Evolutionary Computation Techniques for Intrusion Detection in Mobile Ad Hoc Networks

Sevil Sen

PhD Thesis
University of York
Department of Computer Science

March 2010

To my family..

Abstract

Mobile ad hoc networks (MANETs) are one of the fastest growing areas of research. By providing communications in the absence of a fixed infrastructure MANETs are an attractive technology for many applications. However, this flexibility introduces new security threats. Furthermore the traditional way of protecting networks is not directy applicable to MANETs. Many conventional security solutions are ineffective and inefficient for the highly dynamic and resource-constrained environments where MANET use might be expected.

Since prevention techniques are never enough, intrusion detection systems (IDSs), which monitor system activities and detect intrusions, are generally used to complement other security mechanisms. How to detect intrusions effectively and efficiently on this highly dynamic, distributed and resource-constrained environment is a challenging research problem. In the presence of these complicating factors humans are not particularly adept at making good design choices. That is the reason we propose to use techniques from artificial intelligence to help with this task.

We investigate the use of evolutionary computation techniques for synthesising intrusion detection programs on MANETs. We evolve programs to detect the following attacks against MANETs: ad hoc flooding, route disruption, and dropping attacks. The performance of evolved programs is evaluated on simulated networks. The results are also compared with hand-coded programs. A good IDS on MANETs should also consider the resource constraints of the MANET environments. Power is one of the critical resources. Therefore we apply multi-objective optimization techniques (MOO) to discover trade-offs between intrusion detection ability and energy consumption of programs, and optimise these objectives simultaneously.

We also investigate a suitable IDS architecture for MANETs in this thesis. Different programs are evolved for two architectures: local and cooperative detection in neighbourhood. Optimal trade-offs between intrusion detection ability and resource usage (energy, bandwidth) of evolved programs are also discovered using MOO techniques.

Contents

List of Figures

List of Tables

Preface

“Idle reader: thou mayest believe me without any oath that I would this book, as it is the child of my brain, were the fairest, gayest, and cleverest that could be imagined. But I could not counteract Nature’s law that everything shall beget its like; and what, then, could this sterile, ill-tilled wit of mine beget but the story of a dry, shrivelled, whimsical offspring, full of thoughts of all sorts and such as never came into any other imagination–just what might be begotten in a prison, where every misery is lodged and every doleful sound makes its dwelling?...”

Miguel de Cervantes

Acknowledgements

Firstly I would like to thank my supervisor, John A. Clark for all his support and advice throughout. I would also like to thank Daniel Kudenko and Pascal Bouvry for their valuable feedback. Thanks to Juan for all the help and his friendship.

I would like to thank David for his great company and friendship. I would like to thank Kamran and Yow Tzu for their support and friendship. Thanks to Simon, Chen Hao, Saman, Shaheen, and Arturo for their help. I would also like to thanks to Paul and all my office mates who I enjoyed sharing an office with.

I would like to thank Hacettepe University who have supported this research.

I would like to thank my parents Seher and Şendoğan, my sister Filiz, and my brother Tolga for all their love and support throughout my life. Huge thanks to Erdem who has always been there for me along the way.

Declaration

The work presented in this thesis has been drawn from research undertaken between October 2006 and March 2010 at the Department of Computer Science, University of York. Much of the work has been published elsewhere as follows :

- S. Sen, J.A. Clark, J.E. Tapiador. Security Threats in Mobile Ad Hoc Networks, Security of Self-Organizing Networks: MANET, WSN, WMN, VANET, Chapter 5, Auerbach Publications, CRC Press, USA, 2010 (to appear).
- S. Sen, J.A. Clark, J.E. Tapiador. Power-Aware Intrusion Detection on Mobile Ad Hoc Networks, In Proceedings of Ad Hoc Networks (AdhocNets'09), LNICST 28, pp. 224-239, Springer, 2009.
- S. Sen, J.A. Clark. A Grammatical Evolution Approach to Intrusion Detection on Mobile Ad Hoc Networks, In Proceedings of ACM Wireless Security (WiSec'09), pp. 95-102, ACM, 2009.
- S. Sen, J.A. Clark, Intrusion Detection in Mobile Ad Hoc Networks, Guide to Wireless Ad Hoc Networks, Chapter 17, pp. 1-28, 2009.
- S. Sen, J.A. Clark. Evolving Intrusion Detection Rules on Mobile Ad Hoc Networks, In Proceedings of Pacific Rim International Conferences on Artificial Intelligence (PRICAI'08), LNAI 5351, pp. 1053-1058, Springer, 2008.
- J.A. Clark, J. Murdoch, J.A. McDermid, S. Sen, H.R. Chivers, O. Worthington, P. Rohatgi. Threat Modelling for Mobile Ad Hoc and Sensor Networks, In Proceedings of Annual Conference of ITA, 2007.

I declare that the work in this thesis is original work I undertook between the dates of registration for the Degree of Doctor of Philosophy at the University of York.

I was primary author for all work reported in this thesis and in the papers above. Advice is provided by my supervisor, John A. Clark, and also by Juan Tapiador. For the final paper I was a contributing author.

Chapter 1

Introduction

1.1 Mobile Ad Hoc Networks (MANETs)

Mobile ad hoc networks (MANETs) are one of the fastest growing areas of research. This new type of self-organizing network combines wireless communication with a high degree node mobility. They do not have any fixed infrastructure such as base stations or centralized management points as in conventional networks. The nodes cooperate with each other to provide basic functionality such as routing in a network, independent of any fixed infrastructure or centralized management. This flexibility makes them attractive for many applications. They are especially suited to military applications where the network topology may change rapidly to reflect a force's operational movements, and disaster recovery operations where the existing/fixed infrastructure may be non-operational due to a natural disaster, war, and the like. Virtual conferences, where setting up a network infrastructure is a time consuming high-cost task, are another promising area of use.

MANETs have different properties than conventional networks and present new vulnerabilities. They also share the vulnerabilities of wired networks, such as eavesdropping, denial of service, spoofing and the like; these are simply accentuated by the ad hoc context [58]. First of all, the use of wireless links make them susceptible to many attacks such as active interference and eavesdropping. Unlike wired networks, attackers do not need physical access to the network to carry out these attacks. Secondly, the dynamic topology of MANETs makes it harder to differentiate normal behaviour of the network from anomalous behaviour. Another vulnerability is the use of cooperative algorithms to meet the basic network functions. Routing algorithms for MANETs usually assume that nodes are cooperative and non-malicious. Hence a malicious node can easily become an important routing agent and disrupt network operations by disobeying the protocol specifications. Resource-constraints are a further vulnerability. Devices on MANETs can vary from laptops to handheld devices (*e.g.* PDAs, mobile phones) and may exhibit a wide range of computing and storage capabilities. They are also generally dependent on battery power to provide mobility. This has led to the emergence of new attacks targeting this aspect.

To summarize, the flexibility provided by the open broadcast medium and the cooperativeness of the mobile devices (which have generally different resource and computational capacities, and run usually on battery power) introduce new security risks for MANETs. As part of rational risk management we must be able to identify these risks and take appropriate action. In some cases, we may prevent these risks cost-effectively. In other cases we may have to accept that vulnerabilities exist and seek to take appropriate action when we believe someone is attacking us. That's why intrusion detection systems (IDSs) which monitor system activities and detect anomalies, are usually used to complement other security mechanisms. Intrusion detection on MANETs is the main focus of this research.

1.2 Intrusion Detection in MANETs

Intrusion is any set of actions that attempts to compromise the integrity, confidentiality, or availability of a resource [29] and an intrusion detection system (IDS) is a system for the detection of such intrusions. It detects possible violations of a security policy by monitoring system activities and responding to those that are apparently intrusive. Since prevention techniques cannot be sufficient and new intrusions continually emerge, IDS is an indispensable part of a security system. IDS detects possible violations of a security policy by monitoring system activities and response. If we detect an attack once it comes into the network, a response can be initiated to prevent or minimize the damage to the system.

The specific features of MANETs present a challenge for security solutions. Even though there have been many approaches proposed for intrusion detection for wired networks in the literature, they do not find simple application to MANETs. The traditional way of detecting attacks at the traffic concentration points is no longer suitable for this distributed environment. Furthermore many existing solutions for conventional networks are ineffective and inefficient for this-resource constrained environment. There are new issues that should be taken into account while designing an IDS for MANETs. The dynamic nature of MANETs, the lack of central points, and their highly constrained nodes are the main challenges which make applying existing solutions impractical. Consequently researchers have been working on developing new IDSs for MANETs and adapting existing ones for the last decade.

There are three main intrusion detection techniques employed in the literature:

anomaly-based, misuse-based, and specification based. All intrusion techniques have their own strengths and weaknesses. That is the reason researhers often employ different techniques together for an effective intrusion detection. One of the most commonly proposed intrusion detection techniques in MANETs is specification-based intrusion detection, where intrusions are detected as runtime violations of the specifications of routing protocols. This technique has been applied to a variety of routing protocols in MANETs. However it cannot detect DoS (denial of service) attacks. Anomaly-based techniques profile the symptoms of normal behaviours of the system and detect intrusions as deviations from the normal behaviour patterns. Various researchers have sought to apply anomaly-based approaches to MANETs. The biggest challenge is defining normal behaviour in this technique. Normal behaviour can change over time and IDS systems need to adapt accordingly, otherwise the system may exhibit a high false positive rate. On the other hand, it is capable of detecting unknown attacks. This is important in a new environment such as MANETs where new attacks and new vulnerabilities of systems could be announced constantly. Misuse-based IDS compares known attack signatures with current system activities. The drawback of this approach is that it cannot detect new attacks. Although this technique has been preferred by many commercial IDSs in the literature due to its efficiency and its low false positive rate, there has been little research on signatures of new attacks against MANETs. Few misuse-based IDSs have been proposed so far for MANETs. In this thesis this issue has been addressed by using techniques from artificial intelligence to find intrusion detection rules automatically.

1.3 Thesis Hypothesis

MANETs are a new type of distributed network whose properties are complex and ill-understood. Humans might not be the best choice to design an IDS for this new environment. Accordingly we propose to investigate the use of artificial intelligence based learning techniques to explore this design space more efficiently than a human could.

