case of the game. Each strategy the player chooses affects the game's result.
- *Rational*: A rational player always wants to maximize his/her own benefits. Thus, each player chooses his/her best action, considering the other players decision.

Game theory classifies games into two categorizes: *Non-cooperative* and *cooperative*. Non-cooperative games are games with two or more players who are competing with each other. On the other hand, cooperative games are games with multi-players cooperating with each other in order to achieve the greatest possible total benefits.

Non-cooperative games are classified into two categories: *Zero-sum* and *non-zero-sum*. The zero-sum game, in the case of two players, is what one player wins, the other losses.

Whereas, non-zero-sum game is a game where the players' interests are not always in direct conflict so that there are opportunities for both to gain.

3.1.1 Game Theory Applications to IDS

To predict the optimal strategy used by the intruders to attack a network, the authors of [51] model a non-cooperative game-theoretic model to analyze the interaction between an intruder and IDS in a wired infrastructure network. They solve the problem using a zero-sum non-cooperative game with complete information about the intruder. In the complete information game, the type, strategy spaces, and payoff functions of both players are known. In [3], the authors aim at demonstrating the suitability of game theory for development of various decision, analysis, and control algorithms in intrusion detection. They address some of the fundamental network security tradeoffs, and give illustrative examples in different platforms. They propose two different schemes based on game theoretic techniques and consider a generic model of distributed IDSs equipped with a network of sensors. Bayesian Nash game is used in [55] to analyze the interaction between the intruder and defender in static and dynamic scenarios. The authors provide a hybrid detection approach with lightweight and heavyweight monitoring systems.

3.1.2 How to Define a Game?

A game must contain the following:

1. A list of players P=1, 2, . . . , I. Here, we consider the game of two players where the the players, for example, want to meet each other in Montreal.

2. Each player can select his/her strategy from a strategy set S. For example, the strategy set for each player for meeting each other is $S_1 = S_2 = \{$Coffee shop, Restaurant$\}$. Strategies are classified into two types: Pure and mixed. Pure strategies are deterministic while a mixed strategy is the probability distribution over the set of pure strategies so that a strategy is selected with a certain probability.

3. Strategy profile is the outcome of the game. It is the set of all the strategies chosen by the players. For example, the possible outcomes of the game of meeting are (Coffee shop, Coffee shop), (Coffee shop, Restaurant), (Restaurant, Coffee shop), and (Restaurant, Restaurant). Formally, the set of strategy profiles or outcomes of the game is defined as $S = S_1 \times S_2$.

4. Each player has preferences over the outcomes of the game. Knowing that, players cannot have preferences over the strategies. Utility function of a player i can be considered as a transformation of the outcome to a real number. 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 that will increase its expected utility given the preferences of the outcomes, the structure of the game and the belief of others strategies.

   In the meeting example, players care for meeting each other, which is the output of the game ignoring where they will meet each other, which is the strategy chosen by the players. Preferences over outcomes are represented through a utility function u. Therefore, $u_i = 1$ if both players meet each other and zero otherwise.

The two players and their corresponding strategies could be represented by the following table, where the columns represent the strategies of player 1 and the rows represent the strategies of player 2. The intersection between the row and the column represents the payoff of player 1 and 2.

Strategy	Coffee shop	Restaurant
Coffee shop	1,1	0,0
Restaurant	0,0	1,1

Formally, a normal game consists of:

- A finite set of players $P = \{1, 2, \ldots\}$.
- Strategy sets S_1, S_2, ..., S_I.
- Payoff function u_i.

3.1.3 Non-Cooperative Game Theory

Non-cooperative games [65] are games with two or more players that are competing with each other. A two players game where one player wins what the other losses is called two-person zero-sum game. Matching pennies is the best example for a zero-sum game. It is a two players game where player 1 wins a dollar from player 2, if both choose the same action, otherwise player 1 loses a Dollar.

After we have defined a game, the question that we shall address here is: How to solve a game and find the optimal strategy of each player? In other words, how to play a game

and find its solution? In the following sections, we shall use examples to illustrate how a game is solved.

Zero-Sum Non-Cooperative Game with Dominant Strategy

Zero-sum game with two players is a game in which the player one looses what the player two wins. We prefer to explain the concept of two-person zero-sum game through an example. The following table presents the payoffs and strategies of two players where the intersection between the rows and columns present the payoff gained by player 1, if the payoff is positive and a loss otherwise. Knowing that, the rows present the strategies of player 1 and the columns present the strategies of player 2.

Player 2				
Strategy		S_1	S_2	S_3
Player 1	S_1	1	2	4
	S_2	1	0	5
	S_3	0	1	-1

Before starting to solve the above example, we have to note that both players in the game are rational. Solving the game will lead the players to know their corresponding optimal strategy that will be used by each player. Here, we apply the concept of *dominant-strategy* to rule out a succession of inferior strategies until one choice remains. A strategy is dominated by another strategy if the second strategy is at least as good as the other one regardless of what the other player does. Therefore, the dominated strategy is eliminated directly from the table. Knowing that, players are rational and therefore they will not play

the dominated strategy. Formally, it is expressed as follows:

$$u_i(S_i^*, S_{-i}) \geq u_i(\acute{S}_i, S_{-i}), \forall S_{-i}, \forall \acute{S}_i \neq S_i^*$$

where, S_i^* is the dominant strategy, $\acute{S}_i$ is any other strategy and S_{-i} is the selected strategy by other players.

In our example, strategy 3 of player 1 is dominated by strategy 1 since the latter has larger payoffs regardless of what player 2 does. The table will be as follows:

Strategy	S_1	S_2	S_3
S_1	1	2	4
S_2	1	0	5

Since player two is rational, he will know that player 1 has only the two strategies 1 and 2. For player 2, strategy 3 is dominated by both strategies 1 and 2 since the loss by these two strategies are less. The table will be as follows:

Strategy	S_1	S_2
S_1	1	2
S_2	1	0

Now, player 1 deduce that strategy 2 is dominated by 1 and therefore the table will be as follows:

Strategy	S_1	S_2
S_1	1	2

Strategy 2 of player 2 is dominated by 1, therefore strategy 2 must be eliminated. Finally, the two players will use strategy 1 and the payoff for player 1 is one. This payoff is referred

to as the value of the game. A game with 0 value is called $fair$ game. The concept of dominated strategy is useful on reducing the size of the table and therefore identify the optimal solution for the game.

Non-Zero-Sum Non-Cooperative Game with Dominant Strategy

A game is non-zero-sum, if players interests are not always in direct conflict, so that there is an opportunity for both to gain. A well known example of this type of games is Prisoners' dilemma. In this game, a policeman arrested two persons with incomplete evidence about their crime. To make the arrested persons confess about their crime, he puts them in two separated prisons. Then, he formulated the game in such a way that both players will confess. The game is formulated in the following table:

Player 1			
Strategy		Confess (Defect)	Do not Confess (Cooperate)
Player 2	Confess (Defect)	**-5, -5**	0, -10
	Do not Confess (Cooperate)	-10, 0	-1, -1

In the above table, if player one and two confess then both are sentenced to 5 years and 1 year otherwise. On the other hand, if player one confesses while player two did not then player one (who confessed) will be released and player two will be sentenced to 10 years and vise versa otherwise. Applying dominant strategy to this game will lead the two players to confess since it is the dominant strategy against each other.

The other form of this game which has been used in routing in MANET to enforce co-operation [61] is stated in the above table with "cooperate" and "defect" as the strategies of the players. It is clear from the solution of the game that "defect" is the optimal strategy of

each player, which contradicts the characteristics of MANET. Since every player can record the move of his opponent, a strategy can be predefined and the game can be played iteratively, which is known as *repeated game*. If Prisoners' dilemma game is played iteratively then the solution will be changed to "cooperate" as the dominant strategy which MANET requires [66]. Tit-for-tat is one of the strategies that can be used in repeated games where a player plays "cooperate" as first move, then he/she plays as his/her opponent played in the previous iteration. In [61], the authors designed their own strategy where the players (nodes) cooperate on the first round of the game then cooperate according to opponent's reputation. The objective of this strategy is to make cooperation as the dominant strategy among all players. The authors design the strategy in this way to make the strategy more generous than Tit-for-tat since player moves will not depend on an opponent's previous move but it depends on the average of the previous B moves. This is because in MANET some nodes might not cooperate due to un-control problems; such as, channel collision which prevents nodes from relaying others' traffic.

Nash Equilibrium

Now, let us try to solve the following non-zero sum game using dominant strategy.

		Player 2		
Strategy		S_1	S_2	S_3
Player 1	S_1	0, 4	4, 0	5, 3
	S_2	4, 0	0, 4	5, 3
	S_3	3, 5	3, 5	6, 6

After checking the strategy of each player, we find that this game cannot be solved using dominant strategy. Thus, a new solution is proposed by John Nash [67] to find the solution of such a game which is known as Nash equilibrium. A Nash equilibrium in pure strategies is a pair (r^*, c^*); such that $u_1(r^*, c^*) \geq u_1(r, c^*)$ for all row strategies r of player 1, and $u_2(r^*, c^*) = u_2(r^*, c)$ for all column strategies c of player 2. This means that neither player can take advantage of the other player's strategy to improve his own benefits. Thus, selecting a strategy other than r^* by player 1 will never make him happier than selecting r^* against his opponent which already selected c^*.

Now, let us solve the above game using Nash equilibrium. For each player, and for each strategy for that player, we have to determine the other player's best response to that strategy. If player 2, were to play strategy S_1. Then, player 1 best response would be S_2 since 4 exceeds 3 and 0. Therefore, payoff 4 will be underlined as shown in the following table. If player 2, were to play strategy S_2. Then, player 1 best response would be S_1 since 4 exceeds 3 and 0. Thus, payoff 4 will be underlined. Now, if player 2, were to play strategy S_3. Then, player 1 best response would be S_3 since 6 exceeds 5. Therefore, payoff 6 will be underlined. Similarly, player 2 will do in order to find his best response with respect to player 1 strategies which will be S_1, S_2 and S_3 respectively. The respective payoffs are over-lined as shown in the following table. The entry which is the intersection between the strategies (row and column) is called a saddle point or Nash equilibrium point. Thus, players 1 and 2 best strategy would be S_3. Finally, we note that a pair of strategies satisfies Nash Equilibrium if each player's strategy is the best response to other's strategy.

Player 2				
Strategy		S_1	S_2	S_3
Player 1	S_1	0, $\overline{4}$	$\underline{4}$, 0	5, 3
	S_2	$\underline{4}$, 0	0, $\overline{4}$	5, 3
	S_3	3, 5	3, 5	$\underline{6}$, $\overline{6}$

Minimax strategy

Another way to find the Nash equilibrium for the following game is to apply the minimax or minmax concept which is illustrated through the following example.

Player 2				
Strategy		S_1	S_2	S_3
Player 1	S_1	-3, 3	-2, 2	6, -6
	S_2	2, -2	0, 0	2, -2
	S_3	5, -5	-2, 2	-4, 4

It is clear that dominant strategy does not work here so we have to use different technique which is each player should play in a way to minimize his maximum losses whenever the resulting selection of strategy cannot be exploited by the other player to then improve his selection. This concept is known as *Minimax* [8] strategy selection. Minimax says that player 1 selects a strategy that would be the best even if the chosen strategy was known by the opponent. According to the above table, player 1 should select the strategy whose minimum payoff is the largest which is strategy S_2. On the other hand, player 2 should select the strategy whose maximum payoff to player 1 is the smallest. Therefore, strategy 2 is the maxmin strategy for player 1 and strategy 2 is the minmax strategy for player 2. The

result of the game is zero and therefore the game is a fair game. This entry which is the intersection between the minimum and maximum is the Nash equilibrium, which is neither 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 2 then choosing a strategy other than strategy 2 will lead to a loss for player 2.

We have to note that some games might not have a Nash equilibrium using pure strategies. In this case, game theory advises each player to assign probability distribution over his set of strategies which is known as mixed strategy.

Mixed Strategy

Whenever a game is not solved using pure strategy, mixed strategy is used. This concept was introduced by von Neumann and Morgenstern in their book [92]. However, they only considered the case of zero-sum game where they showed that for any zero-sum game with finite set of strategies there is a mixed-strategy Nash equilibrium. This result was later improved by Nash in [67] to cover all types of games with finite number of players and strategies. He proved that there is at least one Nash equilibrium (mixed-strategy) in any game with finite number of players and strategies.

Mixed strategy can be defined mathematically as follows:

- x_i= is the probability of strategy i by player 1, where $i = 1, 2, ..., m$ and m is the number of available strategies.

- y_j= is the probability of strategy j by player 2, where $i = 1, 2, ..., n$ and n is the number of available strategies.

This probability distribution of the pure strategies is called mixed strategies. Expected payoff theorem could be used for evaluating mixed strategies. The expected value for player 1 is equal to $\sum_{i=1}^{m} \sum_{j=1}^{n} u_{ij} x_i y_j$, where u_{ij} is the payoff.

Game theory extended the concept of minmax to games with mixed strategies.

Minmax theorem states : If mixed strategies are used, the pair of mixed strategies that is optimal with respect to minmax provides Nash equilibrium with minmax equals maxmin equals the value of the game, so that neither player can take advantage by changing his strategy.

To make the idea of mixed strategy clear, let us consider the following game.

		Player 2	
Strategy		S_1	S_2
Player 1	S_1	2, -2	-3, 3
	S_2	-3, 3	4, -4

After applying all the above concepts, it is clear that non of the above concepts is able to solve this game. Thus, mixed strategies are used by player 2 through assigning a probability x_1 to strategy S_1 and x_2 to S_2 with $x_2 = 1 - x_1$. On the other hand, player 1's mixed strategies will be y_1 and y_2 with $y_2 = 1 - y_1$. Then, the expected payoff (U) of player 1 will be equal to the following:

$$2y_1x_1 + (-3)y_1(1 - x_1) + (-3)(1 - y_1)x_1 + 4(1 - y_1)(1 - x_1) = 12y_1x_1 - 7y_1 - 7x_1 + 4$$

Player 2 objective is to minimize the expected payoff of player 1. Therefore, player 2

calculates the first derivative to find the values of x_1 and x_2 respectively which is: $\frac{\delta(U)}{\delta(y_1)} = 12x_1 - 7 = 0 \rightarrow x_1 = 7/12$ and $x_2 = 5/12$. Following the same concept, player 1 can compute his mixed strategy y_1 and y_2.

Note that finding the optimal mixed strategy for more complex cases will be done through transforming the problem to a linear programming problem. Simplex method or any other appropriate method can be used to solve such a problem [8].

Defender/Attacker Game Example

Now, let us see how game theory can be used for intrusion detection. We will play a game between IDS and intruder where the intruder strategy is selected from his strategy space S={Attack, Not Attack}. On the other hand, the IDS (defender) strategy is selected from the strategy space S={Monitor, Not Monitor}. As an example, we will consider the following payoff table where we will solve the game using mixed strategy to find the Nash equilibrium. Knowing that there is no Nash equilibrium using pure strategy for such a game.

IDS			
Strategy		Monitor (x_1)	Not Monitor (x_2)
Attacker	Attack (y_1)	$A_L = -4,\ D_G = 4$	$A_G = 5,\ D_L = -5$
	Not Attack (y_2)	$0,\ D_F = -3$	$0,\ 0$

We define A_L as the loss of the attacker once it is detected by the IDS. The D_G is the gain of the IDS once the attack is detected. The A_G is the gain of the attacker once it is not detected by the IDS. The D_L is the loss of the IDS once an attack is not detected (i.e., false

negative). Finally, the D_F is the false positive rate by the IDS.

To solve the game, the attacker calculates his expected utility (U_A) which is:

$$U_A = -4x_1y_1 + 5y_1(1 - x_1)$$

The objective of the attacker is to maximize the above function which is done by finding the first derivative with respect to y_1 and equating it to zero: $\frac{\delta(U_A)}{\delta(y_1)} = -4x_1 + 5 - 5x_1 = 0 \Rightarrow x_1 = 5/9$. This means that the defender will monitor with probability $5/9$ and not monitor with probability $4/9$.

On the other hand, the defender calculates his expected utility (U_D) to know the optimal strategy. This is done as follows:

$$U_D = 4x_1y_1 - 5y_1(1 - x_1) - 3x_1(1 - y_1)$$

The defender goal is to maximize the above expected utility by finding the first derivative with respect to x_1 and equating it to zero. This leads y_1 to be equal to $3/12$, which means that the attacker will attack with probability $3/12$ and not attack with probability $9/12$ to avoid being detected by the IDS.

Here, we have to note that both players have complete information about each other. Thus, both are able to solve the game and find the Nash equilibrium. The question that we need to address later on is: What if the identity of the attacker is not known? In other words, players have incomplete information about each other. So, how the game is played? The answer is given in the Bayesian games section.

Bayesian Games

Bayesian games are games with incomplete information about the players. Each player has a type θ which is drawn from the set space Θ, where for example Θ = {Malicious, Selfish, Normal}. Additionally, players have initial beliefs about the type of each player. A belief is defined as a probability distribution over the possible types of a player. Players update their beliefs, according to the Bayes' rule given below, as a play takes place in a game [8].

$$\mu_{t_{k+1}}(\theta_i|s) = \frac{\mu_{t_k}(\theta_i)\ P_{t_k}(s|\theta_i)}{\sum_{\theta_i \in \Theta} \mu_{t_k}(\theta_i)\ P_{t_k}(s|\theta_i)}$$

where $\mu_{t_{k+1}}$ is the posterior belief probability, which means: What is the probability of being of type θ_i given the observed strategy s? Moreover, $\mu_{t_k} \geq 0$ is the prior belief and $P_{t_k}(s_i|\theta_i)$ is the probability that strategy s is observed at this stage of the game given the type θ_i of the player. The belief a player holds about another player's type might change on the basis of the strategies the player has played.

In the defender/attacker game, the defender identity is known by the attacker while the identity of the attacker is not known and is selected from the type set Θ = {Malicious (M), Normal (N)}. If the attacker chooses to select the normal type then the only strategy is not to attack. This leads the defender (IDS) to use the Bayes' rule to find the posterior belief of the opponent. Then the defender calculates his expected utility, which is equal to:

$$U_D = (4x_1y_1 - 5y_1(1 - x_1) - 3x_1(1 - y_1))\mu(\theta = M) + (-3x_1)(1 - \mu(\theta = M))$$

To maximize the above function, the defender will find the first derivative with respect

to x_1 and equate it to zero. The result leads that the attacker will attack with probability $\frac{3}{12\ \mu(M)}$.