Evolutionary computation has already showed considerable promise for creating IDS components for conventional networks, but has seen very little application in the MANET domain. Accordingly, we propose to investigate its potential in the new area. The main hypothesis of this research is given as follows:

Hypothesis 1: *Evolutionary Computation will be able to discover complex*

> *properties of mobile ad hoc networks and evolve intrusion detection programs suitable for this new environment. Programs evolved using Genetic Programming and Grammatical Evolution techniques will be able to detect specific routing attacks on mobile ad hoc networks (namely ad hoc flooding, route disruption, and dropping attacks) effectively.*

Efficiency is as important as effectiveness for intrusion detection in mobile networks. Resource-constrained nodes on MANETs require different trade-offs to be made between intrusion detection ability of programs and their resource usage. Humans are not particularly adept at selecting good choices when complex trade-offs have to be made. In this research a multi-objective evolutionary algorithm, which allows us to optimize multiple objectives simultaneously, is employed to detect programs both effectively (*i.e.* detect intrusions without a high false positive rate) but also efficiently (*i.e* is power-aware). Furthermore a suitable intrusion detection architecture for MANETs is investigated. Distributed and cooperative intrusion detection programs which take into account their resource usage (energy, bandwidth) beside their intrusion detection ability are evolved by using multi-objective evolutionary computation. The second hypothesis of this thesis is given below:

> ***Hypothesis 2***: *Multi Objective Evolutionary Computation will allow us to discover trade-offs between functional (intrusion detection ability) and non-functional (power and bandwidth usage) properties of MANET intrusion detection programs.*

As stated earlier different characteristics of MANETs should be considered while designing a suitable intrusion detection system for these new type of networks. The approach proposed in this research takes into account the specific nature of MANETs as follows:

- *dynamic topology*: features reflecting different mobility level of networks have been considered while designing an intrusion detection system for MANETs.
- *limited resources*: trade-offs between intrusion detection ability and energy usage of evolved programs are discovered. (Power is the critical resource in mobile nodes.)
- *lack of concentration points*: distributed and cooperative intrusion detection programs are evolved in order to detect network attacks more effectively. Furthermore they do so in a resource-efficient way (*i.e.* with limited bandwidth and energy consumption).

1.4 Thesis Overview

The remainder of the thesis is organized as follows.

Chapter 2 introduces mobile ad hoc networks and their specific characteristics. This new networking is by its very nature more vulnerable to attacks than wired networks. Furthermore, existing security solutions proposed for conventional networks are not effective and efficient for this new environment. Researchers have been working on new solutions or adapting existings ones to MANETs. However we must understand the vulnerabilities and the attacks against MANETs in order to propose suitable security solutions for them. This is the main aim of this chapter. The specific vulnerabilities of MANETs are described and a detailed classification of the attacks/attackers against these complex distributed systems is presented.

Chapter 3 starts with an introduction to intrusion detection. Intrusion detection in MANETs is the main focus of this thesis and is a complex and difficult task. This is mainly due to the dynamic nature of MANETs, their highly constrained nodes, and the lack of central monitoring points. This chapter outlines issues of intrusion detection for MANETs and reviews the main solutions proposed in the literature.

This thesis investigates the use of evolutionary computation techniques to develop intrusion detection in MANETs. Chapter 4 introduces evolutionary computation and two evolutionary computation techniques employed in this research: genetic programming and grammatical evolution. Why these techniques are chosen in this research is also discussed. The specific attacks targetted by our work (ad hoc flooding, route disruption, dropping) are described and the overall approach to using evolutionary computation is defined.

Chapter 5 presents the evaluation results. The ability of evolved programs using genetic programming and grammatical evolution to detect known attacks is shown on simulated networks with varying mobility traffic patterns. The results are also compared with hand-coded programs. Moreover the two techniques are compared fairly using a design of experiments methodology and the comparison results are presented in this chapter.

Power is a significant constraint for many MANET nodes. Therefore the efficiency of evolved programs are as important as their effectiveness. Since power is one of the most critical resources in MANETs, the work decribed in Chapter 6 aims to evolve intrusion detection programs optimizing their energy usage as well. In order to discover trade-offs between classification accuracy and energy consumption of evolved programs, a multi-objective evolutionary technique is employed. The technique is also introduced.

MANETs do not have concentration points as in wired networks where we can monitor the network traffic. The network data is distributed to all nodes. Therefore a distributed and cooperative intrusion detection architecture would appear to be worthy of serious consideration for MANETs. The research presented in Chapter 7 investigates this idea. An architecture *cooperative detection in neighbourhood* and how to choose monitoring nodes in this architecture in terms of energy and bandwidth usage are investigated. Chapters 6 and 7 provide evidence to support *hypothesis 2*.

Chapter 8 concludes the thesis. The suitability of the proposed approach to MANETs is discussed and the thesis hypotheses are evaluated.

Chapter 2

Security in Mobile Ad Hoc Networks

This chapter starts with an introduction to a new type of networking, mobile ad hoc networks (MANETs). In order to develop suitable security solutions for this new environment, we must first understand how MANETs can be attacked. This chapter provides a comprehensive survey of attacks against a specific type of target, namely the routing protocols used by MANETs. The security issues specific to this environment are introduced in Section 2.2 and a detailed classification of the attacks and attackers against these complex distributed systems is presented in Section 2.3.

2.1 Mobile Ad Hoc Networks

A mobile ad hoc network (MANET) is a self-configuring network of mobile nodes connected by wireless links. MANETs do not have any fixed and pre-established infrastructure such as centralized management or base stations in wireless networks. The union of nodes forms an arbitrary network topology. However the network topology changes frequently because the nodes can move, leave, and join the network randomly due to mobility.

Conventional networks use dedicated nodes to carry out basic functions like packet forwarding, routing, and network management. In ad hoc networks these are carried out collaboratively by all available nodes. Nodes on MANETs use multi-hop communication: nodes that are within each other's radio range can communicate directly via wireless links, while those that are far apart must rely on intermediate nodes to act as routers to relay messages. So every node acts both as router and host in MANETs.

Mobile nodes can move, leave, and join the network and routes need to be updated frequently due to the dynamic network topology. For example, node S can communicate with node D by using the shortest path S-A-B-D as shown in Figure 2.1 (the dashed lines show the direct links between the nodes). If node A moves out of node S's range, he has to find an alternative route to node D (S-C-E-B-D). A routing protocol in such a network is responsible for finding routes and providing communication between end points through cooperating intermediate nodes.

A variety of new protocols have been developed for finding/updating routes and generally providing communication between end points, but no proposed protocol has been accepted as standard yet. There are two kinds of routing protocols on MANETs: proactive and reactive protocols. Proactive routing protocols such as OLSR [46] use periodic exchange of control messages between nodes to build up a routing table. In contrast, reactive routing protocols such as AODV [73] and DSR [47] discover routes when they are needed. There are also hybrid approaches which combine both proactive and reactive approaches. For example, ZRP uses a proactive approach for communication with neighbouring nodes, while it uses a reactive approach for communication with other nodes [35]. These new routing protocols, based on cooperation between nodes, are vulnerable to new forms of attacks. Unfortunately, many proposed routing protocols for MANETs do not consider security, but there are some secure routing protocols proposed in the literature. They take into account active attacks that aim at intentionally tampering with the execution of routing protocols, but do not address passive attacks and selfishness [65].

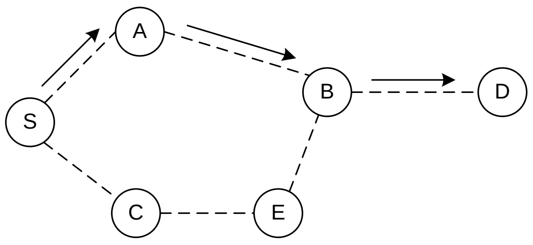


Figure 2.1: Communication between Nodes on MANETs

MANETs of various forms have emerged in recent years, supporting the increasing usage of mobile devices such as PDAs, mobile phones, and laptops. Since these devices are getting smaller, cheaper and more powerful, they are becoming increasingly popular. Moreover, the flexibility of MANETs makes them attractive for many applications such as military applications, where the network topology may change rapidly to reflect a force's operational movements, and disaster recovery operations, where the existing/fixed infrastructure may be non-operational. The ad hoc self-organisation also makes them suitable for virtual conferences, where setting up a traditional network infrastructure is a time consuming, high-cost task. The main applications of MANETs are military applications in which aircrafts, tanks and moving personnel and the like can communicate at peace or in war time.

2.2 Vulnerabilities of MANETs

If we are to develop security solutions for MANETs we must first have a comprehensive understanding of their possible vulnerabilities and security risks. They share the vulnerabilities of wired networks, such as eavesdropping, denial of service, spoofing and the like, which are accentuated by the ad hoc context [58]. They have further vulnerabilities, such as those that take advantage of the cooperative nature of routing algorithms. These vulnerabilities of MANETs are summarized below.

Wireless Links: First of all, the use of wireless links makes the network susceptible to attacks such as eavesdropping and active interference. Unlike wired networks, attackers do not need physical access to the network to carry out these attacks. Furthermore wireless networks typically have lower bandwidths than wired networks. Attackers can exploit this feature, consuming network bandwidth with ease to prevent normal communication among nodes. However some solutions to protect the radio interface from attacks such as eavesdropping and jamming attacks have been proposed in the literature: spread spectrum communication, frequency hopping, and the like [45].

Dynamic Topology: MANET nodes can leave and join the network, and move independently. As a result the network topology can change frequently. It is hard to differentiate normal behaviour of the network from anomalous behaviour in this dynamic environment. For example, a node sending disruptive routing information can be a malicious node, or else simply be using outdated information in good faith, or a node who does not collaborate with other nodes can be a malicious node or a failed node unable to perform due to power failure or other environmental factors. Moreover mobility of nodes means that we cannot assume nodes, especially critical ones (servers, etc.), are secured in locked cabinets as in wired networks. Nodes with inadequate physical protection may often be at risk of being captured and compromised.

Cooperativeness: Routing algorithms for MANETs usually assume that nodes are cooperative and non-malicious. As a result, a malicious attacker can easily become an important routing agent and disrupt network operations by disobeying the protocol specifications. For example, a node can pose as a neighbour to other nodes and participate in collective decision-making mechanisms, possibly affecting networking significantly.

Lack of a Clear Line of Defence: MANETs do not have a clear line of defence; attacks can come from all directions [106]. The boundary that separates the inside network from the outside world is not very clear on MANETs. For example, there is no well defined place where we can deploy our traffic monitoring and access control mechanisms. Whereas all traffic goes through switches, routers, or gateways in wired networks, network information in MANETs is distributed across nodes that can only see the packets sent and received in their transmission range. Unlike wired networks, attackers do not need to gain physical access to the network to exploit some kinds of attacks such as passive eavesdropping and active interference (these require only radio contact) [106].

Limited Resources: Resource constraints are a further vulnerability. There can be a variety of devices on MANETs, ranging from laptops to handheld devices such as PDAs and mobile phones. These will generally have different computing and storage capacities that can be the focus of new attacks. For example, mobile nodes generally run on battery power. This has led to emergence of innovative attacks targeting this aspect, *e.g.* "Sleep Deprivation Torture [86]". Furthermore, the introduction of more security features into the network increases the computation, communication and management load [102]. This is a challenge for networks that are already resource-constrained.

2.3 Attacks on MANETs

At the highest level, the security goals of MANETs are not that different from other networks: most typically authentication, confidentiality, integrity, availability, and non-repudiation. *Authentication* is the verification of claims about the identity of a source of information. *Confidentiality* means that only authorized people or systems can read or execute protected data or programs. It should be noted that the sensitivity of information in MANETs may decay much more rapidly than in other information systems. For example, yesterday's troop location will typically be less sensitive than today's. *Integrity* means that the information is not modified or corrupted by unauthorized users or by the environment. *Availability* refers to the ability of the network to provide services as required. Denial of Service (DoS) attacks have become one of the most worrying problems for network managers. In a military environment, a successful DoS attack is extremely dangerous, and the engineering of such attacks is a valid modern war-goal. Lastly, *non-repudiation* ensures that committed actions cannot be denied. In MANETs security goals of a system can change in different modes (*e.g.*

peace time, transition to war, and war time of a military network).

The characteristics of MANETs make them susceptible to many new attacks. At the top level attacks can be classified according to network protocol stacks. Table 2.1 gives a few examples of attacks at each layer. Some type of attacks could occur in any layer of the network protocol stack, *e.g.* jamming at physical layer, hello flood at network layer, and SYN flood at transport layer are all DoS attacks. Because new routing protocols introduce new forms of attacks on MANETs, we mainly focus on network layer (routing) attacks in this chapter.

Layer	**Attacks**
Application	data corruption, viruses and worms
Transport	TCP/UDP SYN flood
Routing	hello flood, blackhole
Data Link	monitoring, traffic analysis
Physical	eavesdropping, active interference

Table 2.1: Some Attacks on the Protocol Stack

2.3.1 Adversary Model

Attackers against a network can be classified into two groups: insider and outsider attackers. Whereas an outsider attacker is not a legitimate user of the network, an insider attacker is an authorized node and a part of the routing mechanism on MANETs. Routing algorithms are typically distributed and cooperative in nature and affect the whole system. While an insider MANET node can disrupt the network communications intentionally, there might be other reasons for its apparent misbehaviours. A node can be *failed*, unable to perform its function for some reason, such as running out of battery, or collisions in the network. The threat of failed nodes is particularly serious if they are needed as part of an emergency/secure route [103]. Their failure can even result in partitioning of the network, preventing some nodes from communicating with other nodes in the network. A *selfish* node can also misbehave to preserve its resources. Selfish nodes avail themselves of the services of the other nodes, but do not reciprocate. This research focuses on the attacks carried out by *malicious* nodes who intentionally aim to disrupt the network communication.

The misuse goals of attackers should also be considered in threat modelling. In routing attacks attackers do not follow the specifications of routing protocols and aim to disrupt the network communication in the following ways:

- *Route Disruption*: modifying existing routes, creating routing loops, and causing packets to be forwarded along a route that is not optimal, non-existent, or otherwise erroneous.

- *Node Isolation*: isolating a node or some nodes(s) from communicating with other nodes in the network, partitioning the network, etc.

- *Resource Consumption*: decreasing network performance, consuming network bandwidth or node resources, etc.

Ning *et al.* consider each of these goals in their research which analyses insider attacks against AODV [67]. Achieving these goals depends on the capabilities of the adversary. The main factors affecting the performance of an attack are identified below.

Computational power: This clearly affects the ability of an attacker to compromise a network. Such power need not be localised to the attached network –eavesdropped traffic can be relayed back to high performance super-computing networks for analysis.

Deployment capability: Adversary distribution may range form a single node to a pervasive carpet of smart counter-dust, with a consequent variation in attack capabilities [26]. This sort of distinction may affect the ability to eavesdrop, to jam a network effectively, and to escape destruction (*e.g.* a single powerful jammer can easily be taken out, distributed jamming is harder to extinguish).

Location control: The location of adversary nodes has may have a clear impact on what the adversary can do. An adversary may be restricted to placing attack nodes at the geographical boundary of an enemy network (but may otherwise choose the precise locations), may plant specific nodes (*e.g.* nodes left behind in territory about to be vacated), or may have the ability post facto to create a pervasive carpet of smart dust (where arbitrary degrees of pervasiveness may be achieved).

Mobility: Mobility generally brings an increase in power. (A mobile node can always remain stationary.) On the other hand, mobility may prevent an attacker from continually targeting one specific victim. For example, a node on the move might not receive all falsified routing packets initiated by the attacker. In [89] this phenomenon is defined as being a “partial victim”. Moreover they have stated that even if it reduces the damage caused by the attacker, it makes detection more difficult since the symptoms of an attack and those arising due to the dynamic nature of the network are difficult to distinguish. In conclusion, the impact of mobility on detection is a complex matter.

Degree of physical access (including node capture ability and ability to carry out physical deconstruction).

Given the agile nature of MANETs determining an applicable adversary model is difficult. However, systems can be evaluated against a range of representative threat models.

2.3.2 Attacks

We can classify attacks as passive or active at the top level.

Passive attacks: In a passive attack an unauthorized node monitors and aims to find out information about the network. The attackers do not otherwise need to communicate with the network. Hence they do not disrupt communications or cause any direct damage to the network. However, they can be used to get information for future harmful attacks. Examples of passive attacks are eavesdropping and traffic analysis.

Eavesdropping Attacks, also known as disclosure attacks, are passive attacks by external or internal nodes. The attacker can analyse broadcast messages to reveal some useful information about the network. Solutions protecting the radio interface from attacks such as eavesdropping (and jamming) attacks have been proposed in the literature, *e.g.* spread spectrum communication and frequency hopping [45].

Traffic Analysis is not necessarily an entirely passive activity. It is perfectly feasible to engage in protocols, or seek to provoke communication between nodes. Attackers may employ techniques such as RF direction finding, traffic rate analysis, and time-correlation monitoring. For example, by timing analysis it can be revealed that two packets in and out of an explicit forwarding node at time t and t+ε are likely to be from the same packet flow [54]. Traffic analysis in ad hoc networks may reveal:

- the existence and location of nodes;
- the communications network topology;
- the roles played by nodes;
- the current sources and destination of communications; and

- the current location of specific individuals or functions (*e.g.* if the commander issues a daily briefing at 10am, traffic analysis may reveal a source geographic location).

Active Attacks: These attacks cause unauthorised state changes in the network such as denial of service, modification of packets, and the like. These attacks are generally launched by users or nodes with authorisation to operate within the network. We classify active attacks into four groups: dropping, modification, fabrication, and timing attacks. It should be noted that an attack can be classified into more than one group.

Dropping Attacks: Malicious or selfish nodes deliberately drop all packets that are not destined for them. While malicious nodes aim to disrupt the network connection, selfish nodes aim to preserve their resources. Dropping attacks can prevent end-to-end communications between nodes if the dropping node is at a critical point. It might also reduce the network performance by causing data packets to be retransmitted, new routes to the destination to be discovered, and the like.

Unfortunately most routing protocols (DSR is an exception [103]) have no mechanism to detect whether data packets have been forwarded or not by intermediate nodes. However, attacks against a node can be detected by his neighbouring nodes through passive acknowledgement or hop-by-hop acknowledgement at the data link layer.

An attacker can choose to drop only some packets to avoid being detected; this is called a *selective dropping attack*. Besides data packets or route discovery packets, an attacker can also drop route error packets, causing the source node to be unaware of failed links (thus interfering with the discovery of alternative routes to the destination).

Modification Attacks: Insider attackers modify packets to disrupt the network. For example, in the *sinkhole attack* the attacker tries to attract nearly all traffic from a particular area through a compromised node by making the compromised node attractive to other nodes. It is especially effective in routing protocols that use advertised information such as remaining energy and nearest node to the destination in the route discovery process. A sinkhole attack can be used as a basis for further attacks like dropping and selective forwarding attacks. A black hole attack is like a sinkhole attack that attracts traffic through itself and uses it as the basis for further attacks. The goal is to prevent packets being forwarded to the destination. If the black hole is a virtual node or a node outside the network, it is hard to detect [22].

Fabrication Attacks: Here the attacker forges network packets. In [67], fabrication attacks are classified into "active forge" in which attackers send faked messages without receiving any related message and "forge reply" in which the attacker sends fake route reply messages in response to related legitimate route request messages.

In the forge reply attack, the attacker forges a Route Reply (RREP) message after receiving a Route Request message. The reply message contains falsified routing information showing that the node has a fresh route to the destination node on AODV in order to suppress real routes to the destination. It causes route disruption by causing messages to be sent to a non-existent node or putting the attacker itself into the route between two endpoints of a communication channel if the insider attacker has genuinely a route to the destination. Figure 2.2 shows an example of a forge reply attack defined in [67]. The best route (with minimum hops) from node S to node D is S-I1-I2-D. Malicious node M forges a Route Reply message to the source node S through node I1. The message claims to come from the destination node D with higher destination sequence number to suppress the existing route. The faked message results in the updating of the route entry to the destination node in the routing tables of node S and I1. Node I1 forwards data packets to the malicious node instead of node I2 since node M seems to have a fresh route to node D, so the new route becomes S-I1-M-I2-D (it is not the optimal route).