On the other hand, the attacker calculates his utility function as shown in the previous section which leads the defender to monitor with probability $5/9$.

Here, we need to note that all the above solution concepts; such as, dominant, Nash and Bayesian can be used to solve games in static and dynamic form. In static games, players select their strategies simultaneously without knowing the strategy selected by other players. Payoff tables are used to represent static games. On the other hand, in dynamic games, players play their strategies over a series of steps or stages. In such games, players are able to capture the move of other players and can learn information about the preferences. Such games are usually represented by a tree structure. Note that tree structure games can be interpreted to payoff tables in order to solve such type of games.

3.1.4 Cooperative Game Theory

Cooperative games [8] are games with multi-players cooperating with each other in order to achieve the greatest possible total benefits. Cooperative games consist of: (1) a set of players $N = \{1, 2, \ldots, n\}$ and (2) a characteristic function $v(S)$ specifying the value produced by different group of players S. As an example, let us consider a game with 3 players $N = \{1, 2, 3\}$ where player 1 is a seller for one item where the cost of the item is \$5. Players 2 and 3 are two potential buyers. Player 2 is willing to buy the item with \$10 while player 3 is willing to buy the item with \$12. Therefore, the characteristic functions v of the game are as follows: $v(\{1\}) = v(\{2\}) = v(\{3\}) = 0, v(\{1, 2\}) = 10 - 5 = 5, v(\{1, 3\}) =$

$12 - 5 = 7$, $v(\{2,3\}) = 0$, and $v(\{1,2,3\}) = 12 - 5 = 7$ (because player 1 has only one item and it will be sold to the player with highest value). After calculating the characteristic functions of the different groups, we have to investigate the marginal contribution of each player in the formulated groups. Shapley [29] has studied the contribution of each player in a coalition. His study is known as Shapley value [24]. Mathematically, the Shapley value is defined as follows:

$$\phi_i(v) = \sum_{S \subseteq N, i \in S} \frac{((|S|-1)!(n-|S|)!)}{n!}(v(S) - v(S - \{i\}))$$

This formula means the following: Assume there is a coalition with n players arriving one after the other in a conference room. Consider all the permutations of the n players. Now, if player i arrives in the room and finds the coalition $S - \{i\}$ already there, his contribution to the coalition is $v(S) - v(S - \{i\})$. Therefore, the Shapley value is the expected value of the contribution of the player i.

3.2 Mechanism Design

Mechanism design is used to implement an optimal system-wide solution for a distributed optimization problem with self-interested players where the input of the mechanism is players' private information. Mechanism design allows a game designer to define the rules of the game in terms of the social choice function such that players will play according to these rules. This requires that the equilibrium point leads to the social choice function. Thus, games should be designed in such a way players' dominant strategy is the one that

leads to the objective function (SCF). In the following, we present how mechanisms are designed followed by the well known mechanism VCG which is the mechanism we use in this thesis.

3.2.1 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 by Nisan and Ronen [70] for solving least cost path and task scheduling problems using algorithmic mechanism design. Distributed mechanism design based on VCG is first introduced in [31] 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 [79] where the authors considered the consequences that may appear after the computation phase is finished. Currently in MANET, mechanism design is mainly used for routing purposes, such as a truthful adhoc-VCG mechanism for finding the most cost-efficient route in the presence of selfish nodes [5]. In [20], 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.

3.2.2 How to Design a Mechanism?

A standard mechanism design model is defined by a set of n agents where each agent $i \in \{1, \ldots, n\}$ has a private information about his preferences, $\theta_i \in \Theta_i$, known as the agent's type. Moreover, it defines a set of strategies S_i for each agent i. The agent can choose any strategy $s_i \in S_i$ to input to the mechanism based on agent's type which is represented by $s_i(\theta_i)$. We refer to $s_i(\theta_i)$ as s_i. According to the inputs $(s_i, \ldots, s_n)$ of all the agents, the mechanism calculates an output $o = o(a_1, \ldots, a_n) \in O$ where O is the set of all possible outputs. For simplicity, we refer to $o = o(a_1, \ldots, a_n)$ as o(.). The objective of mechanism design is to define rules that make the agents select their corresponding strategy, which leads to the social choice function $f(.)$. The question now is how to make the output $o(.)$ equals to the social choice function $f(.)$? This what we shall address in this section.

We say that a mechanism M implements the social choice function $f(.)$, if the outcome of the mechanism at equilibrium is the solution of the social choice function and thus, $f(.) = o(.)$. To achieve this goal, game theory tools are needed to analyze the outcome of a mechanism in order to find the equilibrium. In other words, a mechanism, $M = (S_1, \ldots, S_n, o(.))$ implements a social choice function, $f(.)$ if $o((S_1^*(\theta_1), \ldots, S_n^*(\theta_n)) = f(.)$, where the strategy profile, $s^* = (s_1^*, \ldots, s_n^*)$ is the equilibrium of a game.

Here, the equilibrium strategy can be based on any equilibrium concept, such as dominant, Nash, Bayesian, etc. Generally, dominant strategy is preferred over other techniques since it has less assumption than other concepts regarding players information. Mechanisms implemented using dominant strategy are known as strategy-proof. Vickrey-Clarke-Groves (VCG) is the only strategy-proof mechanism which is given in Section 3.2.3.

Quasi-Linear Utility Function

The utility function $u_i(s_i(\theta_i), o)$ is used to calculate agents preference over different outcome. In mechanism design, it is assumed that this function is defined using the quasi-linear utility function. The general form of the quasi-linear function for an agent i with type θ_i and strategy s_i is:

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

where outcome o defines a choice x, which is an element in the discrete choice set X and p_i is the payment of an agent. The preference of each player from the output is calculated by a valuation function, $v_i(x)$, which is a quantification to evaluate the value of x in real number. Thus, the utility of a node is calculated as a combination of output measured by the valuation function and payment defined by the mechanism. Agent's utility function is maximized only when agent's receives x with less payment. Thus, agents can reveal fake information about their valuation function v in order to have the choice x with less payment.

As an example, we consider an auction of a single item. The outcome of the auction which is, in our case, the SCF is to determine the best winner that can valuate item x the most among all the agents. If the value of the item by the winner is \$15 then the agent has a positive utility as long as the payment is less than \$15.

The question is: How to design the payments in a way that makes agents reveal their truthful valuation function?

Incentive Compatible Mechanism

Incentives must be given to motivate agents on revealing truthfully their private information. This leads to a social choice function. Let us consider an example about incentives where we have two children squabbling over a piece of cake. So, we need a mechanism to divide the piece of cake fairly. The best solution that usually the parents follow is to ask one children to cut the piece and the second to choose. This mechanism will motivate the first child to cut the piece equally as possible since the second child will choose the larger half. This mechanism, first cut and second choose, is a simple example of incentive compatible mechanism. Recently in 2007, L. Hurwicz, E. Maskin, and R. Myerson received the Nobel prize in economics for their study of incentive compatible mechanisms [44]. Another example for incentive compatible mechanism is the second-price sealed bid auction for allocating a single item which is created by Vickrey. Vickrey has awarded in 1996 a Nobel prize for his work in mechanism design [63]. The general case of Vickrey is given in Section 3.2.3.

A mechanism is called as dominant-strategy incentive compatible or strategy-proof if the dominant strategy of each agent is to report his private information truthfully. Note that VCG is the only well known strategy-proof mechanism.

Revelation Principle

The Revelation Principle is a way that simplifies the analysis of mechanism design problems. Direct revelation mechanism is a mechanism that the only actions available to agents

are to reveal directly their preferences to the mechanism. If agents report their truthful information about their preferences in equilibrium then an incentive compatible mechanism is a direct-revelation mechanism.

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

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

3.2.3 Vickrey-Clarke-Groves (VCG) Mechanism

Vickrey-Clarke-Groves (VCG) is a strategy-proof mechanism that has been proposed by Vickrey [91], Clarke [23] and Groves [33]. VCG is the only allocation-efficient and strategy-proof mechanism amongst all direct revelation mechanisms. It is used to solve problems with a set of possible alternatives, X, and players have quasi-linear preferences.

In VCG mechanism, each player $i \in N$ calculates his valuation function v for alternative $x \in X$ given his type θ. Based on the collected valuations, the goal of the mechanism is to find the outcome $o = (x^*, p)$ where p is the set of payments that will be given to/by the players. Thus, the goal is to maximize the total valuation over all the players which is as follows:

$$SCF = x^* = \max \sum_i v_i(x, \theta_i)$$

The payment in VCG takes the following form:

$$p_i = v_i(x^*, \theta_i) - \{V_N - V_{N-i}\}$$

where $V_N = \max_{x \in X} \sum_i v_i(x, \theta_i)$ is the total reported value of x^*. On the other hand, $V_{N-i} = \max_{x \in X} \sum_{j \neq i} v_j(x, \theta_j)$ is the best total reported value of the choice without considering i. Note that the payment of a player is not influenced by its reported valuation.

VCG Example

Let us consider the shortest path problem where the mechanism goal is to find the shortest path between a given source A and destination E as shown in Figure 10.

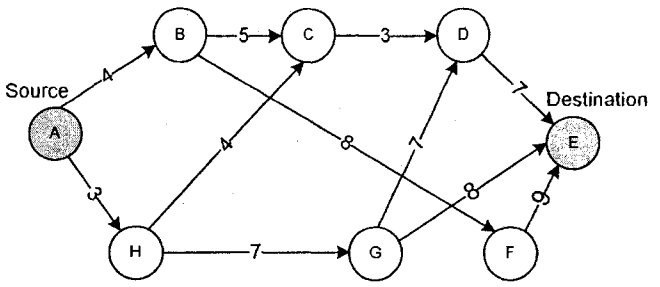

Figure 10: Shortest path example using VCG

We assume that each link has a private cost. If the information is revealed truthfully then the shortest path from A to E is 17 which is path $AHCDE$. The mechanism will output the shortest path $AHCDE$ which is the SCF and the payment of each link according to VCG.

Assuming that the sender will pay for the links, the payment of link CD, for example, is the link's cost 3 − (the total cost of the path which is 17 − the total cost of the shortest path excluding link CD which is 18 since path $AHGE$ is the shortest path). This leads that the payment of link $CD = 4$.

3.3 Summary

From the above discussions, it is clear that game theory and mechanism design are strong candidates for providing the much-needed mathematical framework for analyzing the interaction between IDSs and intruders. Moreover, to motivate the selfish users to participate through revealing their truthful information. To make the thesis self contained, we provided the basics of game theory and mechanism design. We showed how games are defined and how they are solved through illustrative examples. We provided a simple example to show how game theory can be used to model a game between intruder and IDS. Then we showed, through an example, how to solve the games with incomplete information about the players. Last but not least, we can conclude that game theory can help us to find the optimal threshold for adding more monitors according to the security needs. Moreover, it can be used to formulate our sampling problem and guide the IDS to find the optimal sampling strategy against its opponent (intruder/intruders).

After illustrating game theory, we explained the basics of mechanism design that we have used through out this thesis. Finally, we can conclude that 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 [26] is to address this problem by designing incentives for players (nodes) to provide truthful information about their preferences over different outcomes.

Chapter 4

Leader Election Mechanisms

In this chapter, we propose a solution for balancing the resource consumption of IDSs among all nodes while preventing nodes from behaving selfishly. To address the selfishness behavior, we design incentives in the form of reputation to encourage nodes to honestly participate in the election scheme by revealing their cost of analysis. The cost of analysis is designed to protect nodes' sensitive information (resources level) and ensure the contribution of every node on the election process (fairness). To motivate nodes in behaving normally in every election round, we relate the amount of detection service that each node is entitled to the nodes' reputation value. Besides, this reputation value can also be used to give routing priority and to build a trust environment. The design of incentives is based on a classical mechanism design model, namely, Vickrey, Clarke, and Groves (VCG) [59]. The model guarantees that truth-telling is always the dominant strategy for every node during each election phase. Last but not least, we address these issues in two possible settings,

namely, Cluster Independent Leader Election (CILE) and Cluster Dependent Leader Election (CDLE). In the former, the leaders are elected according to the received votes from the neighbor nodes. The latter scheme elects leaders after the network is formulated into multiple clusters. In both schemes, the leaders are elected in an optimal way in the sense that the resource consumption for serving as IDSs will be balanced among all nodes overtime.

To catch and punish the misbehaving leader after election, we propose a catch and punish mechanism where a set of checkers are elected randomly to monitor the behavior of the leader. To reduce the resource consumption by the checkers we assume that checkers are mirroring a portion of the work handled by the leader to verify its behavior. Checkers are added gradually according to detection threshold. To reduce the false positive by the checkers, a cooperative decision model based on cooperative game theory is proposed. Finally, our proposed model is a hybrid approach that combines the advantages of random selection and mechanism design-based selection.

The rest of this chapter is organized as follows: Section 4.1 formulates the problem. Section 4.2 describes our leader election mechanism where the cost of analysis function, reputation model and payment design are given. Section 4.3 analyzes our mechanisms against selfish and malicious nodes. Section 4.4 describes the catch and punish mechanism where checkers are added gradually according to the behavior of the leader. To reduce the false positive of the checkers, we analyze the contribution of the checkers on the final decision. This is done using cooperative game theory.

4.1 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 assume that nodes belong to different users/organizations and thus there will be no collusion among the nodes to disrupt our leader election. 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 (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):

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

Clearly, in order to minimize this SCF, the following must be achieved. First, we need to design incentives for encouraging each node in revealing its true cost of analysis value c, which will be addressed in 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.

4.2 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 (reputation). If the player's strategy is normal then the node reveals the true cost of analysis. In Section 4.3 a detailed analysis is given. We assume that each player i has a utility function [59]:

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

where, θ_{-i} is the type of all the other nodes except i. v_i is the valuation of player i of the output $\mathbf{o} \in O$, knowing that O is the set of possible outcomes. In our case, if the node

is elected then v_i is the cost of analysis c_i. Otherwise v_i is 0 since the node will not be the leader and hence there will be no cost to run the IDS. $p_i \in \Re$ is the payment given by the mechanism to the elected node. Payment is given in the form of reputation. Nodes that are not elected receive no payment.

Note that, $u_i(\theta_i)$ is what the player usually seeks to maximize. It reflects the amount of benefits gained by player i if he follows a specific type θ_i. Players might deviate from revealing the truthful valuation for the cost of analysis if that could lead to a better payoff. Therefore, our mechanism must be strategy-proof where truth-telling is the dominant strategy. To play the game, every node declares its corresponding cost of analysis where the cost vector C is the input of our mechanism. For each input vector, the mechanism calculates its corresponding output $\mathbf{o} = o(\theta_1, \ldots, \theta_n)$ and a payment vector $\mathbf{p} = (p_1, \ldots, p_n)$. Payments are used to motivate players to behave in accordance with the mechanism goals. In the following subsections, we will formulate the following components:

1. Cost of analysis function: It is needed by the nodes to compute the valuation function.
2. Reputation system: It is needed to show how:
   (a) Incentives are used once they are granted.
   (b) Misbehaving nodes are catched and punished.
3. Payment design: It is needed to design the amount of incentives that will be given to the nodes based on VCG.

4.2.1 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 EL_i, and the number of expected alive slots is denoted by NT_i. Based on these requirements, each node i has a power factor $PF_i = EL_i/NT_i$. We introduce the set of $l-1$ thresholds $P = \{\rho_1, \ldots, \rho_{l-1}\}$ to categorize the classes as follows:

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

The reputation of node i is denoted by R_i. At the first round of the game the reputation is initialized to one. 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 (Joule/packet) and EL_{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** $(EL_i < EL_{ids})$ **then**
2. $\quad c_i = \infty$
3. **else**
4. $\quad c_i = \frac{PS_i}{PF_i} = \frac{\frac{R_i}{\sum_{i=1}^{N} R_i} \times NT_i}{EL_i}$
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 this thesis, we will use cost and cost-of-analysis interchangeably.

Table 1: 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%

We show the effect of our cost function over PS through an example. Table 1 shows the PS for 20 nodes divided equally over 4 energy classes where nodes in class 4 have the most resources. Table 1 indicates that initially nodes belonging to lower energy level have a small budget. As the time goes by, the nodes belonging to lower energy class gains more budget while the budget of higher classes decreases. This justifies that our cost function is able to balance the energy of the nodes and gives a fair budget to all nodes.

4.2.2 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 11 shows the abstract model of our reputation system where each node has the following components:

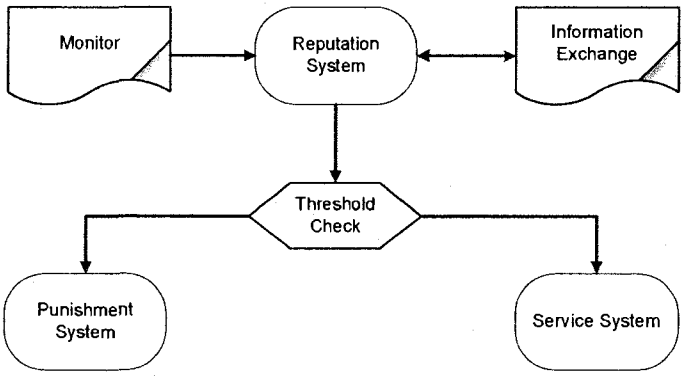

Figure 11: Reputation system model

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. The catch and punish model is given in Section 4.4.

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 a source node has an unacceptable low reputation then its packet will have less priority. Hence, in every round, nodes will try to increase their reputation by becoming a 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 formulate in Section 4.4 a cooperative game-theoretical model to efficiently catch and punish misbehaving leaders with low false positive rate. Our catch-and-punish model is 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.

4.2.3 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:*

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

$$\rho_k = c_k + \frac{1}{\sum_{i \in N} vt_k(C, i)} \times [\sum_{j \in N} c_j \sum_{i \in N} vt_j(C|c_k = \infty, i) - \sum_{j \in N} c_j \sum_{i \in N} vt_j(C, i)] \tag{5}$$

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 [70]. Here valuation refers to the total cost incurred by a node. According to [46], 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,

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

Which is reduced to

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

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

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

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. □

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 Chapter 5.

4.2.4 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. Here, we use the scheme of [52] to cluster the nodes into one-hop clusters. Each cluster then independently elects a leader among all the nodes to handle the monitoring process based on nodes' analysis-cost. Our objective is to find the most cost-efficient set of leaders that handle the detection process for the whole network. Hence, our social choice function is still as in Equation 1.

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

$$P_k = \sum_{i \in N} vt_k(C, i) B \rho_k, \text{ where} \tag{14}$$

$$\rho_k = \min \sum_{j \in -n_k} c_j(\theta_j, o(\theta_j, \theta_{-j})) \tag{15}$$

Proof: According to the standard notation in mechanism design [70], the second best price is the simplest form of VCG mechanism. Here, $\sum_{j \in -n_k} c_j(\theta_j, o(\theta_j, \theta_{-j}))$ denotes the best cost excluding n_k. This is because nodes in the cluster have to select one node from the same cluster to be a leader. Unlike CILE where nodes can vote to its one-hop neighbor and then clusters are formed. □

4.3 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.

4.3.1 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.

4.3.2 Presence of Malicious Nodes

A malicious node can disrupt our election algorithm as follows:

- First, 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 propose in Section 4.4 a decentralized catch-and-punish mechanism using random *checker* nodes to monitor the behavior of the leader. 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 consequently excluded from future services of the cluster.

- Second, by the malicious use of the voting function. Since malicious nodes are rational and thus they need to attack without being catched, they have no interest to vote to other nodes to be elected as leaders. This is because the number of monitors will increase and thus the probability to be catched by the leaders will increase.

4.4 Catch and Punish Model

Due to un-control problems such as channel collision, the leader-IDS could not be able to monitor and analyze the traffic of some protected nodes for a specific period of time. Hence, a checker that is monitoring the behavior of the leader-IDS could report a misbehaving event and therefore the leader-IDS is punished and a new leader is elected. This problem

motivated us to propose a cooperative game theoretical model that is able to efficiently catch and punish a misbehaving leader-IDS with less false positive rate. We propose the concept of detection-levels to be $DL = \{dl_1, \ldots, dl_k\}$ which enables us to respond better to misbehaving leader-IDS depending on which detection-level it belongs to. Hence, the percentage of checkers will vary with respect to the detection-level. Our catch and punish model is made up of k detection-levels, each level represents the severe behavior of the leader-IDS. We introduce the set of $k-1$ thresholds T to categorize the detection-levels where $T = \{t_1, \ldots, t_{k-1}\}$. Now, we introduce the aggregate function to be:

$$F(n) = \sum_{i \in n} R_i \times f(i) \tag{16}$$

where R_i is the reputation of checker i, n is the set of checkers and $f(i)$ is the catch function that takes a value between 0 and 5 according to the severe behavior of the leader-IDS. To decentralize the catch decision, *F(n)* will be calculated by all the checkers after the secure exchange of $f(i)$. $F(n)$ sums up the catch-function of each checker i, in a cluster n, while considering the reputation of each checker. Now, we categorize the detection-levels as follows:

$$DL = \begin{cases} dl_1 & if\ F(n) < t_1 \\ dl_i & if\ t_{i-1} \leq F(n) < t_i;\ i \in [2, k-1] \\ dl_k & if\ F(n) \geq t_{k-1} \end{cases} \tag{17}$$

Categorizing the misbehavior of leader-IDS into different levels help in catching the

misbehaving leader-IDS with less false positive rate and reduce the performance overhead of the checkers. We can use non-cooperative game theory as in Chapter 5 in Section 5.5 to calculate the values of thresholds T and detection-levels DL in order to have better results with respect to catch and punish. Here, cooperative game theory is used to formally illustrate the problem by analyzing the contribution of each checker on the decision.

4.4.1 Analyzing the Contribution of Checkers

The design and analysis of our proposed model is done using cooperative game theory [29]. The l checkers will be modeled as a set N of l players in *N-person* game with $N = \{N_1, \ldots, N_l\}$ [8]. We introduce a coalition in cooperative game theory to be:

$\Delta \subseteq N \; and \; \forall x \in \Delta$.

In other words, we define a coalition to be a set of checkers, where each checker reports the behavior of elected leader-IDS. Therefore, each checker in Δ reports a risk in the cluster. Let δ be the number of checkers in a coalition in a cluster. We use the aggregate function over Δ,

$$\sum_{x \in \Delta} R_x \times f(x)$$

to assign the behavior of leader-IDS to its equivalent detection-level dl_j.

Assigning an anticipated marginal contribution to each player (checker) in the game with respect to a uniform distribution over the set of all permutations on the set of players will be presented by Shapley value [29]. To find the contribution of checker N_i in coalition Δ, we consider all the different permutations for the checkers, Π_Δ, in the coalition. Then

we calculate the difference between the function including all checkers in the permutation before checker N_i, including N_i, and the function of all checkers prior to N_i, excluding N_i. We define $P_\pi^{N_i}$ to be the set of checkers before the node N_i in the permutation $\pi \in \Pi_\Delta$. Then taking the average of all these differences, we get the marginal contribution of checker N_i in coalition Δ. In other words, the contribution would be the following:

$$\phi_{N_i}(\Delta) = \frac{1}{\delta!} \sum_{\pi \in \Pi_\Delta} V(P_\pi^{N_i} \cup \{N_i\}) - V(P_\pi^{N_i}) \tag{18}$$

We define the value function for the coalition game as follows:

$$V(N) = \begin{cases} 1 & if\ F(N) \geq t_i \\ 0 & Otherwise \end{cases}$$

The Shapley value of checker N_i indicates the relative contribution for a given threshold t_i. Therefore, we can tune the values of the thresholds either by using statistical data in order to reduce the false positives or by following the same concept in Section 5.5. In each detection-level, we have a different response varies from adding more checkers to dropping the leader-IDS and electing new one. This illustrates the importance of analyzing the relative contribution of a checker to decide the detection-level. The following example illustrates our model.

Illustrative Example

Consider three checkers in a cluster of 10 nodes. A checker noticed that the elected leader-IDS is not doing the job of detection. Note that the leader could misbehave either due to a channel collision or due to selfish/malicious behavior. To efficiently catch and punish the misbehavior leader-IDS and reduce false positive, a cooperative decision is made among the checkers in the cluster.

As an example, let us consider that we have four detection-levels $DL = \{dl_1, dl_2, dl_3, dl_4\}$. Knowing that dl_1 indicates normal behavior, dl_2 and dl_3 mean more checkers have to be added to monitor the behavior of node while dl_4 means a new leader-IDS has to be elected. The threshold set $T = \{2, 4, 6\}$, the detection-levels DL are classified as follows: $dl_1 < 2$, $2 \leq dl_2 < 4$, $4 \leq dl_3 < 6$, $dl_4 \geq 6$, the reputation of nodes $R_1 = 0.5$, $R_2 = 0.8$, $R_3 = 0.2$,, and $f(1) = 3$, $f(2) = 4$, $f(3) = 3$. Using equation 18, we calculate the marginal contribution of each checker on detection level dl_3, which is: $\phi_{N_1} = 0.5$, $\phi_{N_2} = 0.5$ and $\phi_{N_3} = 0$. This means that more checkers have to be added to monitor the behavior of the leader where node N_3 contribution has no effect on the cooperative decision at dl_3 since its reputation is low. This will avoid the malicious use of the punishment system.

4.5 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. Last but not least, we analyzed our mechanism against selfish and malicious nodes. Finally, we presented the catch and punish model based on cooperative game theory where the contribution of each checker on the final decision is derived. This model reduced the false positive and avoid the malicious use of such a model. In the following chapter, we present the leader election algorithms to implement our mechanism. Moreover, we express our moderate to robust model followed by simulation results.

Chapter 5

Leader Election Algorithms and Robust IDS Model

To use 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. 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. Additionally, the performance overhead is considered during the design of the given algorithm where computation, communication and storage overhead are derived.

Relying on leaders for providing IDS service by examining a portion of all nodes' packets is suitable whenever the probability of attack is low. This is usually known as a *moderate* intrusion detection model. However, a *robust* model where the victim nodes launch their own IDSs will be more desirable when the threat level is high. Running the

moderate model in a non-secure environment raises the need for more nodes to launch their IDS according to attack's severity. Thus, more nodes should launch their own IDS according to node's security risk. This will help to prolong the lifetime of nodes and increase nodes' security. The question that we address here is: *What is the optimal threshold value needed to inform the victim node to launch its own IDS in order to minimize both resource consumption and security risk?* We tackle this important issue in this chapter.

To achieve this goal, we formalize the tradeoff between security and IDS resource consumption as a non-cooperative game between leader-IDS and attacker with incomplete information about the attacker. This game guides the leader-IDS and intruder to derive their optimal strategy against each other. For the leader-IDS, the game solution derives the threshold for informing the victim node to launch its own IDS once the probability of attack exceeds the derived threshold. The game will be repeated such that in every election round the leader-IDS will be monitoring via sampling the protected node's incoming traffic and deciding according to the game solution whether to inform the victim node to launch its IDS or not. On the other hand, the attacker's strategy will be to attack once the probability of stepping into the robust mode (that is, the victim node will be running its own IDS) is low.

Finally, we evaluate the performance of our election model with respect to random and connectivity models. We simulate the schemes using Network Simulator 2 ($NS2$) [60] and MATLAB. Moreover, empirical results indicate that our moderate to robust scheme can effectively reduce the resource consumption of IDSs without sacrificing security.

This chapter is organized as follows: Section 5.1 presents the objectives and assumptions. Section 5.2 devises the election algorithm needed to handle the election process taking into consideration the addition and removal of nodes. Section 5.3 provides the performance analysis of our algorithm. Section 5.4 illustrates an informal analysis of the correctness and security properties of the algorithm. Section 5.5 provides the moderate to robust game model with an illustrative example. Section 5.6 presents the empirical results. Finally, Section 5.7 summarizes the chapter.

5.1 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 [47, 99].
- 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 [73] 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 [34].

5.2 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. Note that the payment is calculated according to network settings (i.e., CILE or CDLE). 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. Initially, the reputation of all the nodes is initialized to one.
- *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.

5.2.1 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 5.4.2.

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**($neighbors(k)$=Ø) **then**
4. Launch IDS.
5. **end if**

On expiration of T_1, each node k checks whether it has received all the hash values from its neighbors. Nodes from whom the *Hello* message have not received are excluded from the election. On receiving the *Hello* from all neighbors, each node sends *Begin-Election* as in Algorithm 1, which contains the cost of analysis of the node and then starts timer T_2. If node k is the only node in the network or it does not have any neighbors then it launches its own IDS and terminates the algorithm.

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

Algorithm 2 (Executed by every node)

/* Each node votes for one node among the neighbors */
1. **if** ($\forall\ n \in neighbor(k),\ \exists\ i \in n : c_i \leq c_n$) **then**
2. send $Vote(ID_k,\ ID_i, cost_{j \neq i})$;
3. *leadernode(k)*:= i;
5. **end if**

the least-cost value among its neighbors and sends *Vote* for node i as in Algorithm 2. The *Vote* message contains the ID_k of the source node, the ID_i of the proposed leader and second least cost among the neighbors of the source node $cost_{j \neq i}$. Then node k sets node i as its leader in order to update later on its reputation. Note that the second least cost of analysis is needed by the leader node to calculate the payment. If node k has the least cost among all its neighbors then it votes for itself and starts timer T_3.

Algorithm 3 (Executed by Elected leader node)

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

On expiration of T_3, the elected node i calculates its payment using equation 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 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 in CILE, Figure 12 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 2.

Table 2: 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

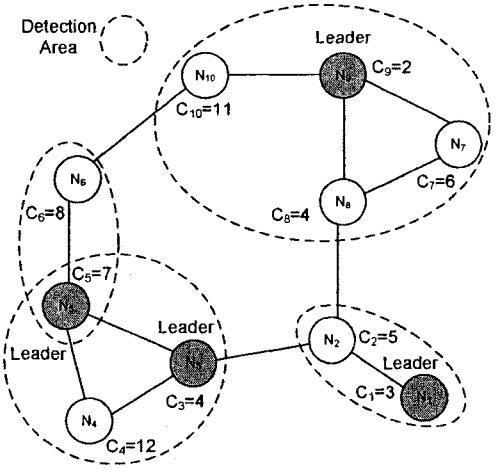

Figure 12: MANET after leader election

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 mechanism based on their type (*Selfish* or *Normal*). The corresponding cost of analysis is given in the second row of Table 2. 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 $P_9 = \sum v_9 B \rho_9 = 4 \times 5 \times 5 = 100$, 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: $S = \{S_7 = \frac{90 \times 20}{470}, S_8 = \frac{160 \times 20}{470}, S_9 = \frac{110 \times 20}{470}, S_{10} = \frac{110 \times 20}{470}\}$. 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 2.

5.2.2 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 $Status(k, n)$;

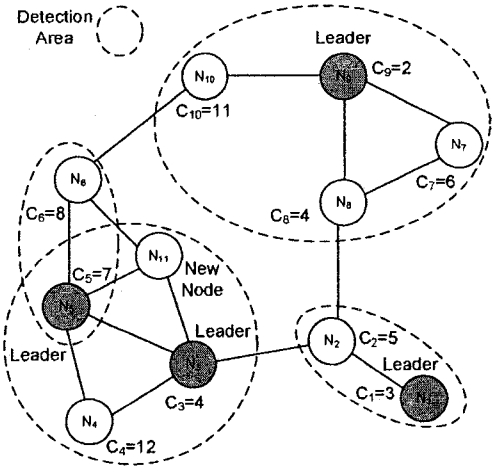

Figure 13: MANET after adding a new node

On receiving the *Status* messages from the neighbors, the new node n sends *Join* to the leader node. If two of its neighbors are leaders with the same cost, then the new node can send *Join* to any of the nodes depending on its physical location (i.e; signal strength).

We assume that an ordinary node have no interest to be a leader during the service time since it will not receive any payment from others. The algorithm does not make the new node as a leader for others before the new election (i.e., to reduce performance overhead). Detailed analysis is presented in Section 5.4. 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 12. The resulting network after adding the new node is shown in Figure 13. The new node 11 is connected with nodes 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.

5.2.3 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 ϵ 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 update its serving list.

Example: Here, we consider the removal either of an ordinary node or a leader node. Considering the network in Figure 12, let us assume that node 7 has left the network or died. Immediately, nodes 8 and 9 will be aware of the failure. On receiving the *Dead(7)* message, nodes 8 and 9 check whether node 7 is their leader or it's being served by them. As node 7 is an ordinary node, node 8 does nothing. In case of node 9, it updates its serving

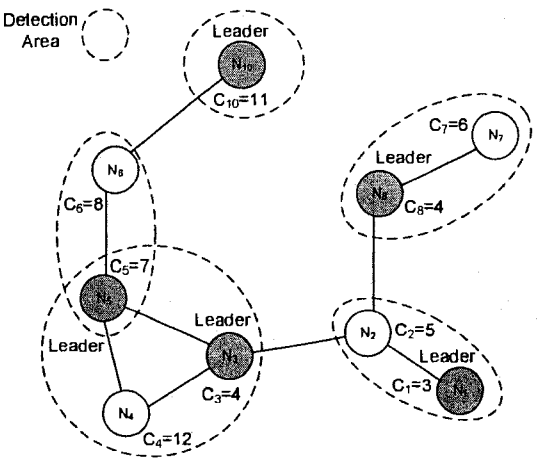

Figure 14: MANET after adjustment

list as in Algorithm 4. Assume now that the links of node 9 have been broken as shown in Figure 14. 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.

5.3 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.

5.3.1 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 [34] to verify a signature in $0.43s$. Since leader election will take place after a certain interval, this computational overhead is tolerable.

5.3.2 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.

5.3.3 Storage Overhead

According to the algorithm, each node maintains a *reputation-table*, *neighbors* list and two variables: *Leadernode* and *leader* (see section 5.2). 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 [52].

5.4 Leader Election Algorithm Analysis

In this section, we provide an analysis of our algorithm where we analyze its characteristics and security properties.

5.4.1 Algorithmic Analysis

Here, we analyze our leader election algorithm to verify whether it satisfies our leader mechanism characteristics or not.

Proposition 1: Our algorithm confirms that each node is monitored by a leader node.

Proof: It is easily noticeable that after executing the election algorithm, each node is assigned a role. According to Algorithm 2, a node is either a leader or ordinary within a finite time. 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 cannot do anything but to send the *Vote* message to the deserving candidate. If a node does not have any neighbor, it becomes the leader node according to Algorithm 1. Besides, if a node loses its connection with the leader due to change in the network topology, it can always get associated with another leader through Algorithms 4 and 5. Thus, in all cases a node is either a leader or ordinary (monitored by a leader node). □

Proposition 2: The overall cost of analysis for protecting the whole network is minimized.

Proof: According to above proposition, 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 do not 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. □

5.4.2 Security Properties Analysis

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.

Proposition 3: The algorithm provides basic security requirements.

Proof: 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 cannot 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 whether the message is replied or not. Thus our algorithm is also safeguarded against reply attack. □

Proposition 4: The algorithm is cheat proof.

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. □

5.5 Moderate to Robust Game Model

Leader election model is considered as a moderate intrusion detection since it only monitors and analyzes a portion of all events occurring in the network. This model can be used whenever the threat of attack is low. It will help to reduce the overall resource consumption of IDSs and communication overhead. Once the probability of attack against a node is high, the victim node should launch its own IDS to detect and thwart intrusions. Therefore, the detection steps into the robust mode. A mechanism is needed to decide when to go from moderate mode to robust mode. To formally address this issue, we formulate a game with two players: Leader-IDS and intruder. The objective of the intruder is to attack a node without being detected, where that of the leader-IDS is to detect such intruders. In order to detect an intrusion, the leader-IDS samples the incoming packets for a target node based on a sampling budget determined through that target node's reputation. Once the probability of attack goes beyond a threshold, the leader-IDS will notify the victim node to launch its own IDS. Our goal is to find the optimal threshold value needed to notify the victim node.