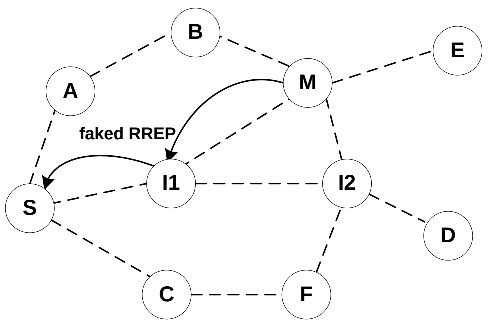


Figure 2.2: Forge Reply Attack

Attackers can initiate frequent packets to cause denial of service (DoS). Example DoS attacks that exploit MANETs' features are sleep deprivation torture attacks, routing table overflow attacks, ad hoc flooding attacks, rushing attacks, and the like. The *sleep deprivation torture attack* consumes a node's battery power and so disables the node. It does so by persistently making service requests of one form or another. This attack was introduced by Stajano *et al.* [86] who emphasized that it is more powerful than better known DoS attacks such as CPU exhaustion, since most mobile nodes are run on battery power. The *ad hoc flooding attack*, introduced in [104], is another DoS attack against on-demand protocols, in which nodes send Route Request messages when they need a route. The attacker exploits this property of Route Discovery by broadcasting many Route Request messages for a destination node that is not in the network. Another attack at the Route Discovery phase is the *routing table overflow attack*. Here the attacker sends a lot of route advertisements for nodes that do not exist. Since proactive protocols update routing information periodically before it is needed, this attack, which results in overflowing the victim nodes' routing tables and preventing new routes from being created, is more effective in proactive protocols than in reactive protocols [101].

Another interesting fabrication attack on MANETs is the *routing cache poisoning attack* [101]. A node can update its table with the routing information in the packets that it hears, even if it is not on the route of the packets. The attacker can make use of this property to poison the routes to a victim node by sending spoofed routing information packets, causing neighbouring nodes to update their tables erroneously.

Timing Attacks: An attacker attracts other nodes by causing itself to appear closer to those nodes than it really is. DoS attacks, rushing attacks, and hello flood attacks use this technique. *Rushing attacks* [41] occur during the Route Discovery phase. In all existing on-demand protocols, a node needing a route broadcasts Route Request messages and each node forwards only the first arriving Route Request in order to limit the overhead of message flooding. So, if the Route Request forwarded by the attacker arrives first at the destination, routes including the attacker will be discovered instead of valid routes. Rushing attacks can be carried out in many ways: by ignoring delays at MAC or routing layers, by wormhole attacks, by keeping other nodes' transmission queues full, or by transmitting packets at a higher wireless transmission power [41]. The *hello flood attack* [50] is another attack that makes the adversary attractive for many routes. In some routing protocols, nodes broadcast Hello packets to detect neighbouring nodes. These messages are received by all one-hop neighbour nodes, but are not forwarded to further nodes. The attacker broadcasts many Hello packets with large enough transmission power that each node receiving Hello packets assumes the adversary node to be its neighbour. It can be highly effective in both proactive and reactive MANET protocols.

A further significant attack on MANETs is the collaborative *wormhole attack*. Here an attacker receives packets at one point in the network, tunnels them to an attacker at another point in the network, and then replays them into the network from this final point [39]. Packets sent by tunneling forestall packets forwarded by multi-hop routes as shown in Figure 2.3 and it gives the attacker nodes an advantage for future attacks. Since the packets sent over tunneling are the same as the packets sent by normal nodes, it is generally difficult to detect wormhole attackers by software-only approaches such as IDS [39]. That is why packet leashes (any information that is added to a packet designed to restrict the packet' s maximum allowed transmission distance [39]) have been introduced for preventing wormhole attacks.

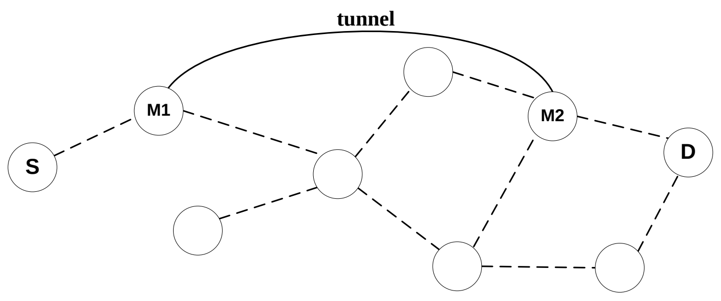


Figure 2.3: Wormhole Attack

The very nature of MANETs renders them open to many attacks. The benefits of significant flexibility come at a price. Many of the attacks described above could be avoided by including authentication techniques in the routing protocol [40][80][16]. The main idea here is to guarantee that all nodes wishing to participate in the routing process are authenticated nodes; *i.e.*, trusted network elements that will behave according to the protocol rules. Authentication should be enforced during all routing phases, thus preventing unauthorised nodes (including attackers) from participating in the routing and so from launching routing attacks. Authentication can be provided based either on public-key or symmetric cryptography.

The use of cryptography comes in hand with an associated problem: the necessity of a mechanism for issuing, exchanging, and revoking keys. Key management in MANETs is generally more difficult than in classical wired networks due to the absence of any infrastructure or central administrative authorities. There is no obvious point(s) where services such as certification authorities (CA) or key servers (KS) can be placed. Furthermore cryptography and authentication are expensive tasks in terms of resource usage in a mobile wireless environment.

Although we may put in place mechanisms to prevent particular types of compromise, we will always be open to others. In general we prevent compromise where we can (proactive solutions), but then seek to detect and deal with it when prevention does not work (reactive solutions). In this thesis we focus on the detection part of the security process. How security compromises on MANETs could be detected will be presented in the subsequent chapter.

Chapter 3

Intrusion Detection in Mobile Ad Hoc Networks

This chapter introduces intrusion detection systems and gives a brief summary of the research done in this area in Section 3.1. MANETs have different properties than conventional networks and introduce new issues for security solutions. The issues of intrusion detection in MANETs are outlined in Section 3.2. The main solutions proposed in the literature are reviewed in Section 3.3.

3.1 Intrusion Detection Systems (IDS)

Intrusion is any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource [38] and an intrusion detection system (IDS) is a system for the detection of such intrusions. The development of an IDS is motivated by the following factors:

- Most existing systems have security flaws that render them susceptible to intrusions, and finding and fixing all these deficiencies are not feasible [29].
- Prevention techniques cannot be sufficient. It is almost impossible to have an absolutely secure system [29].
- Even the most secure systems are vulnerable to insider attacks [29].
- New intrusions continually emerge and new techniques are needed to defend against them.

Since there are always new intrusions that cannot be prevented, IDS is introduced to detect possible violations of a security policy by monitoring system activities and response. IDSs are aptly called the second line of defence, since IDS comes into the picture after an intrusion has occurred. If we detect the attack once it comes into the network, a response can be initiated to prevent or minimize the damage to the system. It also helps prevention techniques improve by providing information about intrusion techniques.

3.1.1 Taxonomy of Intrusion Detection Systems

There are three main components of an IDS: data collection, detection, and response. The data collection component is responsible for collection and pre-processing data tasks: transferring data to a common format, data storage, and sending data to the detection module [60].

IDS can use different data sources which are the inputs to the system: system logs, network packets, etc. If an IDS monitors activities on a host and detects violations on the host, it is called host-based IDS (HIDS). An IDS that monitors network packets and detects network attacks is called network-based IDS (NIDS). NIDSs generally listen in promiscuous mode to the packets in a segment of the network, allowing them to detect distributed attacks. There are also intrusion detection systems that use both host-based IDS and network-based IDS. For example, a system can use NIDS and also HIDS for important hosts in the networks such as servers, databases, and the like. Since NIDS cannot monitor encrypted packets, a hybrid approach, network node IDS (NNIDS) is introduced where each host in the network has NNIDS to monitor network packets directed to the host [52].

There are both centralized and distributed IDSs (DIDS) in the literature. In [75] there is a survey of current distributed IDSs which shows that most of the DIDSs are hierarchically organized around a central node and few of them are completely distributed. Generally, only data collection is distributed in DIDSs.

In the detection component data is analyzed to detect intrusion attempts and indications of detected intrusions are sent to the response component. In the literature, three intrusion detection techniques are used. The first technique is anomaly-based intrusion detection which profiles the symptoms of normal behaviors of the system such as usage frequency of commands, CPU usage for programs, and the like. It detects intrusions as anomalies, *i.e.* deviations from the normal behaviours. Various techniques have been applied for anomaly detection such as classification based (*e.g.* neural networks, support vector machines), nearest neighbour based, clustering based and statistical techniques [24]. Defining normal behaviour is a major challenge. Normal behavior can change over time and intrusion detection systems must be kept up to date. False positives – the normal activities which are detected as anomalies by IDS – can be high in anomaly-based detection. On the other hand, it is capable of detecting previously unknown attacks. This is very important in an environment where new attacks and new vulnerabilities of systems are announced constantly.

Misuse-based intrusion detection compares known attack signatures with current system activities. It is generally preferred by commercial IDSs since it is efficient and has a low false positive rate. The drawback of this approach is that it cannot detect new attacks. The system is only as strong as its signature database and this needs frequent updating for new attacks. Both anomaly-based and misuse-based approaches have their strengths and weaknesses. Therefore, both techniques are generally employed for effective intrusion detection.

The last technique is specification-based intrusion detection. In this approach, a set of constraints of a program or a protocol are specified and intrusions are detected as runtime violations of these specifications. It is introduced as a promising alternative that combines the strengths of anomaly-based and misuse-based detection techniques, providing detection of known and unknown attacks with a lower false positive rate [93]. It can detect new attacks that do not follow the system specifications. Moreover, it does not trigger false alarms when the program or protocol has unusual but legitimate behavior, since it uses the legitimate specifications of the program or protocol [93]. It has been applied to ARP (Address Resolution Protocol), DHCP (Dynamic Host Configuration Protocol) [90] and many MANET routing protocols. Defining detailed specifications for each program/protocol can be a very time consuming job. New specifications are also needed for each new program/protocol and the approach cannot detect some kind of attacks such as DoS (Denial of Service) attacks since these do not violate program specifications directly [44].

When an intrusion is detected, an appropriate response is triggered according to the response policy. Responses to detected intrusions can be passive or active. Passive responses simply raise alarms and notify the proper authority. Active responses try to mitigate effects of intrusions and are divided into two groups: those that seek control over the attacked system, and those that seek control over the attacking system [17]. The former tries to restore the damaged system by killing processes, terminating network connections, and the like. The latter tries to prevent an attacker's future attempts, which can be necessary for military applications.

3.1.2 Future Research on IDS

A great deal of research has emerged in the field of IDS. Major research areas [60] are given below:

- Foundations: research on intrusions, intruders and vulnerabilities
- Data Collection: selecting data sources and features, how to collect data, logging, data format
- Detection Methods: finding the best detection technique(s), improving efficiency and effectiveness of the detection techniques
- Reporting and Response: how to respond to detected intrusions, representation of detected intrusions to the proper authority
- IDS environment and architecture: how to distribute IDS agents and facilitate interoperability between IDS agents, IDS issues on different systems, encrypted networks, etc.
- IDS security: protection of IDS and IDS traffic
- Testing and evaluation: how to test and evaluate IDSs
- Operational aspects: maintenance, portability, upgradeability, etc.
- Social aspects: privacy issues

Most of the research areas above are immature. The majority of research has been carried out on detection techniques. There are also researches on developing standards for IDS like the Intrusion Detection Exchange Format (IDEF). The aim of IDEF is to define data formats and exchange procedures for sharing information of interest to intrusion detection and response systems, and to the management systems which may need to interact with them [3].

IDS testing and evaluation needs more attention. Although there are many proposed IDSs both in commercial and academic domain, rigorous comparisons of approaches seem lacking. In [74], a methodology for testing IDSs is given which is inspired by the field of software testing. In [77] an intrusion detection test bed is developed. Six intrusion detection systems which use different detection methods are tested offline using this test bed. Thirty two attacks against Solaris, SunOs and Linux in four categories (Denial of Service, Remote to Local, User to Root, Surveillance/Probing) are executed on each system. The evaluation results show that the detection rate of new attacks is poor. In their next paper [78] they discuss an interesting subject about predicting whether an intrusion detection system will miss a new attack or not. New attacks that network-based IDSs missed are analyzed. They conclude that such misses result from particular protocols or services that are not monitored or analyzed deeply. Moreover, the rules, thresholds or signatures created for old attacks may not work for new attacks. In conclusion, they suggest a few noteworthy research directions: focusing on anomaly detection and other approaches for detecting new attacks, analyzing a wider range of protocols and services, and exploring new input features.

As stated in [17], IDS trends are shifting from host based IDS to network based IDS, and from centralized IDS to distributed IDS. Moreover, the trends are towards having IDSs that are resistant to attacks and interoperate with other IDSs in heterogeneous environments. There are still many ongoing researches on IDS. The research should give more attention to immature areas such as IDS security and IDS testing. Furthermore, using different detection techniques together is a hopeful research area to increase effectiveness (detecting intrusions while keeping the rate of false positives small) of the system. Not only effectiveness but also efficiency should be improved. Obviously, IDS for MANETs is a new area that needs research. Intrusion detection in MANETs is going to be discussed in the subsequent section.

3.2 Intrusion Detection Issues in MANETs

Even though there are many proposed IDSs for wired networks, MANETs specific features make conventional IDSs ineffective and inefficient for this new environment. Consequently, researchers have been working recently on developing new IDSs for MANETs or changing the current IDSs to be applicable to MANETs. There are new issues which should be taken into account when a new IDS is being designed for MANETs.

Lack of Central Points MANETs do not have any entry points such as routers, gateways, etc. These are typically present in wired networks and can be used to monitor all network traffic that passes through them. A node of a MANET can see only a portion of a network: the packets it sends or receives together with other packets within its radio range. Since wireless ad hoc networks are distributed and cooperative, the intrusion detection and response systems in MANETs may also need to be distributed and cooperative [106]. This introduces some difficulties. For example, distribution and cooperativeness of IDS agents are difficult in an environment where resources such as bandwidth, processor speed and power are limited. Furthermore, storing attack signatures in a central database and distributing them to IDS agents for misuse-based intrusion detection systems is not suited to this environment.

Mobility MANET nodes can leave and join the network and move independently, so the network topology can change frequently. The highly dynamic operation of a MANET can cause traditional techniques of IDS to be unreliable. For example, it is hard for anomaly-based approaches to distinguish whether a node emitting out-of-date information has been compromised or whether that node has yet to receive update information [42]. Another mobility effect on IDS is that IDS architecture may change with changes to the network topology.

Wireless Links Wireless networks have more constrained bandwidth than wired networks and link breakages are common. IDS agents need to communicate with other IDS agents to obtain data or alerts and need to be aware of wireless links. Because heavy IDS traffic could cause congestion and so limit normal traffic, IDS agents need to minimize their data transfers [83]. Bandwidth limitations may cause ineffective IDS operation. For example, an IDS may not be able to respond to an attack in real-time due to communication delay. Furthermore, IDS agents may become disconnected due to link breakages. An IDS must be capable of tolerating lost messages whilst maintaining reasonable detection accuracy [92].

Limited Resources Mobile nodes generally use battery power and have different capacities. MANET devices are varied, *e.g.* laptops, hand held devices like PDAs (personal digital assistants), and mobile phones. The computational and storage capacities vary too. The variety of nodes, generally with scarce resources, affects effectiveness and efficiency of the IDS agents they support. For example, nodes may drop packets to conserve resources (causing difficulties in distinguishing failed or selfish nodes from attacker or compromised nodes) and memory constraints may prevent one IDS agent processing a significant number of alerts coming from others. The detection algorithm must take into account limited resources. For example, misuse-based detection algorithm must take into account memory constraints for signatures and anomaly-based detection algorithm needs to be optimized to reduce resource usage.

Lack of a Clear Line of Defense and Secure Communication MANETs do not have a clear line of defense. In this environment IDS traffic should be encrypted to avoid attackers learning how the IDS works [83]. However, cryptography and authentication are difficult tasks in a mobile wireless environment since they consume significant resources. In many cases IDS agents risk being captured or compromised with drastic consequences in a distributed environment. They can send false alerts and make the IDS ineffective. IDS communication can also be impeded by blocking and jamming communications on the network.

Cooperativeness MANET routing protocols are usually highly cooperative. This can make them the target of new attacks. For example, a node can pose as a neighbour to the other nodes and participate in decision mechanisms, possibly affecting significant parts of the network. (A number of attacks exploiting the collaborative nature of various routing protocols have been described in Chapter 2.)

3.3 Proposed Intrusion Detection Systems in MANETs

In this thesis the proposed IDSs on MANET are reviewed to find out how well they address the IDS issues explained above. The following criteria are used in this survey:

Input Data Intrusion detection systems can use host audit data, network packets or statistics of such data (*e.g.* statistics of updates in routing tables and the number of received packets in the last 10 seconds).

Data Gathering Besides using local data, a node in MANETs snoops to gather its neighbours' data. Since MANET uses wireless links, a node can be monitored by its neighbour nodes. Promiscuous monitoring is required for the systems using this method.

Architecture The architectures for IDS on MANET can be classified into stand-alone, distributed and cooperative, and hierarchical IDSs [13].

In stand-alone IDS architectures, every node in network has an IDS agent and detects attacks on their own without collaborating with other nodes. Because this architecture cannot detect network attacks (network scans, distributed attacks, etc.) with the partial network data on the local node, it is generally not preferred.

Since nodes in MANETs have only local data, a distributed and cooperative IDS architecture is generally used to provide a more informed detection approach. In this architecture, every node has its local IDS agent and communicates with other nodes' agents to exchange information, to reach decisions and respond. Both the stand-alone architectures and the distributed and cooperative architectures are more suitable for flat network infrastructure.

The last architecture is hierarchical IDS which is a kind of distributed and cooperative architecture more suitable for multi-layered networks [13]. In this architecture, the network can be divided into groups such as clusters, zones where some nodes (cluster heads, interzone nodes etc.) have more responsibility (providing communication with other clusters, zones) than other nodes in the same cluster. It is the same from intrusion detection point of view: Each node in the cluster carries out local detection while cluster heads and interzone nodes carry out global detection.

Distributed IDS agents (nodes) are generally divided into small groups such as clusters, zones, and one-hop away nodes, enabling them to be managed in a more efficient way.

Interoperability This aspect refers to the means by which IDS agents communicate with each other. This may be done by traditional network packet based communication or else by mobile agent approach. A mobile agent is a composition of computer software and data which is able to migrate from one computer to another autonomously and continue its execution on the destination computer [6]. It reduces network load by moving computation to data that it is a significant feature for MANETs which have lower bandwidth than wired networks.

Detection Method The most commonly proposed intrusion detection method in MANETs to date is specification-based detection. This can detect attacks against routing protocols with a low rate of false positives. However, it cannot detect some kind of attacks, such as DoS attacks. There are also some anomaly-based detection systems implemented in MANETs. Unfortunately, mobility of MANETs increases the rate of false positives in these systems. There have been few signature-based IDSs developed for MANETs and little research on signatures of attacks against MANETs. Updating attack signatures is an important problem for this approach.

Decision-Making Two different decision-making mechanisms are used in distributed and cooperative IDSs: collaborative decision-making, where each node can take active part in the intrusion detection process, and independent decision-making, where particular nodes are responsible for decision-making [49]. Both decision-making mechanisms have pros and cons. Collaborative-decision making systems are more reliable. If all nodes contribute to a decision, a few malicious nodes cannot easily disrupt the decision-making. However, if any node can trigger a full-force response, it can affect the entire network and be vulnerable to a DoS attack [49]. A collaborative-decision making approach is also more resilient to benign failure of nodes. On the other hand, failing or compromise of particular nodes in independent decision-making systems can have drastic effects. However, these systems are less prone to spoofed intrusion attacks than collaborative decision-making systems [49].

Response Mechanism The system can have a passive response or an active response to detected intrusions. Reputation systems reviewed in [56] are example methods used on MANETs for active responses.

IDS Testing This is the area of IDS testing and evaluating its effectiveness and efficiency.

IDS Security There can be attacks against IDS itself; this feature is the degree of IDS security in opposition to these attacks.

All these features above are summarized in Table 3.1.