5.5.1 Game Definition

We model the game as nonzero-sum noncooperative game with incomplete information about the players where each player has a private information about his/her preferences. In our case, the leader-IDS type is known to all the players while the sender type is selected

Table 3: Moderate to robust game

Strategy	Robust	Moderate
Attack	$\overline{E_r}V - C_a,\ E_rV - C_r$	$\overline{E_m}V - C_a,\ E_mV - C_m$
Not-Attack	$0, -C_r$	$0, -C_m$

from the type set $\Theta = \{Malicious\ (M),\ Normal\ (N)\}$. Knowing that the sender type is a private information. *Bayesian Equilibrium* [95] dictates that sender's action depends on his/her type θ. By observing the behavior of the sender at time t_k, the leader-IDS can calculate the posterior belief evaluation function $\mu_{t_{k+1}}(\theta_i|a_i)$ using the following Bayes rule:

$$\mu_{t_{k+1}}(\theta_i|a_i) = \frac{\mu_{t_k}(\theta_i)\ P_{t_k}(a_i|\theta_i)}{\sum_{\theta_i \in \Theta} \mu_{t_k}(\theta_i)\ P_{t_k}(a_i|\theta_i)} \tag{19}$$

where $\mu_{t_k}(\theta_i) > 0$ and $P_{t_k}(a_i|\theta_i)$ is the probability that strategy a_i is observed at this stage of the game given the type θ of the node i. It is computed as follows:

$$P_{t_k}(Attack|\theta_i = M) = E_m \times O + F_m(1 - O)$$

$$P_{t_k}(Attack|\theta_i = N) = F_m$$

where O is the observed behavior, which is determined by the IDS monitor. F_m is the false rate generated by the leader-IDS due to sampling and E_m is the expected detection rate via sampling in moderate mode.

We define the intruder's pure strategy as $A_i = \{Attack, Not - Attack\}$. On the other hand, leader-IDS strategy is selected from the strategy space $A_{IDS} = \{Robust, Moderate\}$.

By solving this game using pure strategy, there is no Nash equilibrium. Thus, mixed strategy is used to solve the game where q is the probability to run in robust mode and p is the probability to attack by the attacker. In Table 3, the game is defined where the utility function of the IDS by playing the *Robust* strategy while the attacker plays the *Attack* strategy is defined as $E_r\ V - C_r$. It represents the payoff of protecting the monitored node, which values V, from being compromised by the attacker, where $E_r\ V >> C_r$. On the other hand, the payoff of the attacker if the intrusion is not detected is defined as $\overline{E_r}\ V - C_a$. It is considered as the gain of the attacker for compromising the victim node. Additionally, we define $E_m\ V - C_m$ as the payoff of IDS, if strategy *Moderate* is played while the attacker's strategy remains unchanged. Conversely, the payoff of the attacker if the intrusion is not detected is defined as $\overline{E_m}\ V - C_a$. Now, if the attacker plays *Not-Attack* strategy and the IDS strategy is *Robust* then the losses of the IDS is C_r while the attacker gains/losses nothing. Moreover, the payoff of the attacker with the same strategy and IDS strategy is *Moderate* is 0 while the losses of the IDS is defined as C_m which is the cost of running the IDS in moderate mode. Where,

- $\overline{E_r}$ is the expected probability of not detecting an intrusion in the robust mode while E_r is the expected probability of detecting an intrusion in the robust mode. Note that E_r = Probability $(E_{leader} \bigcup E_{victim})$, where E_{leader} and E_{victim} are the expected probability of detection by leader-IDS and monitored node (victim) respectively. We define the expected probability of detection as E= $\sum_{l \in L} x_l y_l$, where x_l is the probability of detecting an intrusion via sampling and it is equal to s_l/f_l. On the other hand, y_l is the probability of selecting link l by an intruder to attack a victim

node. Knowing that s_l is the sampling rate at victim's incoming link l and f_l is the flow at link l where L is the set of incoming links. We consider the link l as the route link that connect the victim node with other nodes.

- $E_m = E_{leader}$ is the expected detection in the moderate mode, where only the leader-IDS is running the IDS to detect intrusions. On the other hand, $\overline{E_m}$ is the expected of not detecting an intrusion in the moderate mode.

- C_r is the cost of running the IDS in robust mode. We define the cost as the aggregation of the cost of monitoring by the leader C_{leader} and cost of monitoring by the victim C_{victim}.

- C_m is the cost of running the IDS in moderate mode which is equal to C_{leader}.

- C_a is the cost of attack by the intruder.

- V is the value of the protected victim node (asset). The value of V could vary from one node to another according to its role in the cluster. For example, gateway nodes are valued more than regular nodes.

5.5.2 Game solution

To solve the game and find the optimal values of p and q, the IDS and attacker compute their corresponding utility functions followed by the first derivative of the functions. From Table 3, the IDS utility function U_{IDS} is defined as follows:

$$U_{IDS} = [qp(E_r\ V - C_r) + p(1-q)(E_m\ V - C_m) - q(1-p)C_r$$

$$-(1-q)(1-p)C_m]\mu(\theta = M) - [qC_r + (1-q)C_m](1-\mu(\theta = M))$$

The main objective of the IDS is to maximize this utility function by choosing for a fixed p^*, a q^* strategy that maximizes the probability of protecting the victim node and leads to equilibrium where the following holds:

$$U_{IDS}(p^*, q^*) \geq U_{IDS}(p^*, q)$$

To achieve this goal, the IDS will calculate the optimal value of p^* by finding the first derivative with respect to q^* and setting it to zero. This will result to the following: $p^* = \frac{C_r - C_m}{\mu\, V\, (E_r - E_m)}$ and can be reduced to: $\frac{C_{victim}}{\mu\, V\, E_{victim}}$.

The value of p^* is used by the leader-IDS to decide whether to inform the victim node to launch its own IDS or not. Knowing that the leader-IDS is monitoring and analyzing traffic via sampling to detect an intrusion launched by an external attacker i. The IDS is computing the belief μ, as in Equation 19, of each node to check whether it is behaving maliciously or normally. If the sender type is *malicious* and decided to attack by launching an intrusion the expected probability to be detected by leader-IDS is E_{leader}. Since the intrusion could be launched iteratively and could be missed in the coming iterations, the IDS will decide to inform the victim node to launch its own IDS if the probability of attack is greater than $\frac{C_{victim}}{\mu\, V\, E_{victim}}$.

On the other hand, the utility function U_a of the attacker is defined as follows:

$$U_a = qp(\overline{E_r}\, V - C_a) + p(1-q)(\overline{E_m}\, V - C_a)$$

The main objective of the attacker is to maximize this utility function by choosing for a fixed q^*, a p^* that maximizes the probability of compromising the victim node and leads to equilibrium where the following holds:

$$U_a((p^*|\theta = M), q^*) \geq U_a((p|\theta = M), q^*)$$

To maximize the utility function, it is sufficient to set the first derivative with respect to p^* to zero which will be equal to: $q^* = \frac{C_a - \overline{E_m}\, V}{(\overline{E_r} - \overline{E_m})\, V}$. This can be reduced to $\frac{\overline{E_{leader}}\, V - C_a}{E_{victim}\, V}$.

From the solution of the game, the attacker best strategy is to attack once the probability of running the IDS by the victim (robust mode) is less than $\frac{\overline{E_{leader}}\, V - C_a}{E_{victim}\, V}$. To achieve this, the attacker will observe the behavior of the IDS at time t_k to determine whether to attack or not at time t_{k+1} by comparing its estimated observation with the derived threshold.

5.5.3 Illustrative Example

To illustrate the election scheme and the process for gradually adding more monitors, we consider a MANET of 10 nodes, as shown in Figure 15, with 3 external nodes with unknown identities. 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 4.

Table 4: 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

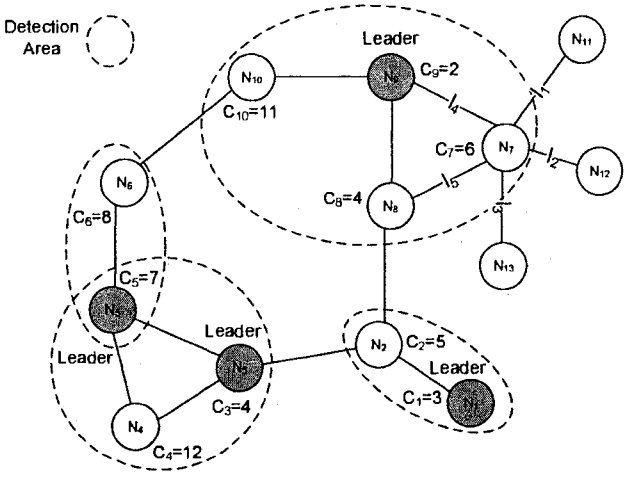

Figure 15: MANET after leader election

To elect a new leader in the 10^{th} round, nodes reveal their cost-of-analysis to the mechanism based on their type (*Selfish* or *Normal*). The corresponding cost-of-analysis is given in the second row of Table 4. 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 our payment function. The payment for elected leaders N_1, N_3, N_5 and N_9 will be 45, 95, 40 and 100 respectively. All the neighboring nodes increase the reputation of the elected leaders, as shown in the third row of Table 4. After election, leaders distribute the IDS sampling budget over the protected nodes according to their reputation. For example, leader N_9 distributes its total budget B over N_7, N_8, N_9 and N_{10} as follows: $S = \{S_7 = \frac{90\times B}{470}, S_8 = \frac{160\times B}{470}, S_9 = \frac{110\times B}{470}, S_{10} = \frac{110\times B}{470}\}$. Here, we elected a set of leader-IDS and divided the network into a collection of protected clusters.

Due to leader limited sampling budget and according to nodes' security threat level, monitored nodes (victim) that are expected to receive an attack in the coming time slots should be notified to launch their own IDS. Thus, nodes' resources are consumed according to the security needs.

After election is completed, nodes are in moderate intrusion detection mode where the leader is sampling and analyzing the packets. To demonstrate how nodes are notified according to the security needs, we show the interaction between leader-IDS N_9 and the 3 external nodes. As an example, we select node N_7 as the target node where an intruder is targeting to attack. Figure 15, describes an attack scenario where an intrusion could be directed to node N_7 either from node N_{11}, N_{12} or N_{13}. Hence, the leader-IDS will use the belief function of Equation 19 to calculate the belief of each external node using the prior observed actions. For example, we assume that the leader's belief regarding each external node connected to N_7 is $\mu = \{\mu_{11} = 0.7, \mu_{12} = 0.2, \mu_{13} = 0.1\}$. According to node's belief, the IDS will compute the threshold that determines the behavior of the external nodes (i.e., attack or not). If the probability of attack is greater than the computed threshold then the leader N_9 should inform the victim node N_7 to launch its own IDS. For example, if the threshold of attack by node N_{11} is 0.18, assuming that $C_{victim} = 10$, $V_7 = 100$ and $E_{victim} = 0.83$, then the victim node will more frequently launch its IDS. This is because the value of the node, with respect to the cluster, is much more than the cost of running the IDS. Hence, launching the IDS by the victim is affected by the ratio of the monitoring cost to the value of the node (gateway, normal, etc.).

5.6 Simulation Results

In this section, we evaluate the performance of our model with respect to random and connectivity models. Also, we evaluate the performance of our moderate to robust model with respect to resource consumption and security. We simulate the schemes using Network Simulator 2 (NS2) [60] and MATLAB.

5.6.1 CILE Simulation Results

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.

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 5: 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 sec
Traffic Type	CBR/UDP
Packet Rate	4 packets/sec
T_{ELECT}	20 secs

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 5 summarizes our simulation parameters.

Simulations

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 16.(a) depicts the impact of selfish nodes on the life of normal nodes. The result indicates that the

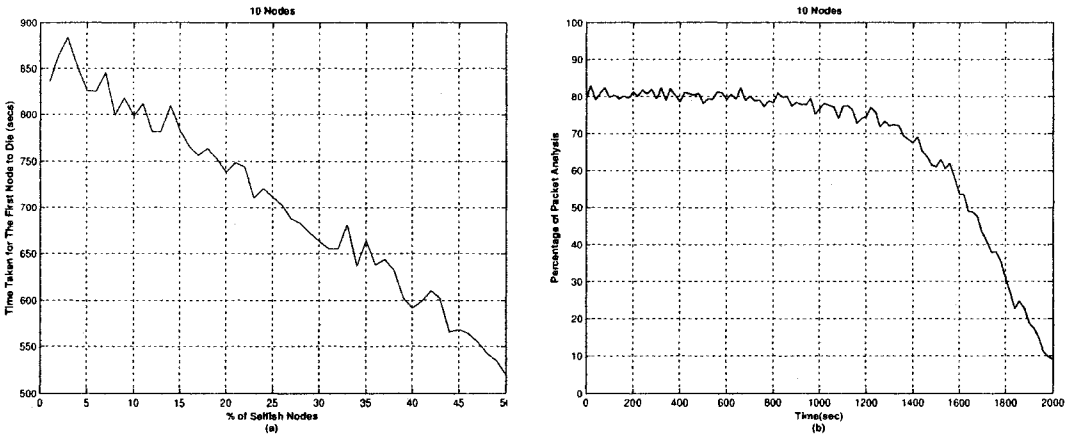

Figure 16: Effect of selfish nodes on the other nodes

normal nodes will carry out more duty of intrusion detection and die faster when there are more selfish nodes. Figure 16.(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 16.(b). This is a severe security concern since fewer packets are analyzed.

In Figure 17, 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 12 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 $T_{ELECT} = 10$ 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

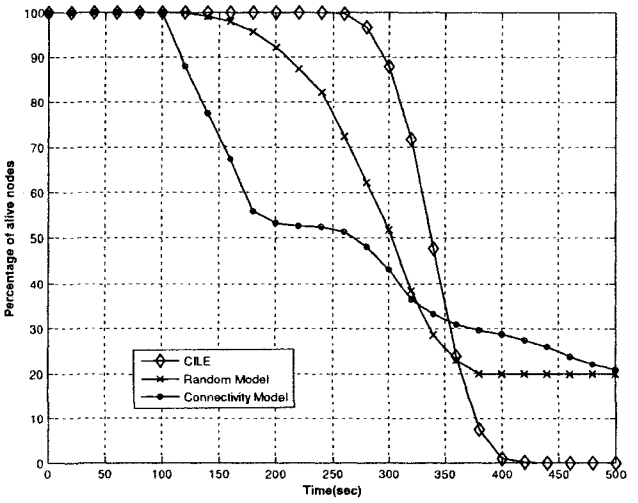

Figure 17: Percentage of alive nodes

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 17.

Figure 18.(a) indicates that our model is able to balance the resource consumption among all nodes. On the other hand, the random (Figure 18.(b)) and connectivity (Figure 18.(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 5. We compare our model only with the connectivity model since we believe that the expected

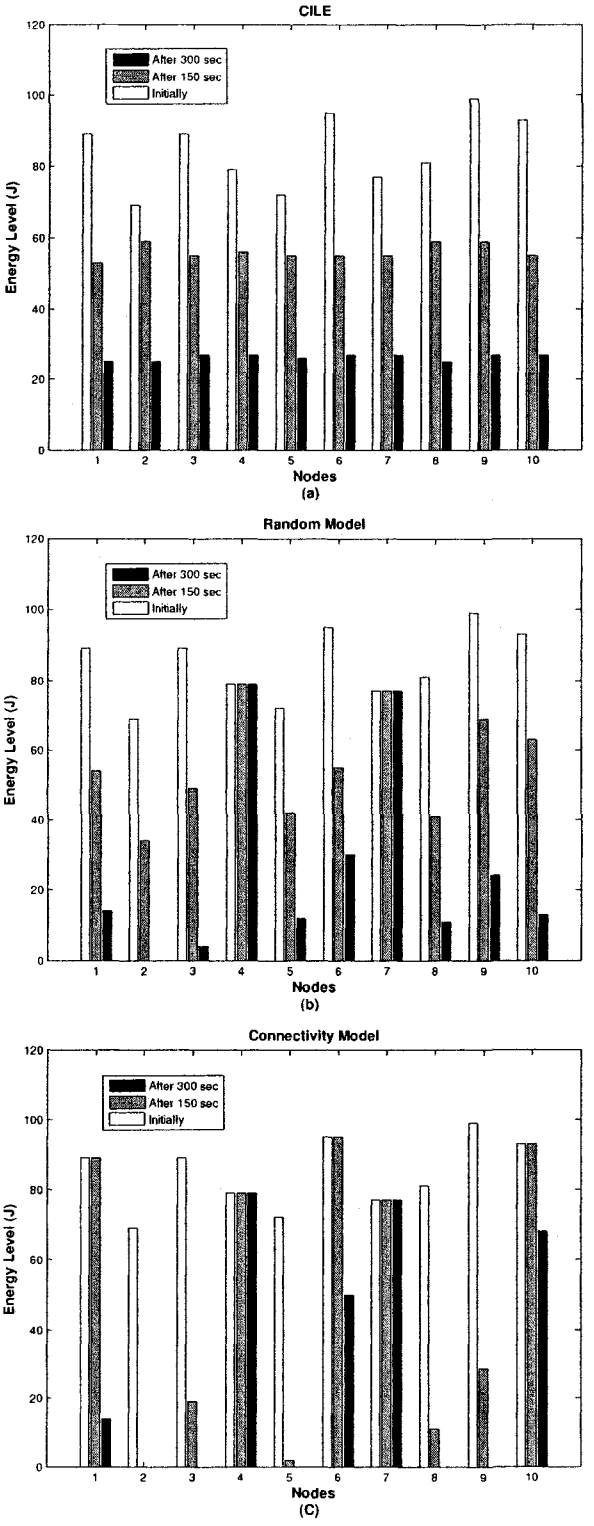

Figure 18: Energy level of the nodes

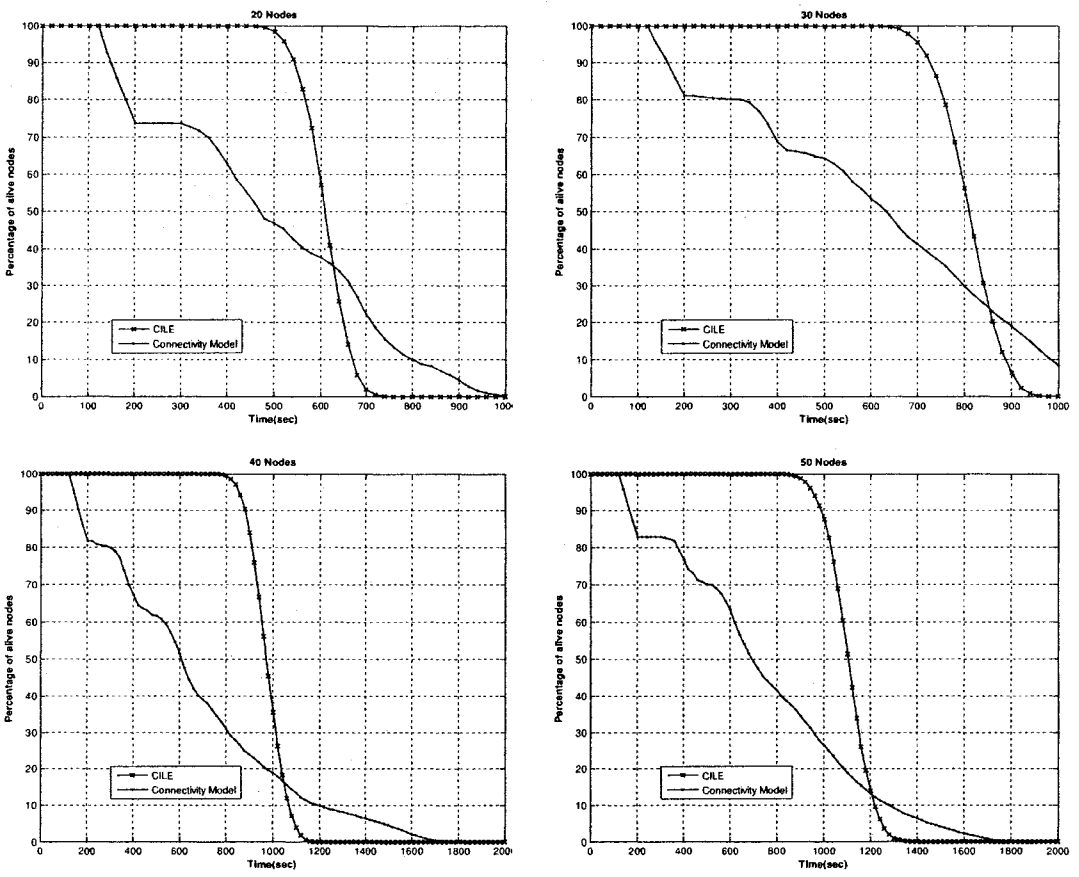

Figure 19: Percentage of alive nodes in dynamic network

performance of the random model will be close to the one given with low mobility (static network). Figure 19 shows that more nodes are alive in our model compared to the connectivity one. Knowing that we embedded our motivation and punishment model to both CILE and connectivity models since the latter suffers from the presence of selfish nodes. From the figure, we can realize that 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.

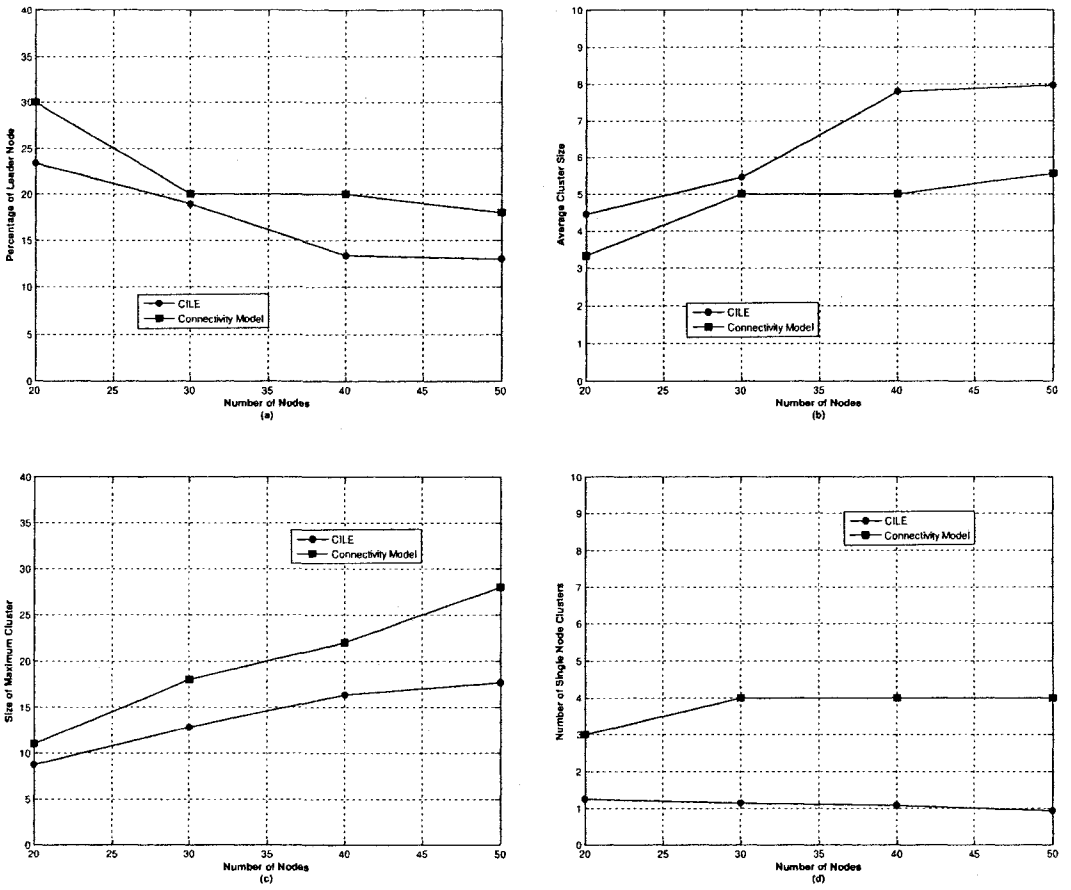

Figure 20: Comparison of cluster characteristics

Last but not least, we compare some of the cluster characteristics of our model with those of the connectivity model. Figure 20.(a) shows the percentage of the leader nodes. The percentage of leaders for our model is less as compared to those of the connectivity model that saves the energy of nodes. Figure 20.(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. Moreover, this can help us to formulate efficiently our catch and punish model.

Figure 20.(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 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 20.(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.

5.6.2 Moderate to Robust Model Simulation results

To simulate our model, we assume the leader-IDS collects packets via sampling in each round to determine whether there is an attack or not. The output of the leader-IDS ranges between 0 and 1. If the computed output is less than 0.8 then it is classified as a normal behavior, otherwise it is abnormal (attack). Figure 21.a shows the behavior of an external node (node N_{11} in the previous example) for two different attack scenarios for 40 consecutive rounds. To determine the type of the sender (N_{11}), the posterior belief function is calculated using Equation 19 with prior belief $\mu_0 = 0.5$, $F_m = 0.1$ and $E_m = 0.83$. Figure 21.b shows the posterior belief of the leader for these two attack scenarios. The belief

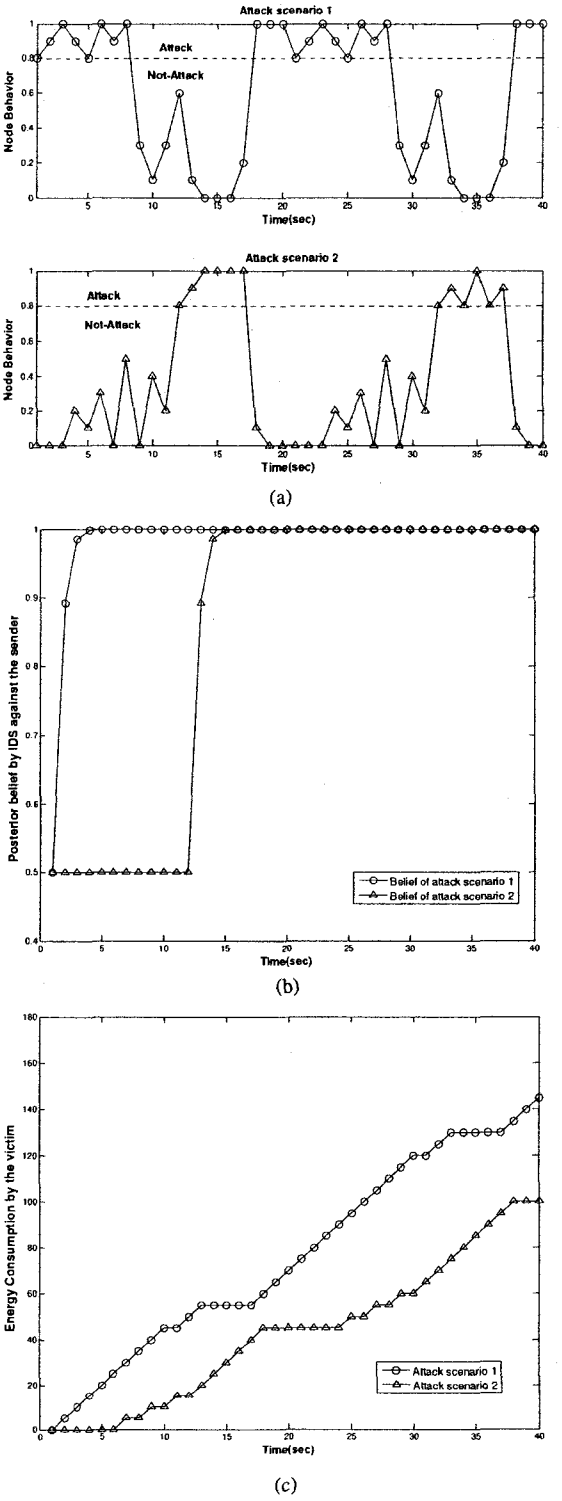

Figure 21: Attack scenarios, posterior belief function and energy consumption

for the first attack scenario converges to 1 faster than the second attack scenario. This is because in the first scenario the attacker starts to attack earlier compared to the second scenario. Once the belief reaches 1, it does not go down even if the attacker is not attacking since the type already been identified. After calculating the belief, the leader-IDS computes the attack threshold. The victim node launches its own IDS according to the notification of the leader-IDS. Figure 21.c illustrates the cumulative energy consumption by the victim node for the two attack scenarios. We assume that the victim node consumes 5 joules of energy for launching the IDS for one round. Thus, if the node is always monitoring it consumes $40 \times 5 = 200J$ for the 40 rounds. On the other hand, in our model the victim node consumes $145J$ and $100J$ for the two attack scenarios respectively. This will prolong the IDS lifetime. Thus, the victim launches its IDS (robust mode) optimally depending on the frequency of attack and the ratio of the monitoring cost to its value.

5.7 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 analyzed the algorithm correctness to verify whether it meets our objectives. Additionally, we discussed different security properties of the algorithm.

The tradeoff between security and the resource consumption of IDSs has motivated us to propose a game-theoretical solution for prolonging the lifetime of nodes and increasing their security. A nonzero-sum noncooperative game was formulated to analyze the interaction between the leader-IDS and intruder. The game guided the two players to know their optimal strategy against each other. The leader-IDS notifies the victim node to launch its own IDS once the probability of attack is greater than the game-derived threshold.

To study the performance of our election model, we compared it 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 lifetime 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 lifetime 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 fast. 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. Finally, we showed the simulation results of the moderate to robust model where we described how our model is able to reduce IDS resource consumption according to the security risk.

In the next chapter, we find the optimal sampling strategy that can increase the expected probability of detection.

Chapter 6

Multi-fragment Intrusion Detection via Sampling

In this chapter, we consider the detection of multi-fragments intrusion that can be launched from a MANET targeting another one in different geographical area. Knowing that MANETs in different geographical areas are connected over wired lines. We decided to generalize our solution to wired infrastructure based network since such an intrusion is still applicable to it. To detect such an intrusion, an intrusion detection system is needed as a second line of defense to detect intrusions and consequently generate the appropriate responses. Typically, detecting an unusual activity is done through monitoring and analyzing the network traffic searching for an unusual activity. Analyzing the traffic could be achieved by either considering the whole traffic or by sampling a portion of the traffic looking for intrusions. Analyzing the whole traffic is considered costly since it needs time and consumes a lot of

resources such as memory and CPU. On the other hand, analyzing the network using sampling is less costly but it has the problem of missing some intrusions due to its sampling budget constraint. Therefore, finding a strategy that is capable of enhancing the probability of detection using sampling is considered as a challenging problem especially when we consider the case of smart intruders and cooperative intruders that are capable of sending an intrusion through multiple fragments[1] [75].

Our main goal, in this chapter, is to consider the following two scenarios:

1. First, we consider the presence of a smart intruder that is able to divide the intrusion over different fragments to attack a victim node. Moreover, the intruder is able to select the routing paths to inject the fragments to harden the possibility of detection. On the other hand, the IDS objective is to sample according to the sampling budget looking for the fragments and collecting at least m out of n. Knowing that n is the total number of fragments that form the intrusion.

2. Second, we consider the case where we have a group of cooperative intruders. The intruders initiate an intrusion by sending a series of fragments from different sources using different routes. The IDS objective is to divide the sampling budget over the intruders and sample for each one according to the new budget. Note that the IDS detect an intrusion if all the fragments are analyzed.

In summary, our work aims at developing a network packet sampling policy to effectively reduce the success chances of an intruder by finding the value of the game using

[1]Fragrouter is a network intrusion detection evasion tool that could be used by an attacker to divide an intrusion into different fragments. It implements most of the attacks described in [75].

a min-max strategy [65]. This game can guide the Internet Service Provider (ISP) to optimally distribute the intrusion detection budget over the network links to maximize the probability of detection. Non-cooperative game theory with complete information about the players is used to formally express our problems, where the players are: (1) the cooperative intruders or a smart intruder (depends on which scenario we are solving) and (2) the intrusion detection system. This game theoretic model will guide the IDS to have an optimal sampling strategy in order to detect the malicious packets. The strategy for each intruder node is the probability of choosing each possible path to send its malicious packet to the victim node. Consequently, the optimal strategy for the IDS is to assign the sampling rates to each link to maximize the probability of detection while not exceeding the total predetermined budget.

The rest of the chapter is organized as follows. In Section 6.1, we consider the first scenario where an attacker distributes an intrusion over multiple fragments. We present the problem statement and then illustrate the assumptions. Next, we introduce the game and discuss the constraints and objective of the game. Sections 6.2 and 6.3 present the game formulation and solution respectively. In the solution, we consider the case where m out of n fragments are needed to detect an intrusion. Sections 6.4 and 6.5 present the second scenario where a distributed attack is launched via cooperative intruders. First, a game theoretic framework is built and then the solution of the game is introduced providing strategies for both the IDS and the intruders. A case study is presented as well. Section 6.6 discusses the game results through numerical results, which is followed by the concluding remarks in Section 6.7.

6.1 Problem Statement

The problem set-up is outlined in four steps. First, we discuss the assumptions in the network. Then, we introduce the game defining the adversaries in a game theoretic framework. Afterward, we describe the objective of the game that is played between the adversaries and finally introduce strategies for the players.

6.1.1 Network Model and Assumptions

The network is modeled as a directed graph, $G = (N, E)$ where N is the set of nodes and E is the set of unidirectional links. It is also assumed that there are k nodes and l links in the network. The capacity of link $e \in E$ is denoted by c_e and the amount of traffic flowing on link e is represented by f_e where f_e value can be controlled by the ISP. Given two nodes u and v in the network, let ρ_u^v represent the set of paths from u to v in G. We present the maximum flow between u and v with $MF_u^v(c)$, where c is the capacity vector. Corresponding to the maximum flow between nodes u and v, there is a minimum cut [83] consisting of a set of links in the network. The set of links in this minimum cut will be represented by $Mincut_u^v$. In our study, we consider two scenarios. In the first scenario, an attacker can split an intrusion over n packets each containing a fragment of the attack. We assume the intruder injects a fragment of the intrusion by selecting a path to the target using the Multi-Protocol Label Switching (MPLS) protocol [78]. This harden the detection possibility by IDS. Moreover, we assume the intruder source is known to the IDS which could be a region by itself. On the other hand, we have a distributed intrusion detection

system that is detecting attacks over multiple packet and the intrusion is being detected if a fraction of the *a-fragments* are being analyzed. The IDS can detect the intrusion if m *a-fragments* are being detected where $m \leq n$. For the second scenario, we introduce Ω to be the set of cooperating intruders, each sending a fragment of the intrusion to the target node t, in order to initiate the attack, where $|\Omega|$ is the number of intruders. Also, we introduce s_e to be the sampling rate on link e.

6.1.2 Introducing the Games

In the first scenario, we assume that the game is played on an infrastructure-based network between two players: The IDS and the intruder. The objective of the intruder is to inject n *a-fragments* from some attacking node $a \in N$ with the intention of attacking a target node $t \in N$. An intrusion is successful when at least m *a-fragments* out of the n *a-fragments* reach the desired target node, t, without detection. In order to detect the intrusion, the IDS is allowed to sample packets in the network. Without loss of generality, it is assumed that sampling takes place on the links in the network. The game is illustrated in Figure 22-a. In the second scenario, the game is played between the IDS and the cooperating intruders. Assuming the set of cooperative intruders as one player, we model the game as a zero-sum game with complete information about the: IDS and intruders. The objective of each intruder $x \in \Omega$ is to send an *a-fragment* to the target node t. An intrusion is successful when all the fragments reach the desired target node t without detection. In order to detect the intrusion, the IDS samples packets in the network via its agents. Furthermore, the agents sample the traffic on each link in the network as shown in Figure 22-b.

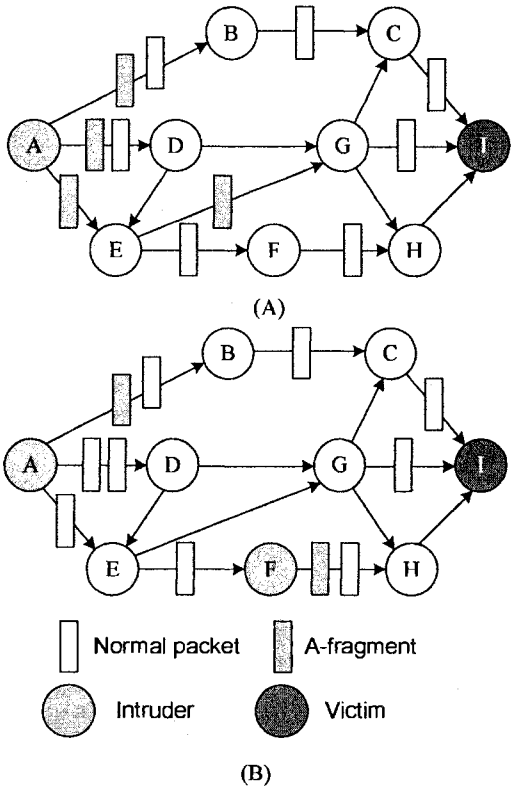

Figure 22: Single intruder and cooperative intruder games

6.1.3 Game Objectives and Constraints

Sampling all the packets flowing on a link and examining these packets can be fairly expensive to perform in realtime. Therefore, we assume that the IDS has a sampling budget of B_s packets/second over the entire network. This sampling effort can be distributed arbitrarily over the links in the network. Here, we assume a distributed agent based IDS [32] and not a central one to avoid single point of failure. The IDS samples the packets on each link via the agents while not exceeding the sampling budget, B_s. The sampling bound can be viewed as the maximum rate at which the intrusion detection system can process packets in realtime. If a link e, with traffic f_e flowing on it, is sampled at rate s_e, then the probability

of sampling a malicious fragment on this link is given by $p_e = s_e/f_e$. Therefore, we have the sampling constraint $\sum_{e \in E} s_e \leq B_s$. The game theoretic problem that we are going to discuss in the next sections, is formulated in terms of p_e. We assume that all the players have complete information about the topology of the network and all the link flows in the network.