Feature	**Explanation**	**Classification**
Input Data	monitored data	network packets, MIB data, host data, statistic, etc.
Data Gathering	from where data is gathered	host-participatory or promiscuous listening
Architecture	structure and organization of IDS agents	standalone, distributed and cooperative, hierarchical
Grouping	how to group distributed IDS agents	clusters, zones, one-hop away nodes
Interoperability	how to communicate with IDS agents	network packets or mobile agents
Detection Method	method used to detect intrusions	anomaly-based, specification-based, misuse-based
Decision Making	how to make decisions about intrusions	local, collaborative or independent
Response Mechanism	how to react to detected intrusions	passive or active response (on controlled or on controlling system)
IDS Security	vulnerabilities of IDS agents	single point of failure, attacks against mobile agents, DoS attacks, etc.

Table 3.1: Survey Features

The main proposed IDSs for MANETs in the literature are described below.

3.3.1 Distributed and Cooperative IDS [106][107]

The first IDS for MANETs proposed by Zhang and Lee is a distributed and cooperative IDS. In this architecture, every node has an IDS agent which detects intrusions locally and collaborates with neighboring nodes (through high-confidence communication channels) for global detection whenever available evidence is inconclusive and a broader search is needed. When an intrusion is detected an IDS agent can either trigger a local response (*e.g.* alerting the local user) or a global response (which coordinates actions among neighboring nodes).

Since expert rules can detect only known attacks and the rules cannot easily be updated across a wireless ad hoc network, statistical anomaly-based detection is chosen over misuse-based detection. The local data is relied on for statistical anomaly-based detection: the node's movement (distance, direction, velocity) and the change of routing table (PCR: percentage of changed routes, PCH: percentage of changes in the sum of hops all the routes).

A multi-layer integrated intrusion detection and response is proposed allowing different attacks to be detected at the most effective layer. It is believed to achieve a higher detection rate with a lower false positive rate than a single-layer intrusion detection achieves.

The RIPPER and SVM-Light classification algorithms are used. In their subsequent research [107], these algorithms are evaluated on three routing protocols: AODV, DSR and DSDV using detection rate and false alarm rate metrics. SVM-Light is shown to have better performance than RIPPER. It is also shown that the protocols with strong correlation among changes of different types of information (location, routing, etc.) have better performance, so reactive (on-demand) protocols are more appropriate for this system than proactive (table-driven) protocols, since reactive protocols reflect the changes in the network better than proactive protocols. Moreover, it is stated that the IDS works better with protocols which include some redundancy (such as path redundancy in DSR). However, the mobility effect is not discussed. Their results on AODV routing protocol are presented in Table 3.2. They use five different testing data. *normal* represents a network without attacks. *rt* shows the data with intrusions on route logic and *tf* shows the data with distortions on traffic patterns. The number 100 and 10 represents the running time (100.000 and 10.000 seconds) and the number of intrusion sessions.

Test Data	RIPPER Algorithm		SVM-Light Algorithm	
	DR	FPR	DR	FPR
normal	N/A	1.45±0.72%	N/A	2.36±1.07%
100k-rt	91.71±3.23%	20.2±6.27%	95.3±0.79%	1.27∓0.38%
100k-tf	88.48±4.14%	17.8±5.10%	93.6±0.72%	2.06∓0.63%
10k-rt	92.36±3.79%	14.4±4.87%	94.7±0.51%	3.28∓0.93%
10k-tf	89.91±5.31%	15.7±3.39%	97.1±0.32%	3.57∓0.79%

Table 3.2: The Performance of RIPPER and SVM-Light Algorithms on AODV

This is one of the few approaches considering mobility by monitoring node movements. This can decrease false positives resulting from the node's mobility. However, it only reflects the local mobility not the network's mobility. Also, every node has to have a built-in GPS (Global Positioning System) to obtain this mobility data. It is emphasized that it can be applied to all routing protocols since it uses the minimal routing information. It also allows addition of new features for a specific protocol. From the security point of view the system is reliable unless the majority of nodes are compromised [106] (These can send falsified data). Furthermore, the collaborative detection mechanism can be prone to denial of service and spoofed intrusion attacks [49].

3.3.2 Cooperative IDS using Cross-Feature Analysis in MANETs [42][43]

Huang *et al.* use data-mining techniques to automatically construct an anomaly detection model [43]. They use an analysis technique that targets multiple features and which acknowledges the characteristic patterns of correlation between them. The basic assumption here for anomaly detection is that normal and abnormal events have different feature vectors that can be differentiated.

In cross-feature analysis, they train the following classification model C$_i$ from normal data based on exploring the correlations between each feature and all other features [42]. Each feature value is estimated by other feature values in the model.

$$C_i : f_1, f_2, ..., f_{i-1}, f_{i+1}, ..., f_L \rightarrow f_i where f_1, f_2, ..., f_L\ is\ the\ feature\ set. \tag{3.1}$$

In practice, each feature f$_i$ is analyzed and compared with the predicted values of f$_i$. Then, the average match count is evaluated by dividing the number of total true matches of all features by L and used to detect anomalies which are below the threshold. Instead of count values, probabilities can also be used. Different classification algorithms C4.5, Ripper, and NBC are investigated to calculate the probability function [42]. Since C4.5 shows better performance, it is the chosen method in their subsequent research [43].

Due to resource-constraints in MANETs, they propose a cluster-based IDS architecture. A fair and secure cluster-head assignment is presented. Cluster-heads are selected randomly to facilitate security. Otherwise few assigned cluster-heads could be the target of attacks easily. Equal service time is assigned to all selected cluster-heads.

Simple rules are also introduced to determine attack types and sometimes attackers. The rules are executed after an anomaly is detected. They are based on statistics such as the number of incoming/outgoing packets on the monitored node and are pre-computed for known attacks. For example, unconditional packet dropping of a node m is formulated as follows [43]:

$$FP_m\ (forward\,percentage) = \frac{packets\ actually\ forwarded}{packets\ to\ be\ forwarded} \tag{3.2}$$

If the denominator is not zero and FP$_m$ is 0, it means that node m is dropping all packets. The attacker is identified by a neighbour of node m who can promiscuously overhear node m's traffic.

It is implemented on the ns-2 simulator by using traffic related and non-traffic related features. Traffic related features are packet type, flow direction, sampling periods and statistics measures (counts and standard deviations of inter-packet intervals). Non-traffic related features represent a view of network topology and routing operations and comprise information such as the number of routes added by route discovery, total route change, and absolute velocity (the physical velocity of a node). The AODV protocol is targeted and the following metrics are used for evaluation: detection rate, false positive rate, and detection rate of attack types. Their results shown in Table 3.3 are promising.

It is the first approach that uses feature correlations. They propose to investigate how computational cost can be reduced [42]. Attacker identification and attacks against the IDS (a major issue for a cluster-head architecture) are identified as future research [43].

Attack	**Detection Rate**	**False Alarm Rate**
BlackHole and Sleep Deprivation	85%	0.97%
Selfishness and DoS	98%	0.89%
Sleep Deprivation	99%	0.95%
Routing Loop	87%	0.98%

Table 3.3: The Performance of The IDS Technique using Cross-Feature Analysis in MANETs

3.3.3 Zone-Based IDS [89]

In [89], a non-overlapping zone-based IDS is proposed. In this architecture, the network is divided into zones based on geographic partitioning to save communication bandwidth while improving detection performance by obtaining data from many nodes. The nodes in a zone are called intrazone nodes, and the nodes which work as a bridge to other zones are called interzone (gateway) nodes. As shown in Figure 3.1 there can be more than one gateway node in a zone, for instance the nodes 1, 6, 7 are gateway nodes in zone 5. Each node in the zone is responsible for local detection and sending alerts to the interzone nodes.

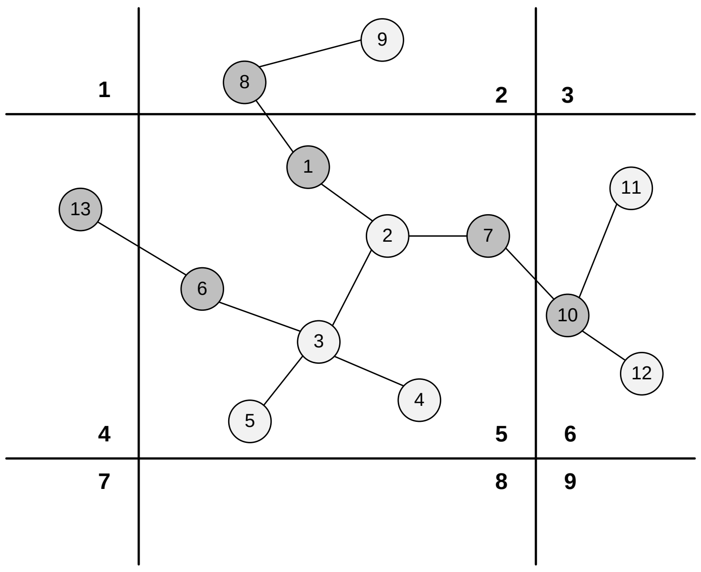


Figure 3.1: Zone-Based IDS Architecture in MANETs

Their framework aims to allow the use of different detection techniques in each IDS agent; however they use only Markov chain anomaly detection in their research. Inputs to IDS agents are the routing table updates (PCR and PCH) as in [42][43].

Intrazone nodes carry out local aggregation and correlation, while gateway nodes are responsible for global aggregation and correlation to make final decisions and send alarms. So only gateway nodes participate in intrusion detection. The alerts sent by interzone nodes simply show an assessment of the probability of intrusion, the alarms generated by gateway nodes are based on the combined information received. In their aggregation algorithm, gateway nodes use the following similarities in the alerts to detect intrusions: classification similarity (classification of attacks), time similarity (time of attack happening and time of attack detection), and source similarity (attack sources). Source similarity is the main similarity used, so the detection performance of aggregation algorithm could decrease with the increasing of the number of attackers [89].

One of the contributions in this paper is MIDMEF (MANET Intrusion Detection Message Exchange Format) which defines the format of information exchange between IDS agents. It is consistent with Intrusion Detection Message Exchange Format (IDMEF) proposed by the Internet Engineering Task Force (IETF) [3].