6.1.4 Players' Strategies

In the case of the intruder, in the two scenarios, a pure strategy would be to pick a path $P \in \rho_x^t$ for the malicious packet to traverse from x to t. The intruder, in our case, can use a mixed strategy. In the case of a mixed strategy, the intruder has a probability vector $q_x = (q(P_{1_x}), ..., q(P_{z_x}))$ over the set of paths in $\rho_x^t = \{P_{1_x}, P_{2_x}, ..., P_{z_x}\}$ such that $\sum_{P \in \rho_a^t} q(P) = 1$. Moreover, let $V_x = \{q : \sum_{P \in \rho_x^t} q(P) = 1\}$ represent the set of feasible probability allocations over the set of paths between x and t. The intruder, x, then picks a path $P \in \rho_x^t$ with probability $q_x(P)$ for each malicious packet. The strategy for the IDS is to choose the sampling rate s_e on link e such that $\sum_{e \in E} s_e \leq B_s$. We also introduce $U = \{p : \sum_{e \in E} f_e p_e \leq B_s\}$ to represent the set of detection probability vectors $p = (p_{e_1}, ..., p_{e_l})$ that satisfy the sampling budget constraint. The strategy for the IDS is to pick a set of detection probabilities at the links which belongs to the set U.

6.2 Intruder-IDS Game Formulation

Having the intruder and IDS each chosen their strategies, (i.e., their probability distributions, (1) q over the set of paths in ρ_a^t, and (2) p a set of detection probabilities at the links for the intruder and IDS respectively). The payoff for both the IDS and the intruder depends on the probability of the intrusion being detected as it goes from a to t. The probability of sampling an *a-fragment* traversing from node a to node t is the sum of probability of taking each path times the probability of sampling the *a-fragment* on that particular path over all possible routes from a to t. Denote α to be such probability then we have:

$$\alpha = \sum_{P \in \rho_a^t} q(P)[1 - \prod_{e \in P} (1 - p_e)] \tag{20}$$

Therefore, the probability of detecting an intrusion that requires exactly m *a-fragments* is,

$$\alpha^m \times (1 - \alpha)^{n-m}. \tag{21}$$

Notice that the IDS will detect the intrusion if at least m *a-fragments* are sampled. Hence, the IDS will detect the intrusion with probability,

$$\sum_{i=m}^{n} \alpha^i \times (1 - \alpha)^{n-i} \tag{22}$$

Accordingly, the IDS will choose a strategy that maximizes the detection probability:

$$\max_{p \in U} \sum_{i=m}^{n} \alpha^i \times (1 - \alpha)^{n-i}, \tag{23}$$

where,

$$U = \{p : \sum_{e \in E} f_e p_e \leq B_s\}.$$

On the other hand, the objective of the intruder is to choose a distribution q and number of fragments n that minimize this maximum value. In other words, the objective of the intruder is:

$$\min_{n \in \mathbb{N}, q \in V} \max_{p \in U} \sum_{i=m}^{n} \alpha^i \times (1-\alpha)^{n-i}. \tag{24}$$

Using a similar argument, the objective of the IDS becomes:

$$\max_{p \in U} \min_{n \in \mathbb{N}, q \in V} \sum_{i=m}^{n} \alpha^i \times (1-\alpha)^{n-i}. \tag{25}$$

This is a classical two person zero-sum game. According to minmax theorem [95], there exists an optimal solution to the intrusion detection game where the following noted min-max result holds,

$$\begin{aligned} \theta &= \max_{p \in U} \min_{n \in \mathbb{N}, q \in V} \sum_{i=m}^{n} \alpha^i \times (1-\alpha)^{n-i} \\ &= \min_{n \in \mathbb{N}, q \in V} \max_{p \in U} \sum_{i=m}^{n} \alpha^i \times (1-\alpha)^{n-i}, \end{aligned} \tag{26}$$

and θ is the value of the game.

6.3 Game Solution

In this section, we suppose to solve the game for the case where the IDS requires at least m a-fragments to detect an intrusion as illustrated in Section 6.2. Due to the mathematical complexity on solving the game in Equation 26, we solve the game for the case an intrusion detection requires only m a-fragments out of n. By recalling Equation 21, the game is reduced to the following:

$$\theta = \max_{p \in U} \min_{n \in \mathbb{N}, q \in V} \alpha^m \times (1-\alpha)^{n-m} = \min_{n \in \mathbb{N}, q \in V} \max_{p \in U} \alpha^m \times (1-\alpha)^{n-m}, \tag{27}$$

Considering the intruder problem the game is reduced to the following:

$$\min_{n \in \mathbb{N}, q \in V} \max_{p \in U} \alpha^m \times (1-\alpha)^{n-m} \tag{28}$$

For a fixed $q \in V$ and n, it is sufficient to solve the following:

$$\max_{p \in U} \alpha^m \times (1-\alpha)^{n-m} \tag{29}$$

To maximize equation 29, it is sufficient to maximize α. Thus, we have to find the value of p_e that maximizes the function which is done as follows:

$$\max_{p \in U} [\sum_{P \in \rho_a^t} q(P)[1 - \prod_{e \in P} (1-p_e)]], \tag{30}$$

which is equal to

$$\max_{p \in U}[\sum_{P \in \rho_a^t} q(P) - \sum_{P \in \rho_a^t} [q(P) \prod_{e \in P} (1 - p_e)]], \tag{31}$$

or alternatively one can minimize the following,

$$\min_{p \in U} \sum_{P \in \rho_a^t} [q(P) \prod_{e \in P} (1 - p_e)]. \tag{32}$$

This objective function is non-linear with respect to p_e which makes the problem intractable. Therefore, we have to linearize this function before optimizing it. Given the assumption of sampling is bounded with a budget that restricts the sampling efforts, we are going to allocate our sampling efforts on the links that belongs to the $Mincut_a^t$ set. This strategy will reduce the sampling efforts from sampling all the links between a and t to sampling some critical links. Since sampling will be done for at most one link in path P, we can rewrite equation (32) as:

$$\min_{p \in U} \sum_{P \in \rho_a^t} [q(P)(1 - \sum_{e \in P} p_e)]. \tag{33}$$

To minimize this equation, it is sufficient to maximize the following:

$$\max_{p \in U} \sum_{P \in \rho_a^t} [q(P) \sum_{e \in P} p_e]. \tag{34}$$

Subject to the following constraints:

$$\sum_{e \in E} f_e p_e \leq B_s,$$

$$p_e \geq 0.$$

Associating a dual variable λ [72], we obtain the following dual optimization problem with the corresponding constraints:

$$\min B_s \lambda, \tag{35}$$

$$\forall e \in E, \sum_{e \in E} f_e \lambda \geq \sum_{P \in \rho_a^t, e \in P} q(P),$$

$$\lambda \geq 0,$$

$$\sum_{P \in \rho_a^t} q(P) = 1.$$

Interpreting $q(P)$ as a flow on path P, the constraint

$$\sum_{P \in \rho_a^t, e \in P} q(P) \leq \sum_{e \in E} f_e \lambda,$$

restricts the flow for all links to be at most $\sum_{e \in E} f_e \lambda$. Hence, $\sum_{e \in E} f_e \lambda$ can be interpreted as the capacity of link e. The constraint $\sum_{P \in \rho_a^t} q(P) = 1$ enforces one unit flow to be sent from node a to node t.

The objective of the game is therefore to determine the smallest λ so that a flow of one unit can be sent from node a to node t. This can be done as follows:

- Assume that link e has capacity f_e and determine the maximum flow from a to t,

$MF_a^t(f)$, using these capacities.

- Scale the capacities by $MF_a^t(f)^{-1}$ so that a flow of one unit can be sent from node a to node t.
- λ will be $MF_a^t(f)^{-1}$.
- The value of the game is $\theta = B_s MF_a^t(f)^{-1}$.

From the network flow duality, corresponding to the maximum flow value there is a minimum cut. The IDS computes the maximum flow from a to t using f_e as the capacity of the link e. Let $e_1, e_2, ..., e_r$ denote the arcs in the corresponding minimum cut with flows $f_1, f_2, ..., f_r$. From duality $\sum_{i=1}^{r} f_i = MF_a^t(f)$. The IDS samples link e_i at rate $B_s f_e MF_a^t(f)^{-1}$. On the other hand, we assume the intruder fragments the intrusion into n fragment where $n \leq l$, where l is the number of paths. Moreover, the intruder uses the standard flow decomposition techniques to decompose the maximum flow into flow on paths $P_1, P_2, ..., P_l$ from node a to node t with flows of $m_1, m_2, ..., m_l$ respectively (note that $\sum_{i=1}^{l} m_i = MF_a^t(f)$). The intruder transmits each malicious fragment packet along the path P_i with probability $m_i MF_a^t(f)^{-1}$.

We now illustrate the results in Section 6.3 on the example shown in Figure 23. The numbers next to the links are the flows on the links. Suppose that there is a sampling budget B_s of 12 units for the IDS. Additionally, we assume the intruder's fragmentation is equal to 3 where $a = A$ and $t = I$ are the intruder and victim respectively. The links (C, E), (B, D) and (B, G) belong to the minimum $a - t$ [83] cut which are shown in bold lines. The minimum cut (and hence the maximum flow) has a value of 29 units.

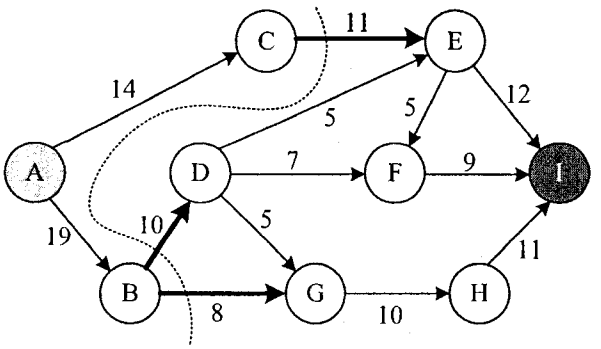

Figure 23: Single intruder with multiple *a-fragments*

The intruder launches the attack over 3 fragments where each fragment is forwarded by selecting a path with the following probabilities:

- Path $A - C - E - I$ with probability $11/29$.
- Path $A - B - G - H - I$ with probability $8/29$.
- Path $A - B - D - F - I$ with probability $7/29$.
- Path $A - B - D - G - H - I$ with probability $2/29$.
- Path $A - B - D - E - F - I$ with probability $1/29$.

Correspondingly, the distributed IDS's strategy is the following:

- Sample link (C, E) with the sampling rate $s_e = (12 \times 11)/29$.
- Sample link (B, G) with the sampling rate $s_e = (12 \times 8)/29$.
- Sample link (B, D) with the sampling rate $s_e = (12 \times 10)/29$.

Note that the total sampling budget is equal to 12.

6.4 Cooperative Intruders-IDS Game Formulation

In this section, we extend the previous game to the case where multiple intruders will cooperate with each other to attack the same target. Knowing that the intrusion is fragmented to n fragments. The objective of each intruder $x \in \Omega$ is to send a fragment of the intrusion to the target node t where Ω is the number of intruders. Here, we assume that the intrusion is successful if all the fragments are delivered without detection. Therefore, the game is played between cooperative intruders and IDS where the IDS samples the packets for each link in the intruder's $Mincut_x^t$ set while not exceeding the total sampling budget, B_s.

The intruders and IDS should choose their strategies, which are the probability distributions: q_x over the set of paths in ρ_x^t, and p a set of detection probabilities at the links for the intruders and the IDS respectively. The objective of each intruder is to inject a fragment of the intrusion by selecting the path that can reduce the IDS probability of detection. Thus, the payoff for both the IDS and intruders depends on the probability of the intrusion being detected as it goes from the intruding nodes to the target node t. For any node $x \in \Omega$, the probability of detecting a fragment of the intrusion traversing from node x to node t is the sum of probability of taking each path times the probability of sampling the packet on that particular path over all possible routes from x to t. We introduce α_x to be the probability of detecting the intrusion, when intruder x is attacking node t, which is given by:

$$\alpha_x = \sum_{P \in \rho_x^t} q(P)[1 - \prod_{e \in P} (1 - p_e)] \tag{36}$$

Next, we define the function Φ to be the mean value of detecting the intrusion through

sampling:

$$\Phi = \frac{1}{|\Omega|} \sum_{x \in \Omega} \alpha_x \tag{37}$$

The main goal of the IDS is to maximize Φ. In other words, the IDS aims at maximizing the following:

$$\max_{p_e \in U} (\Phi = \frac{1}{|\Omega|} \sum_{x \in \Omega} \alpha_x) \tag{38}$$

where:

$$U = \{p : \sum_{e \in E} f_e p_e \leq B_s\}$$

On the other hand, the cooperative intruders aim at minimizing Equation (38). The intruders will fulfil this objective by assigning probabilities for all possible routes to the target node:

$$\min_{q \in V_x} \max_{p \in U} (\Phi = \frac{1}{|\Omega|} \sum_{x \in \Omega} \alpha_x) \tag{39}$$

where:

$$V_x = \{q : \sum_{P \in \rho_x^t} q(P) = 1\}$$

Using a similar argument, the objective of the IDS becomes:

$$\max_{p \in U} \min_{q \in V_x} (\Phi = \frac{1}{|\Omega|} \sum_{x \in \Omega} \alpha_x) \tag{40}$$

This is a mixed strategy zero-sum game, for which the following min-max theorem holds:

$$\beta = \max_{p \in U} \min_{q \in V_x} (\Phi = \frac{1}{|\Omega|} \sum_{x \in \Omega} \alpha_x)$$

$$= \min_{q \in V_x} \max_{p \in U} (\Phi = \frac{1}{|\Omega|} \sum_{x \in \Omega} \alpha_x) \tag{41}$$

where β is the value of the game.

6.5 Game Solution

In this section, we propose our solution to the min-max problem formulated in section 6.4. First, we consider the intruders' problem:

$$\min_{q \in V_x} \max_{p \in U} (\Phi = \frac{1}{|\Omega|} \sum_{x \in \Omega} \alpha_x) \tag{42}$$

For a fixed q the problem reduces to the following:

$$\max_{p \in U} (\Phi = \frac{1}{|\Omega|} \sum_{x \in \Omega} \alpha_x) \tag{43}$$

In order to maximize the previous equation, it is sufficient to maximize all the terms. Therefore, the problem simplifies to the following:

$$\max_{p \in U} \alpha_x \tag{44}$$

We know that each node is sending one packet to the target node. Therefore, we can divide the budget, B_s among the $|\Omega|$ intruders. Thus, each intruder, x, will have the budget constraint $\frac{B_s}{|\Omega|}$ or $B_s|\Omega|^{-1}$. Replacing α_x, using Equation (36) and rewriting Equation (44)

the problem can be written as follows:

$$\max_{\pi^x \in U_x} \sum_{P \in \rho_x^t} q(P)[1 - \prod_{e \in P} (1 - \pi_e^x)] \tag{45}$$

where,

$$\sum_{x \in \Omega} \pi_e^x = p_e,$$

$$U_x = \{\pi^x : \sum_{e \in E} f_e \pi_e^x \leq B_s |\Omega|^{-1}\}$$

Having the sampling constraint:

$$\sum_{e \in E} f_e \pi_e \leq B_s |\Omega|^{-1} \tag{46}$$

Thus, using the same approach as in Section 6.3, the subgame reduces to the following:

$$\min B_s \lambda \tag{47}$$

Subject to:

$$\sum_{e \in E} f_e \lambda \geq \sum_{P \in \rho_x^t, e \in P} q(P), \forall x \in \omega, \forall e \in E \tag{48}$$

$$\lambda \geq 0 \tag{49}$$

$$\sum_{P \in \rho_x^t} q(P) = 1 \tag{50}$$

Next, we calculate the maximum flow from x to t, $MF_x^t(f)$. Knowing that the maximum flow is equal to the summation of the flows on all the paths from x to t, we then normalize

the flows in the network (with respect to the $MF_t^x(f)$). Therefore, the normalized flow on each path can be interpreted as $q(P)$ and constraints (50) hold. Furthermore, interpreting $q(P)$ as the flow on path P suggests $\sum_{P \in \rho_x^t, e \in P} q(P)$ to be the normalized flow on link e. Hence, in order to minimize λ, satisfying constraint (48), we introduce λ to be the maximum flow; that is, $\lambda = MF_x^t(f)^{-1}$. Therefore, the value of the game is:

$$B_s|\Omega|^{-1}MF_x^t(f) \tag{51}$$

The game would guide us to the following strategies satisfying the budget constraint. Hence, the strategy for intruder x is:

- Calculate the maximum flow from x to t using f_e as the capacity of the link e.
- Use the standard flow decomposition techniques [1] to decompose the maximum flow into flows on paths $P_1, P_2, ..., P_{l_x}$ from node x to node t with flows of $m_1, m_2, ..., m_{l_x}$ respectively, knowing that $\sum_{i=1}^{l_x} m_i = MF_x^t(f)$ and $|\rho_x^t| = l_x$.
- Transmit the malicious packet along the path P_i with probability $m_i MF_x^t(f)^{-1}$.

Consequently, the IDS's strategy is:

- For each node $x \in \Omega$ find the minimum cut.
- Let $Mincut_x^t$ denote the set of arcs in the corresponding minimum cut.

- Sample link e at rate:

$$\sum_{x \in \Omega, e \in Mincut_x^t} B_s |\Omega|^{-1} MF_x^t(f)^{-1} f_e \tag{52}$$

Note that, $\sum_{e \in E} \sum_{x \in \Omega, e \in Mincut_x^t} B_s |\Omega|^{-1} MF_x^t(f)^{-1} f_e = B_s$ and therefore, satisfying the budget constraint.

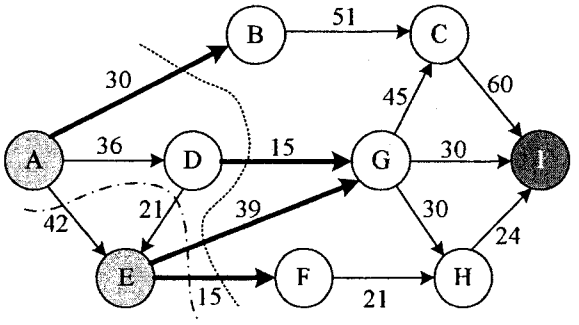

Figure 24: Cooperative multi-intruder attack

Now, we illustrate the game with an example as shown in Figure 24, where nodes A and E are the cooperative intruders and node I is the target. In other words, $\Omega = \{A, E\}$ and $|\Omega| = 2$. The budget constraint, B_s, is 60. Therefore, $B_s|\Omega|^{-1} = 30$. The maximum flow from A to I is 99 and the maximum flow from E to I is 54 as shown in Figure 24 in bold links. $Mincut_A^I = \{AB, DG, EG, EF\}$ and $Mincut_E^I = \{EG, EF\}$. Hence the IDS will sample the links as follows:

- AB with sampling rate $30 * 30/99 \simeq 9.09$
- DG with sampling rate $30 * 15/99 \simeq 4.54$
- EG with sampling rate $30 * 39/99 + 30 * 39/54 \simeq 33.47$

- EE with sampling rate $30 * 15/99 + 30 * 15/54 \simeq 12.87$

Note that the total sampling is $59.97 \leq 60$.

The intruders send the malicious packet as follows:

- Node A sends the malicious packet through path $ABCI$ with probability 30/99.
- Node A sends the malicious packet through path $ADGI$ with probability 15/99.
- Node A sends the malicious packet through path $ADEGCI$ with probability 21/99.
- Node A sends the malicious packet through path $AEGCI$ with probability 18/99.
- Node A sends the malicious packet through path $AEFHI$ with probability 15/99.
- Node E sends the malicious packet through path $EGCI$ with probability 39/54.
- Node E sends the malicious packet through path $EFHI$ with probability 15/54.

6.6 Numerical Results

In this section, we evaluate the reliability of our game model on improving the probability of detection compared to two different approaches: Random and uniform. Random is a model where sampling is done on random links. While uniform model is achieved through dividing the sampling effort equally over the links. Note that, all the models must satisfy the sampling budget constraint. To implement the three models, we use C++ as the programming language and Figure 24 as the network graph.

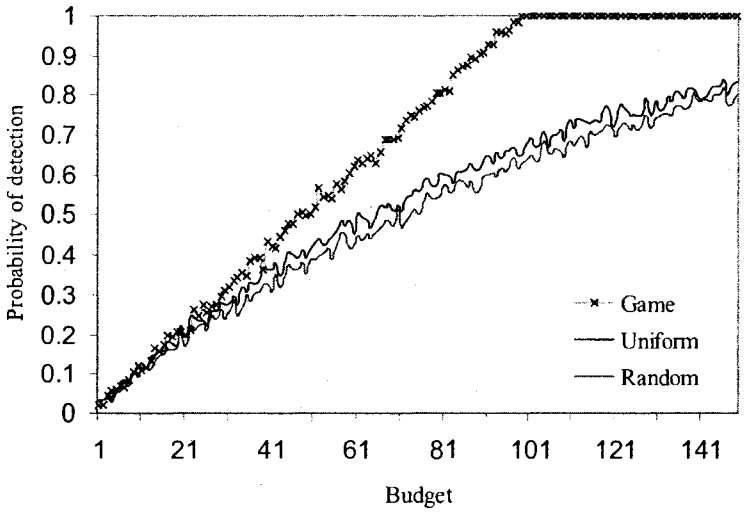

Figure 25: One intruder sending 2 *a-fragments*

First, we consider the scenario where a single intruder transmits the *a-fragments* to a target node in order to launch the attack. We consider throughout our implementation the case where an intrusion detection is fulfilled if half of the *a-fragments* are detected. Moreover, we assume that A is the attacker and I is the target. Figure 25 shows the detection probability as a function of the budget, where the budget varies from 1 to 150 (packets/second). From the case study in Section 6.4, the maximum flow between A and I is 99. As it is shown in Figure 25, as the budget reaches the maximum flow, the probability of detection becomes close to 1, no matter how many packets are being sent. This is because we are not sampling randomly or uniformly on all the edges. Instead, we focus all the budget on the minimum cut edges, where every packet transmitted from the attacker to the target has to traverse at least one of the links in the minimum cut set. From the min-cut theorem [83], we know that the summation of flows in the minimum cut is equal to the maximum flow.

Therefore, if the sampling budget is equal to or greater than the maximum flow between the attacker and target, we can sample with a rate equal to the actual flow on each link in the minimum cut. Thus, any packet either normal or malicious would be sampled ensuring that the intrusion is being detected. We can see that the game results are much better than the other two approaches. For very small sampling budgets, all the three approaches have small detection probabilities. As the budget increases, we can see that the detection probability increases, for all approaches. The reason is that we sample with higher sampling rates on the links, and thus greater sampling probabilities on each link. Therefore, the total probability of detection increases. As shown in the figure, our game approach has a greater slope; this is due to the fact that the sampling is done on the critical links where any traffic has to be transmitted through them (i.e., minimum cut). In other words, the budget is distributed over a set of critical links instead of all the links in the network, while all the traffic is still traversing through these links. This improves the detection rate.

Figure 26 illustrates the results of another scenario, where an intruder A transmits different number of *a-fragments* to a target node I having a constant sampling budget equal to 60. The attacker transmits the a-fragments through different paths. Note that there are 12 paths from A to I that could be selected randomly by the intruder. Here, the detection probability is demonstrated as a function of the number of *a-fragments*. As we can see the detection probability for odd number of *a-fragments* is less than the even ones for two consecutive numbers of *a-fragments*. This is due to the fact that the IDS needs half of the *a-fragments*, which is *one* more for the case of odd numbers. In case of larger networks, this difference between odd and even number of packets would be neglected. The results

are illustrated in Figure 26. Using the same terminology as in the previous scenario, our game theoretic framework presents better results than the other two models.

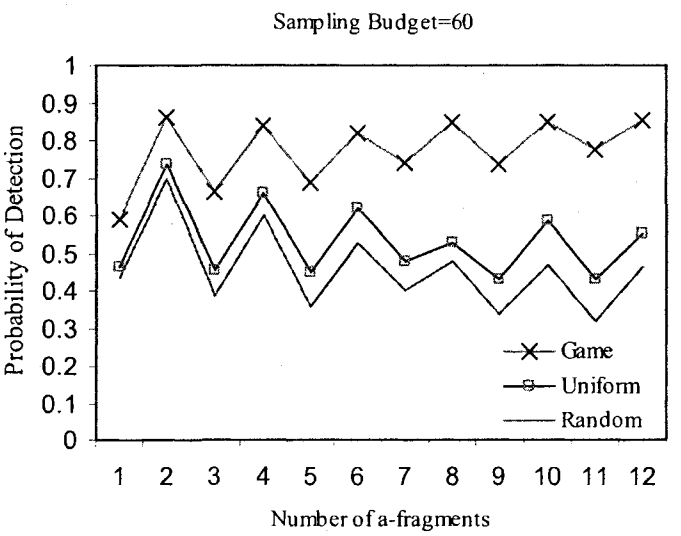

Figure 26: One intruder sending *multi-fragments*

Finally, we illustrate the multi-intruder scenario, where n cooperating intruders distribute the attack over n *a-fragments*, and where each intruder sends one *a-fragment* to a common target node. The attack is successful if half of these *a-fragments* reach the target node without being detected. The numerical results are shown in Figure 27. The detection probability decreases as the number of intruders increases. This is because the IDS has to divide the budget over the number of intruders. When the number of intruders is less than 60% of the total number of nodes in the network, focusing the sampling budget on the union of the minimum cuts for each intruder and the target node, helps in increasing the detection probability. In this case, the number of links in the union of minimum cuts

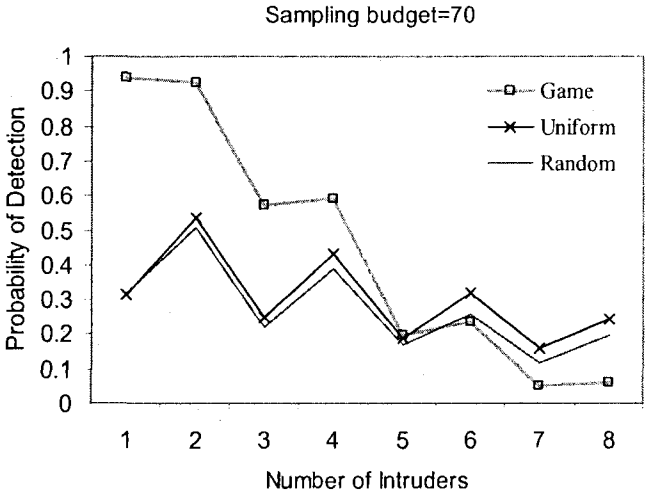

Figure 27: Multi-intruders sending one *fragment* each

are still much less than the total number of links in the network. Therefore, we distribute the total sampling budget over a smaller number of links and consequently the sampling rate increases on each link leading to better results. As the number of intruders increases, more and more links are added to the union of critical edges (union of minimum cut sets for each intruder and the target node). Thus, the set of the links becomes comparable to the total number of links. Here, the *a-fragments* are almost over all the links. In this case, the sampling budget is divided by the number of attackers, which becomes a relatively small number. The sampling rate on the other hand would be multiplied by this small sampling budget and divided by the maximum flow. Thus, the sampling probability decreases. For random and uniform strategy, the budget is independent of the number of attackers. In other words, they continue to sample almost with the same rate for any number of attackers. This shows why the uniform and random methods provide better results over the game one in

the case where intruders presence exceed 50%.

6.7 Summary

We considered the problem of intrusion detection in a network by means of packet sampling. Given a total sampling budget, we developed a network packet sampling strategy to effectively reduce the success chances of an intruder. We considered two different scenarios where the adversary has considerable information about the network and can select paths to minimize chances of detection. In the case of a single intruder, we formulated the intrusion detection problem as a zero-sum two-player game with complete information about the players: IDS and attacker. We formulated the game for the general case where at least m out of n fragments are needed to detect an intrusion. We solved the game considering the case where the intrusion detection requires m out of n fragments. Furthermore, we considered the problem of multiple cooperating intruders where the attackers can select paths independently in order to reduce the chances of detection. We formulated the intrusion detection problem as a zero-sum non-cooperative game with complete information about the IDS and the set of attackers. We solved the game to bring up strategies for both the IDS and the set of intruders. Finally, we evaluated our game solutions via numerical results. Our numerical results show the effectiveness of our game theoretic models in detecting intrusions via sampling over random and uniform models.

Chapter 7

Conclusion

The unique security characteristics of MANET motivated researchers to propose different intrusion detection systems as in Chapter 2. The proposed models suffers from one common problem which is resource consumption. Resource consumption problem is handled by electing a head cluster, which is also known as leader, to handle the intrusion detection service on behalf of all the nodes in the cluster. Such models suffer from the selfishness problem where nodes might not participate to be elected as a leader to avoid consuming their resources. The selfishness in routing for MANET was addressed under the cooperation enforcement mechanism discipline, which was addressed in Chapter 2. Selfishness is considered as a crucial problem for routing and intrusion detection, which we considered by proposing a mechanism based on mechanism design theory that was discussed in Chapter 3. Our mechanism was proposed in Chapters 4 and 5.

The leader election model is adequate in environments with low security risks. It can be seen like a security guard monitoring a home. Once the security risk is high, more guards

should be added which is considered costly in terms of money. If the security controller of a home can predict the moves of the intruders before entering the home, then he can allocate more guards according to the security needs. To achieve this, game theory which was presented in Chapter 3 is a very good tool that can analyze the steps of the intruder and help to find the optimal solution of the game. It helped us to determine the threshold for adding more guards or monitors in our case according to security needs. This game was presented in Chapter 5. So, what about the time of monitoring? In other words, for how long should the guard monitor in order to catch the intruder. Knowing that the continuous monitoring is considered costly. This problem motivated us to extend our work to more general case which can be applied to different types of networks that are connected over wired lines. The game solution guided the intrusion detection system to allocate the optimal monitoring time which is the sampling rate that can increase the probability of detection. This work was presented in Chapter 6.

7.1 Concluding Remarks

The unbalanced resource consumption of IDSs in MANET in the presence of selfish nodes has 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 process on behalf of others. Moreover, the sum of the elected leaders is globally optimal. Here, we

addressed two applications: Cluster Independent Leader Election (CILE) and Cluster Dependent Leader Election(CDLE). The former does not require any pre-clustering whereas CDLE requires nodes to be clustered before running the election mechanism.

To achieve this goal, incentives are given in the form of reputations to motivate nodes in revealing truthfully their costs of analysis. The cost function was designed taking into consideration two main properties: Fairness and privacy. Reputations are computed using the well known VCG mechanism by which truth-telling is the dominant strategy. To motivate nodes to participate in every election round, we related the reputation to detection service. Thus, the leader will sample the incoming packets of each node according to node's reputation. This will motivate nodes to participate in the leader election to increase their reputation value which effects their security level.

After the election mechanisms are designed, we analyzed the performance of the mechanisms in the presence of selfish and malicious nodes where nodes might behave selfishly/maliciosly during the election and after. According to our payment design, nodes' truth-telling strategy is the dominant strategy among all nodes. Thus, selfish nodes that are considered as rational have no incentive to deviate from telling the truth about their cost function. Therefore, our election mechanism can handle the selfishness problem during the election process.

To catch and punish the misbehaving leader (selfish or malicious) after election, we proposed a cooperative decision model based on game theory. This is done by electing randomly a set of checkers that will monitor the behavior of the leader by mirroring a portion of leader's work. Cooperative game theory was used to analyze the contribution of

each checker on the catch decision which will reduce the false positive rate generated by the nodes. According to the detection level, checkers are gradually added to observe the behavior of the leader which reduces the overall resource consumption by the nodes. We assumed that checkers are motivated to perform this job since security risk is valuated much more than the consumed resources. Due to the presence of checkers, malicious nodes have no incentive to be elected as a leader since it will be catched easily. Note that malicious nodes are rational in the sense they want to attack without being detected.

To implement the leader election mechanism, we devised a leader election algorithm with reasonable performance overheads. Moreover, we considered the mobility of nodes where nodes might be moved in and out randomly. Add and remove algorithms were devised to handle this issue. To analyze the correctness of our algorithms, we provided an informal analysis to show that the devised algorithms are able to provide the intrusion detection service for all the nodes where single nodes will launch their own IDS. Moreover, we analyzed some of the well known security properties to verify that our algorithm can resist such type of security flaws. Also, we analyzed our leader election mechanism and algorithm against cheating.

The tradeoff between security and the resource consumption of IDSs has motivated us to propose a game-theoretical solution for prolonging the lifetime of nodes and increasing their security. A nonzero-sum noncooperative game was formulated to analyze the interaction between the leader-IDS and intruder. The game guided the two players to know their optimal strategy against each other. The leader-IDS notifies the victim node to launch its

own IDS once the probability of attack is greater than the game-derived threshold. Simulation results showed that our model is able to reduce IDS resource consumption according to the security risk.

Simulation results showed that our leader election model is able to prolong the lifetime and balance the overall resource consumptions among all the nodes in the network. Moreover, the CILE model can decrease the percentage of leaders, single node clusters, maximum cluster size and increase average cluster size. These properties allowed us to improve the detection service through distributing the sampling budget more uniformly.

After leaders are elected for each MANET, the question that we raised is: How to consider attacks that are launched from one MANET to another? Knowing that all proposed IDS models for MANET are modeled to handle intrusions that are launched in the same network. More specifically, some of the attacks in MANET can be designed using multi-fragments to evade the IDS. Thus, we proposed an intrusion detection model based on game theory that can handle multi-fragment intrusions. Knowing that intrusion detection is accomplished via sampling to reduce the resource consumption. Since such types of intrusions can be launched from any type of network targeting any other type, we preferred to generalize our solution. We assumed that different types of networks, including MANET, in different geographical areas are connected over wired lines. This will help us to have a static flow of links. Our solution considered two scenarios. First, a single intruder with known network source is attacking a victim in a known target network using multi-fragment intrusion. Second, multi-intruders with known network sources are cooperating to launch a multi-fragment intrusion where each intruder injects a fragment of the intrusion. The

attacker attacks by selecting a path to the victim node while the IDS will allocate the critical links that are used by the attacker to distribute the network sampling budget. Minimum cut algorithm was used to allocate the critical links while game theory was used to calculate the optimal sampling strategy of each link to increase the probability of detection. The game solution guided both the attacker and IDS to find their best response against each other. Thus, the attacker strategy was to select the path to inject a fragment of attack according to its maximum flow. On the other hand, the IDS samples according to link's flow and attacker's maximum flow. Numerical results showed that game theory performed much better than random and uniform models with respect to intrusion detection.

7.2 Future Work

As a future work, we will consider the following:

- To extend our model to other election applications in MANET; such as, certificate authority. Moreover, to have a unified model that handle MANET services.
- To establish a cooperative detection model among elected leaders using repeated game theory. In other words, to motivate leaders to cooperate with each other on intrusion detection.
- To enhance the watchdog monitor to handle IDS misbehavior.

7.3 List of Publications

The following are the list of publications derived from and related to the thesis:

- **Book Chapter**:

  1. *Mobile Security: Game Theory.* In the Encyclopedia of Wireless and Mobile Communications Book, CRC Press, Taylor and Francis Group, 2007.

- **Journals**:

  1. *A Game Theoretic Intrusion Detection Model for Mobile Ad-Hoc Networks.* Journal of Computer Communications, Elsevier, special issue on Algorithmic and Theoretical Aspects of Wireless Ad Hoc and Sensor Networks, Vol. 31, Issue 4, March 2008, pages: 708 – 721.