Previous work [88] analyzed how to consider mobility when designing an IDS. Link change rate is proposed to reflect different mobility levels. Suitable normal profiling and proper thresholds can then be adaptively adopted by IDS agents using this measure. Furthermore, it is shown that link change rate reflects the mobility model of the network better than the generally used mobile speed measure. Link change rate of a node is defined as [88]:

$$\frac{\mid N_1 \setminus N_2 \mid + \mid N_2 \setminus N_1 \mid}{\mid t_2 - t_1 \mid}, \tag{3.3}$$

where N_1 is the neighbor set of the node at time t_1 and N_2 is the neighbor set of the node at time t_2.

The proposed IDS is simulated on the GlomoSim simulator and evaluated using the following performance metrics: false positive rate, detection rate, and mean time of first alarm (a measure of how fast intrusion is detected). The system is trained and evaluated under different mobility levels (with duration time between 0-1000 seconds) and it is shown that the anomaly-based detection performs poorly due to the irregularity of data under high mobility. While the detection rate is around 60% under high mobility, it goes up to the perfect detection rate under low mobility. The false positive rate changes between 5% and 50%. Furthermore, the presence of partial victims who do not receive all falsified data because of link breakages resulting from mobility [89] is claimed to make the detection more difficult. The advantages of an aggregation algorithm using the data from both partial and full victims are emphasized: lower false positive and higher detection rate than local IDS achieves. Nevertheless, its performance can decrease with the existence of more than one attacker in the network. They also conclude that communication overhead is increased in proportion to mobility where local IDSs generate more false positives and send more intrusion alerts to gateway nodes. In addition, aggregating data and alerts at interzone nodes can result in detection and response latency, when there is sufficient data for intrusion detection even at intrazone nodes. The authors plan to investigate further attack scenarios at the routing and other layers as well as constructing further security-related features and misuse-based detection approaches.

3.3.4 General Cooperative Intrusion Detection Architecture [87]

In [87], Sterne *et al.* present a cooperative and dynamic hierarchical IDS architecture which uses multiple-layering clustering. Figure 3.2 shows a network with two-level clusters. The nodes annotated with “1” are the first level cluster-heads, essentially acting as a management focus for IDS activity of immediately surrounding nodes. These level 1 cluster heads can form a cluster around high level node “2”, second level cluster-head. This process goes on until all nodes are assigned to a cluster. To avoid single point of failure, they propose choosing more than one cluster-head for the top-level cluster. The selection of cluster heads is based on topology and other criteria including connectivity, proximity, resistance to compromise, accessibility by network security specialists, processing power, storage capacity, energy remaining, bandwidth capabilities, and administratively designated properties [87].

In this dynamic hierarchy, data flow is upward, while the command flow is downward. Data is acquired at leaf nodes and aggregated, reduced and analyzed as it flows upward. The key idea is given as detecting intrusions and correlating with other nodes at the lowest levels for reducing detection latency and supporting data reduction, whilst maintaining data sufficiency. It supports both direct reporting by participants and promiscuous monitoring for correlation purposes.

The proposed intrusion detection architecture for MANETs targets military appli-

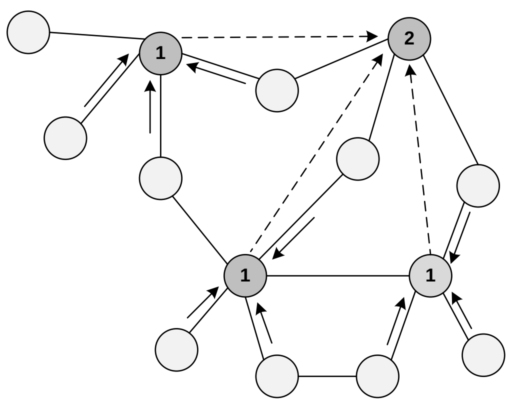


Figure 3.2: IDS Hierarchy with Two-Level Clusters

cations. The authors claim that the dynamic hierarchy feature is highly scalable. It also reduces the communication overhead through the hierarchical architecture. However, the cost of configuration of the architecture in dynamic networks should also be considered.

Neither specific intrusion detection techniques nor the implementation of this architecture is covered. Supporting a broad spectrum of intrusion detection techniques is posed as one of the general requirements of IDS. However, applicability of these techniques to mobile ad hoc networks, which can have resource constrained nodes and no central management points, is not addressed. Examples of usage scenarios, which cover MANET-specific and conventional attacks, are presented (by using different intrusion detection techniques) on the architecture. Some attacks can be drastic in this architecture; for example the capturing of cluster-heads or a malicious node being selected as a cluster-head by sending false criteria. Ongoing areas of investigation are comparison of existing clustering algorithms and communication overhead metrics. They identify the development of Byzantine-resistant techniques for clustering and for intrusion detection and correlation as future work.

3.3.5 Intrusion Detection Using Multiple Sensors [49]

Kachirski and Guha propose an IDS solution based on mobile agent technology which reduces network load by moving computation to data. This is a significant feature for MANETs which have lower bandwidth than wired networks. A modular IDS structure is proposed that distributes the functional tasks by using three mobile agent classes:

monitoring, decision-making and action-taking. The advantages of this structure are given as increased fault-tolerance, communication cost reduction, improved performance of the entire network, and scalability [49].

A hierarchical and distributed IDS architecture is given which divides the network into clusters. Cluster heads are chosen by vote, with each node voting for a node based on its connectivity. Each node in the network is responsible for local detection using system and user level data. Only cluster heads are responsible for detection using network level data and for making decisions. However, depending on the hop attribute of the clusters, network intrusion detection performance can change. For example, every node has direct connection to at least one cluster head in a one-hop clustered network (nodes 1, 2, 5, and 8 are clusterheads), so each packet in the network can be monitored as shown in Figure 3.3(a), while three links in Figure 3.3(b) cannot be monitored by the cluster-heads in a two-hop clustered network (nodes 1, 2, and 5 are cluster-heads). As the degree of monitoring increases the number of cluster heads increases too. So, choosing the hop attribute of the clusters is a trade-off between security and efficiency. However, the nodes not in a cluster head's communication range can move to the monitoring area of another cluster head due to mobility. Having a few links that cannot be monitored by any cluster head is regarded as acceptable for highly dynamic environments.

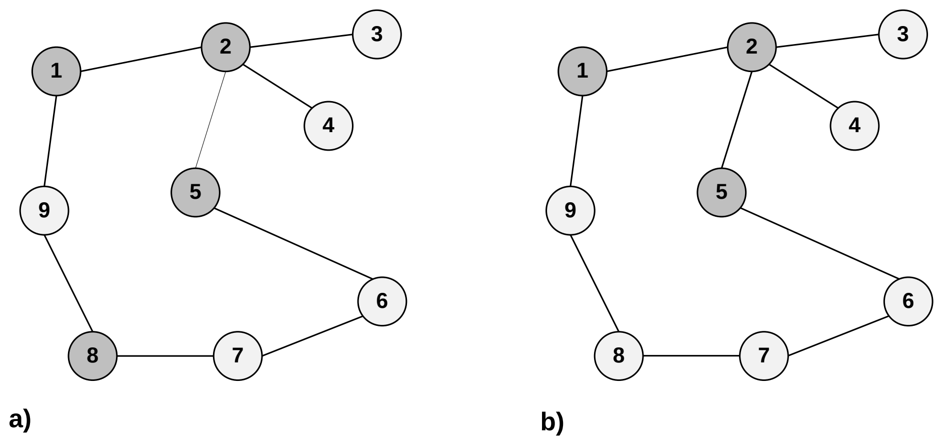


Figure 3.3: IDS Architecture
(a) One-hop Clustered Network (b) Two-hop Clustered Network

Cluster nodes can respond to the intrusions directly if they have strong evidence locally. If the evidence is insufficient they leave decision-making to cluster heads by sending anomaly reports to them.

In this paper a scalable and bandwidth-efficient IDS is proposed by using mobile agents but without giving any validation via simulation or implementation. On the other hand, there are urgent security issues for mobile agents that are set to be investigated in their future research. In addition, details of the anomaly-based detection method are not given, with research on more robust and intelligent cooperative detection algorithms left as future research.

3.3.6 Specification-Based IDS for AODV [90]

The first specification-based IDS in MANETs is proposed by Tseng *et al.* [90]. They use network monitors (NM) which are assumed to cover all nodes. Nodes moving out of the current network monitoring area are also assumed to move into range of other network monitors. Other assumptions are: i) network monitors know all nodes' IP and MAC addresses, and MAC addresses cannot be forged. ii) network monitors and their messages are secure. iii) if some nodes do not respond to broadcast messages, this will not cause serious problems.

Network monitors employ finite state machines (FSM) as specifications of the operations of AODV, especially for the route discovery process, and maintain a forwarding table for each monitored node. Each Route Request (RREQ) and Route Reply (RREP) message in the range of the network monitor is monitored in a request-reply flow. When a network monitor needs information about previous messages or other nodes not in its range, it can ask neighboring network monitors. In high mobility conditions the communication between network monitors increases since monitored nodes or/and packets frequently move out of the range of the monitoring node.

The authors also modify the AODV routing protocol by adding a new field: the previous node. Since RREQs are broadcast messages, it is necessary to keep track of the RREQ path. The previous node is needed to detect some kind of attacks such as sending an RREP to a node that is not on the reverse route [90].

Future work includes experimentation via ns-2 network simulation, profiling network QoS (Quality of Service) to reduce false positives by separating packet loss, packet error, and packet generation through defining reasonable thresholds for the current profile, and refining NM architecture using via a P2P (peer-to-peer) approach.

This is a promising approach that can detect both known and unknown attacks against routing protocols which have clearly defined specifications. It is claimed to detect most of the attacks with minimum overhead in real time. However, some of the assumptions accepted in this paper are not very realistic. For example, assuming the network monitors cover all network nodes and have all nodes' IP and MAC addresses. Scalability is one of the important features on many MANET applications where the nodes can join and leave network independently and move frequently. Assuming MAC addresses cannot be easily forged is unrealistic. Moreover, dropping of some broadcast messages in the network can affect network services if the node dropping messages is at a critical point. Furthermore, the details of the architecture are not addressed (such as the positions of network monitors in MANETs where the topology changes arbitrarily).