  2. *Game Theoretic Models for Detecting Network Intrusions.* Journal of Computer Communications, Elsevier, Vol. 31, Issue 10, June 2008, pages: 1934 – 1944.

- **Conferences**:

  1. *A Moderate to Robust Game Theoretical Model for Intrusion Detection in MANETs.* Proceedings of the 4^{th} IEEE International conference on Wireless and Mobile Computing, Networking and Communications (Security and Privacy Workshop - SecPri-WiMob'08), October 12, 2008, Avignon, France, pages: 608 – 612 .

  2. *A Mechanism Design-Based Secure Architecture for Mobile Ad Hoc Networks.* Proceedings of the 4^{th} IEEE International conference on Wireless and Mobile

Computing, Networking and Communications (WiMob'08), October 12, 2008, Avignon, France, pages: 417 – 422.

3. *A Mechanism Design-Based Multi-Leader Election Scheme for Intrusion Detection in MANET.* Proceedings of IEEE Wireless Communications & Networking Conference (WCNC'08), Las Vegas, Nevada, USA, March 2008, pages: 2816 – 2821.

4. *An Efficient and Truthful Leader IDS Election Mechanism for MANET.* Proceedings of the 3^{rd} IEEE International conference on Wireless and Mobile Computing, Networking and Communications (WiMob'07), USA, 2007.

5. *A Cooperative Approach for Analyzing Intrusions in Mobile Ad hoc Networks.* Proceedings of the 27^{th} International Conference on Distributed Computing Systems - Workshops (ICDCS Workshops 2007), Toronto, Canada, June, 2007.

6. *Testing Intrusion Detection Systems in Mobile Ad-hoc Networks: A Comprehensive Study.* Proceedings of the Communication Networks and Services Research Conference (CNSR'07), NB, Canada, 2007, pages: 364 – 371.

7. *A Game Theoretic Approach to Detect Network Intrusions: The Cooperative Intruders Scenario.* Proceedings of the IEEE GlobeCom 2006 conference, San Francisco, California, USA, 2006.

Bibliography

[1] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, N.J., 1993.

[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. In *proceedings of the International Workshop on Wireless Information Systems*, pages 1 – 12, 2002.

[3] T. Alpcan and T. Basar. A game theoretic analysis of intrusion detection in access control systems. In *proceedings of the* 43^{rd} *IEEE Conference on Decision and Control (CDC)*, volume 2, pages 1568 – 1573. IEEE, 2004.

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

[5] L. Anderegg and S. Eidenbenz. Ad hoc-VCG: A truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents. In *proceedings of the* 9^{th} *Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 245 – 259, San Diego, California, 2003. ACM.

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

[7] F. Anjum, D. Subhadrabandhu, and S. Sarkar. Signature based intrusion detection for wireless ad-hoc networks: A comparative study of various routing protocols. In *proceedings of Vehicular Technology Conference (VTC), Wireless Security Symposium*, volume 3, pages 2152– 2156. IEEE, 2003.

[8] E. N. Barron. *Game Theory: An Introduction*. Wiley-Interscience, 2007.

[9] S. Basagni. Distributed and mobility-adaptive clustering for multimedia support in multi-hop wireless networks. In *proceedings of the* 50^{th} *International Vehicular Technology Conference (VTC)*, volume 2, pages 889 – 893. IEEE, 1999.

[10] S. Basagni. Distributed clustering for ad hoc networks. In *proceedings of* 4^{th} *International Symposium on Parallel Architectures, Algorithms, and Networks*, pages 310 – 315. IEEE, 1999.

[11] M. Bechler, H. Hof, D. Kraft, F. Pahlke, and L. Wolf. A cluster-based security architecture for ad hoc networks. In *proceedings of the IEEE INFOCOM*, pages 2393 – 2403. IEEE, 2004.

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

[13] S. Bhargava and D. P. Agrawal. Security enhancements in AODV protocol for wireless ad hoc networks. volume 4, pages 2143–2147, 2001.

[14] P. Brutch and C. Ko. Challenges in intrusion detection for wireless ad-hoc networks. In *proceedings of the Symposium on Applications and the Internet Workshop*, pages 368 – 373. IEEE, 2003.

[15] S. Buchegger and J. Le Boudec. Performance analysis of the CONFIDANT protocol (cooperation of nodes - fairness in dynamic ad-hoc networks). In *proceedings of the* 3^{rd} *ACM MOBIHOC*, pages 226 – 236. ACM, 2002.

[16] L. Buttyan and J. Hubaux. Nuglets: A virtual currency to simulate cooperation in self-organized ad hoc networks. *Technical Report DSC/2001/001*, 2001.

[17] S. Capkun and J. Hubaux. Secure positioning of wireless devices with application to sensor networks. In *proceedings of the* 24^{th} *IEEE INFOCOM*, pages 1917 – 1928. IEEE, 2005.

[18] S. Carter and A. Yasinsac. Secure position aided ad hoc routing protocol. In *proceedings of the International Conference on Communications and Computer Networks*, 2002.

[19] S. Chapkin, B. Bako, F. Kargl, and E. Schoch. Location tracking attack in ad hoc networks based on topology information. In *proceedings of the IEEE International Workshop on Wireless and Sensor Networks Security*, pages 870 – 875. IEEE, 2006.

[20] K. Chen and K. Nahrstedt. iPass: An incentive compatible auction scheme to enable packet forwarding service in MANET. In *proceedings of the* 24^{th} *International*

Conference on Distributed Computing Systems (ICDCS), pages 534 – 542. IEEE, 2004.

[21] H. S. Chiu and K. S. Lui. DelPHI: Wormhole detection mechanism for ad hoc wireless networks. In *proceedings of the International Symposium on Wireless Pervasive Computing*, 2006.

[22] H. Choi, T. F. La Porta, and P. McDaniel. Privacy preserving communication in MANETs. In *proceedings of the IEEE Communications Society Conference on Sensor, Mesh, and Ad Hoc Communications and Networks*, pages 233 – 242. IEEE, 2007.

[23] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 11(1):17 – 33, 1971.

[24] V. Conitzer and T. Sandholm. Computing shapley values, manipulating value division schemes, and checking core membership in multi-issue domains. In *proceedings of the AAAI conference*, pages 219–225, 2004.

[25] B. J. Culpepper and H. C. Tseng. Sinkhole intrusion indicators in DSR MANETs. In *proceedings of the* 1^{st} *International Conference on Broadband Networks*, pages 681 – 688. IEEE, 2004.

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

[27] B. DeCleene, L. Dondeti, S. Griffin, T. Hardjono, D. Kiwior, J. Kurose, D. Towsley, S. Vasudevan, and C. Zhang. Secure group communications for wireless networks.

In *proceedings of the Military Communications Conference (MILCOM)*, volume 1, pages 113 – 117. IEEE, 2001.

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

[29] A. E.Roth. *The Shapley Value: Essays in Honor of Lloyd S.Shapley*. Cambridge University Press, 1988.

[30] A. Escudero-Pascual, T. Holleboom, and S. Fischer-Hübner. Privacy of location data in mobile networks. In *proceedings of the* 7^{th} *Nordic Workshop on Secure IT Systems (Nordsec)*, 2002.

[31] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP based mechanism for lowest-cost routing. In *proceedings of the ACM annual symposium on Principles of distributed computing*, pages 173 – 182. ACM, 2002.

[32] A. Ghosh and S. Sen. Agent-based distributed intrusion alert system. In *proceedings of the International Workshop on Distributed Computing (IWDC)*, volume 3326, pages 240 – 241. Springer, LNCS, 2004.

[33] T. Groves. Incentives in teams. *Econometrica*, 41:617 – 631, 1973.

[34] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In *proceedings of the Cryptographic Hardware and Embedded Systems (CHES)*, pages 119 – 132. Springer, LNCS, 2004.

[35] S. Gwalani, K. Srinivasan, G. Vigna, E. M. Beding-Royer, and R. Kemmerer. An intrusion detection tool for AODV-based ad hoc wireless networks. In *proceedings of the* 20^{th} *Annual Computer Security Applications Conference (CSAC)*, pages 16 – 27. IEEE, 2004.

[36] L. Hu and D. Evans. Using directional antennas to prevent wormhole attacks. In *proceedings of the Network and Distributed System Security Symposium*, 2004.

[37] Y. Hu, D. B. Johnson, and A. Perrig. SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks. In *proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'02)*, pages 3 – 13. IEEE, 2002.

[38] Y. Hu, A. Perrig, and D. B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *proceedings of* 8^{th} *Annual International Conference on Mobile Computing and Networking (MobiCom'02)*, pages 12 – 23. ACM, 2002.

[39] Y. Hu, A. Perrig, and D. B. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *proceedings of the ACM Workshop on Wireless Security (WiSe)*, pages 30 – 40. ACM, 2003.

[40] Y.C. Hu, A. Perrig, and D. B. Johnson. Wormhole attacks in wireless networks. *IEEE Journal on Selected Areas in Communications*, 24(2):370 – 380, 2006.

[41] Y. Huang and W. Lee. A cooperative intrusion detection system for ad hoc networks. In *proceedings of the* 1^{st} *Workshop on Security of Ad Hoc and Sensor Networks*, pages 135 – 147. ACM, 2003.

[42] Y. A. Huang, W. Fan, W. Lee, and P. S. Yu. Cross-feature analysis for detecting ad-hoc routing anomalies. In *proceedings of the* 23^{rd} *International Conference on Distributed Computing Systems (ICDCS)*, pages 478 – 487. IEEE, 2003.

[43] Y. A. Huang and W. Lee. Attack analysis and detection for ad hoc routing protocols. In *proceedings of the* 7^{th} *International Symposium on Recent Advances in Intrusion Detection (RAID)*, volume 3224, pages 125–145. Springer, LNCS, 2004.

[44] L. Hurwicz, E. Maskin, and R. Myerson. *Mechanism Design Theory*. The Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel, 2007.

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

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

[47] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. *Mobile Computing, Kluwer Academic Publishers*, 353(5):153–181, 1996.

[48] O. Kachirski and R. Guha. Efficient intrusion detection using multiple sensors in wireless ad hoc networks. In *proceedings of the* 36^{th} *Annual Hawaii International Conference on System Sciences (HICSS)*. IEEE, 2003.

[49] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *proceedings of the* 1^{st} *IEEE International Workshop on Sensor Network Protocols and Applications*, pages 113 – 127. IEEE, 2002.

[50] M. Khabbazian, H. Mercier, and V. K. Bhargava. Wormhole attack in wireless ad hoc networks: Analysis and countermeasure. In *proceedings of the Global Telecommunications Conference (GlobeCom)*, pages 1 – 6. IEEE, 2006.

[51] M. Kodialam and T. V. Lakshman. Detecting network intrusions via sampling: A game theoretic approach. In *proceedings of the* 22^{nd} *Annual Joint Conference of IEEE Computer and Communication Societies (INFOCOM)*, pages 1880 – 1889. IEEE, 2003.

[52] P. Krishna, N. H. Vaidya, M. Chatterjee, and D. K. Pradhan. A cluster-based approach for routing in dynamic networks. In *proceedings of the ACM SIGCOMM Computer Communication Review*, volume 27, pages 49 – 64. ACM, 1997.

[53] I. Krontiris, T. Dimitriou, T. Giannetsos, and M. Mpasoukos. Intrusion detection of sinkhole attacks in wireless sensor networks. In *proceedings of* 3^{rd} *International Workshop on Algorithmic Aspects of Wireless Sensor Networks*, pages 150 – 161. Springer, LNCS, 2008.

[54] L. Lazos and R. Poovendran. SeRLoc: Robust localization for wireless sensor networks. *ACM Transactions on Sensor Networks*, pages 73 – 100, 2005.

[55] Y. Liu, C. Comaniciu, and H. Man. A bayesian game approach for intrusion detection in wireless ad hoc networks. In *proceedings of the GameNets*. ACM, 2006.

[56] 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*, 2(4):313 – 332, 2006.

[57] R. Maheshwari, J. Gao, and S. R. Das. Detecting wormhole attacks in wireless networks using connectivity information. In *proceedings of the* 26^{th} *IEEE International Conference on Computer Communications*, pages 107 – 115. IEEE, 2007.

[58] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *proceedings of the* 6^{th} *Annual International Conference in Mobile Computing and Networking*, pages 225 – 265. ACM, 2000.

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

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

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

[62] P. Michiardi and R. Molva. Prevention of denial of service attack and selfishness in mobile ad hoc networks. *Institut Eurecom Research Report RR-02-063*, 2002.

[63] J. Mirrlees and W. Vickrey. *Incentives under Asymmetric Information*. The Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel, 1996.

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

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

[66] R. B. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, 1997.

[67] J. Nash. Non-cooperative games. *Annals of Mathematics, Second Series*, 54(2):286 – 295, 1951.

[68] E. C. H. Ngai, J. Liu, and M. R. Lyu. On the intruder detection for sinkhole attack in wireless sensor networks. In *proceedings of the IEEE International Conference on Communications*, volume 8, pages 3383 – 3389. IEEE, 2006.

[69] P. Ning and K. Sun. How to misuse AODV: A case study of insider attacks against mobile ad-hoc routing protocols. In *proceedings of the* 4^{th} *Annual IEEE Information Assurance Workshop*, pages 60 – 67. IEEE, 2003.

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

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

[72] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization Algorithms and Complexity*. Dover Publications, 1999.

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

[74] R. Poovendran and L. Lazos. A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks. *Wireless Networks*, 13(1):27 – 59, 2007.

[75] T. Ptacek and T. Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. *Secure Networks, Inc*, 1998.

[76] L. Qian, N. Song, and X. Li. Wormhole attacks detection in wireless ad hoc networks: A statistical analysis approach. *Journal of Network and Computer Applications*, 30(1):308 – 330, 2007.

[77] K. Sanzgiri, B. Dahill, B.N. Levine, C. Shields, and E. Belding-Royer. A secure routing protocol for ad hoc networks. In *proceedings of the* 10^{th} *IEEE International Conference on Network Protocols (ICNP)*, pages 78 – 87. IEEE, 2002.

[78] V. Sharma and F. Hellstrand. *Framework for Multi-Protocol Label Switching (MPLS)-based Recovery*. RFC 3469, 2003.

[79] J. Shneidman and D. Parkes. Specification faithfulness in networks with rational nodes. In *proceedings of the* 23^{rd} *Annual ACM Symposium on Principles of Distributed Computing*, pages 88 – 97, Newfoundland, Canada, 2004. ACM.

[80] A. B. Smith. An examination of an intrusion detection architecture for wireless ad hoc networks. 2001.

[81] V. Srinivasan, P. Nuggehalli, C. F. Chiasserini, and R. R. Rao. Cooperation in wireless ad hoc networks. pages 808 – 817. IEEE, 2003.

[82] D. Sterne, P. Balasubramanyam, D. Carman, B. Wilson, R. Talpade, C. Ko, R. Balupari, C. Tseng, and T. Bowen. A general cooperative intrusion detection architecture for MANETs. In *proceedings of the IEEE Workshop on Information Assurance*, pages 57 – 70. IEEE, 2005.

[83] M. Stoer and F. Wagner. A simple min-cut algorithm. *Journal of ACM (JACM)*, 44(4):585 – 591, 1997.

[84] D. Subhadrabandhu, S. Sarkar, and F. Anjum. RIDA: Robust intrusion detection in ad hoc networks. In *proceedings of the IFIP Networking Conference*, pages 1069 – 1082. Springer, LNCS, 2005.

[85] 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.

[86] B. Sun, K.Wu, and U. W. Pooch. Alert aggregation in mobile ad hoc networks. In *proceedings of the ACM Workshop on Wireless Security (WiSe'03)*, pages 69 – 78. ACM, 2003.

[87] K. Sun, P. Peng, P. Ning, and C. Wang. Secure distributed cluster formation in wireless sensor networks. In *proceedings of the* 22^{nd} *Computer Security Applications Conference (ACSAC)*, pages 131 – 140. IEEE, 2006.

[88] L. Tamilselvan and V. Sankaranarayanan. Prevention of blackhole attack in MANET. In *proceedings of the* 2^{nd} *International Conference on Wireless Broadband and Ultra Wideband Communications*, 2007.

[89] S. Vasudevan, B. DeCleene, N. Immerman, J. Kurose, and D. Towsley. Leader election algorithms for wireless ad hoc networks. In *proceedings of the* 3^{rd} *DARPA Information Survivability Conference and Exposition (DISCEX III)*, pages 261 – 272. IEEE, 2003.

[90] S. Vasudevan, J. Kurose, and D. Towsley. Design and analysis of a leader election algorithm for mobile ad hoc networks. In *proceedings of the* 12^{th} *IEEE International Conference on Network Protocols (ICNP)*, pages 350 – 360. IEEE, 2004.

[91] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16(1):8 – 37, 1961.

[92] J. von Neumann and O. Morgenstern. *The Theory of Games and Economic Behavior.* Princeton University Press, 1944.

[93] R. Wang, W. Du, and P. Ning. Containing denial-of-service attacks in broadcast authentication in sensor networks. In *proceedings of the* 8^{th} *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 71 – 79. ACM, 2007.

[94] Y. Wei, Z. Yu, and Y. Guan. Location verification algorithms for wireless sensor networks. In *proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2007.

[95] M. Willem. *Minimax Theorem*. Birkhauser, USA, 1996.

[96] X. Wu and D. K. Y. Yau. Mitigating denial-of-service attacks in MANET by distributed packet filtering: A game-theoretic approach. In *proceedings of the* 2^{nd} *ACM symposium on Information, Computer and Communications Security*, pages 365 – 367. ACM, 2007.

[97] P. Yi, Z. Dai, S. Zhang, and Y. Zhong. A new routing attack in mobile ad hoc networks. *International Journal of Information Technology*, 11(2), 2005.

[98] M. G. Zapata. Secure ad hoc on-demand distance vector (SAODV)routing. In *proceedings of the ACM SIGMOBILE Mobile Computing and Communication Review (MC2R)*, pages 106 – 107. ACM, 2002.

[99] M. G. Zapata and N. Asokan. Securing ad hoc routing protocols. In *proceedings of the ACM Workshop on Wireless Security (WiSe)*, pages 1 – 10. ACM, 2002.

[100] Y. Zhang, W. Lee, and Y. Huang. Intrusion detection techniques for mobile wireless networks. *ACM/Kluwer Wireless Networks Journal*, 9(5):545 – 556, 2003.