There are other specification-based approaches proposed for AODV [36][37]. A specification-based approach combined with cryptography is given in [36]. In this approach RREQ and RREP messages' specifications are given as extended finite state machine in each node. Another finite state machine based approach proposed for DSR is given in [105].

3.3.7 DEMEM: Distributed Evidence-Driven Message Exchanging Intrusion Detection Model [92]

DEMEM is a distributed and cooperative IDS in which each node is monitored by one-hop neighbor nodes. In addition to one-hop neighbor monitors, 2-hop neighbors can exchange data using intrusion detection (ID) messages [92]. The main contribution of DEMEM as stated by the authors is to introduce these ID messages to help detection, which they term evidence-driven message exchange. Evidence is defined as critical information (specific to a routing protocol) used to validate the correctness of the routing protocol messages, for instance hop count and node sequence number in AODV [92]. To minimize ID message overhead ID messages are sent only when there is new evidence, so it is called evidence-driven. DEMEM also introduces an ID layer to process these ID messages and detect intrusions between the IP layer and the Routing Layer (intercepts routing messages between the routing layer and the IP layer) without modifying the routing protocol, so it can be applied to all routing protocols.

DEMEM uses the specification-based IDS model for OLSR proposed in their previous work [91]. In OLSR [46] there are nodes called Multipoint Relays (MPRs) which serve to reduce the flooding of broadcast packets in the network. These nodes are selected by their neighboring nodes called MPR selectors. The packets of an MPR node's MPR selectors are only retransmitted by that MPR node. TC (topology control) messages are sent by each node periodically to declare its MPR selectors. The proposed specification-based system uses the following constraints of OLSR to detect intrusions:

C1: neighbors in Hello messages must be reciprocal.
C2: MPRs must reach all 2-hop neighbors.
C3: MPR selectors must match corresponding MPRs.
C4: Fidelity of forwarded TC messages must be maintained.

The authors state that the system cannot detect collaborative attacks. For example two attackers who falsely claim that they are neighbors might not be detected by the above constraints [91].

DEMEM introduces three authenticated ID messages for OLSR. The first one is ID-Evidence, which is designed for 2-hop-distant detectors to exchange their evidence which is one-hop neighbors, MPRs, and MPR selectors on OLSR. The second message, ID-Forward, is a request to forward any held ID-Evidence messages to other nodes. This means that a detector node can request the holders of evidences (its selected neighbours) to forward their evidences, rather than sending them by itself, so reducing message overhead. It is sent only when there is a new evidence (*e.g.* new neighhbours) on this node. The last message, ID-Request, is designed to tolerate message loss of ID-Evidence messages by requesting them to be sent again. The false positives and delay detection due to message loss are decreased by ID-Request messages. Moreover, they specify a threshold value to decrease false positives due to temporary inconsistencies resulting from mobility. When a detector detects an intrusion, it automatically seeks to correct the falsified data.

DEMEM is simulated on the GlomoSim simulator with the random waypoint mobility model and with different speed and pause time sets for mobile nodes. It is evaluated on few attack scenarios such as man-in-the-middle attack. The approach is very effective for mesh networks where nodes in the network do not move: there is no false positives and no false negatives with 0.05% message overhead in a network with 150 nodes and 3% overhead in a network with 10 nodes. Interestingly, the message overheads of DEMEM are decreased as the number of nodes in the network increases, because the number of Hello and TC messages is greater than ID messages in large networks [92]. The message overhead in the simulation varies between 2% and 30% depending on mobility level. They also show how detection accuracy and detection latency of the system vary with the chosen thresholds.

The applicability of DEMEM to other routing protocols especially on reactive protocols is considered. Because reactive protocols produce fewer routing messages with generally smaller size compared to the periodic routing messages of proactive protocols, IDS on reactive protocols may have a greater message overhead than for proactive protocols [92]. Ongoing research includes implementation of DEMEM on AODV and implementation of a reputation-based cooperative intrusion response model.

There are also other specification-based IDSs proposed for OLSR in the literature. In [95] a general specification-based intrusion detection which can be applied to OLSR and similar MPR-based protocols is proposed. In [32] distributed and collaborative detection (by the victim node and his 1-hop neighbours) is proposed. An extended finite state machine to represent the routing protocol specifications is proposed in [71].

3.3.8 Case-Based Agents for Packet-Level Intrusion Detection [34]

Guha *et al.* [34] propose a case-based reasoning system for packet level monitoring based on a hierarchical IDS architecture. In the case-based reasoning approach, known attacks are formulated as cases in the case archive, which stores the features of known problems as well as the actions to solve these problems. The idea is to search for similar cases in the case archive when a problem is detected on the network. The returned similar cases are used either failure or success, is stored into the archive. In this paper, Snort IDS [10] rules are used as the cases and each node has the database of these rules (which is claimed to be small in size). Since Snort rules need exact matching, this is used instead of searching for similarity in the case archive.

IDS functions (monitoring, decision making and actions) are distributed across several mobile agents. Some of them are presented on all mobile hosts, while others are distributed to only a select of group nodes [34]. All nodes have system-level and user-level monitoring that uses an anomaly-based approach. However, packet-level monitoring, which uses case-based reasoning approach, and decision-making are assigned only to cluster-heads. In their simulation, it is shown that the number of dropped packets by cluster-heads increases as the density of the network increases.

Using both anomaly-based detection for system-level and user-level monitoring, and misuse-based detection for packet-level monitoring increases effectiveness. It is also bandwidth-conscious, since it uses mobile agents. However, the security of the mobile agents still needs research.

In [14], there is a research to investigate the effectiveness of signature-based techniques on MANETs by assuming that the signatures of attacks are known. Only selected n nodes of the network are playing role in intrusion detection for efficiency concern. The results on four different routing protocols (DSDV, AODV, DSR, and TORA) show that signature-based detection techniques will not be effective in the networks which use alternative paths for the same end-to-end nodes. The reason is given as the lack of information on the monitoring nodes due to the use of alternative paths.

3.3.9 An IDS Architecture with Stationary Secure Database [83]

A distributed architecture consisting of IDS agents and a stationary secure database (SSD) is proposed in [83]. All nodes have IDS agents responsible for local detection and collaborating with other agents in need. IDS agents have five components: local audit trail; local intrusion database (LID); secure communication module; anomaly detection modules (ADMs); and misuse detection modules (MDMs). The local audit trail gathers and stores local audit data –network packets and system audit data. The LID is a database that keeps information for IDS agents such as attack signatures, patterns of normal user behavior, etc. The secure communication module is used only by IDS agents to communicate securely with other IDS agents. ADMs use anomaly-based detection techniques to detect intrusions. There can be more than one ADM module in an IDS agent, for example using different techniques for different kinds of audit data. There are also MDMs responsible for misuse-based detection to detect known attacks.

The stationary secure database (SSD) maintains the latest attack signatures and latest patterns of normal user behaviors. It is to be held in a secure environment. Mobile agents get the latest information from the SSD and transfer their logs to the SSD for data mining. The SSD has more storage and computation power than mobile nodes, so it is capable of mining rules faster than the nodes in the network and can keep all nodes' logs [83]. Moreover, updating the SSD rather than all nodes in the network is easy. On the other hand, a stationary database is not suited to all kinds of networks. Military tactical environments with control centres are given as examples of the architecture suitable for SSD. However, nodes in hostile environments may not attach to the SSD. Letting the nodes update themselves with the help of other nodes (which can consume significant bandwidth) is proposed as a solution to this problem.

Implementation and evaluation of this architecture are planned for future work. Although it seems to be an effective approach taking advantage of both anomaly-based detection using data mining techniques and misuse-based detection, it has a single point of failure, the SSD. Moreover, a stationary node goes against the nature of MANETs.

3.3.10 An IDS Model Integrating Different Techniques [44]

Huang and Lee propose an IDS model that uses both specification-based and anomaly-based detection approaches to detect interesting events [44]. A basic (routing) event is defined as the smallest set of causally related routing operations such as receiving/delivering a packet, modifying a routing parameter. An anomalous event is defined as a basic event that does not follow system specifications, such as deleting an entry in the routing table, modifying route messages, etc. [44]. A specification-based approach is used to detect anomalous events that directly violate the specifications of AODV. Anomaly-based detection is used to detect events that do not violate specifications of the routing protocol directly and so require statistical measures.

In the specification-based approach extended finite state automatas (EFSAs) are used to represent the specifications of AODV. Events which include only local node operations are mapped to the transitions of the automata. In the statistical-based approach, features are determined to detect anomalous events that cannot be detected by the specification-based approach, and then a set of detection rules is generated using the RIPPER classifier.

The approach is evaluated using the MobiEmu simulator on some scenarios (not including high a degree of mobility). It is shown that some attacks (*e.g.* route message modification, rushing attacks) are not detected effectively by this approach. It is concluded that these attacks cannot be detected locally [44]. Their results are presented in Table 3.4.

The authors propose a taxonomy of attacks which decomposes an attack into a number of basic events and also propose a model to detect them. They use only local detection, since the local node is the only reliable data source. It is claimed this is the reason why some attacks cannot be detected effectively by this approach. For example the detection of network scan attack needs information from other nodes as well. Furthermore the detection of some attacks needs data from another layers in the protocol stack such as wormhole attack. Therefore the authors plan to investigate multi-layer and global detection. Extracting features for detecting unknown attacks automatically is another issue identified as future research.

Anomalous Basic Event	**Detection Rate**	**False Alarm Rate**
Flooding of Data Packets	92±3%	5±1%
Flooding of Routing Messages	91±3%	9±4%
Modification of Routing Messages	79±10%	32±8%
Rushing of Routing Messages	88±4%	14±2%

Table 3.4: The Performance of Specification-Based and Anomaly-Based Detection

3.3.11 A Modular IDS Architecture [75]

Puttini *et. al.* propose a distributed and cooperative IDS architecture in which each node has a local IDS (LIDS) [75]. They give the detailed modular LIDS architecture based on the intrusion detection model of IETF IDWG (intrusion detection working group) which defines three main components of intrusion detection –sensor, analyzer and manager–, and the communication among these components. Communication between IDS agents are provided by mobile agents which are created, received and managed in the Mobile Agent Framework which is claimed to implement security services. It is allowed to exchange only high level messages such as queries, events and alerts to save resources.

The Management Information Base (MIB) is used as local data source for the following reasons: (i) MIB data has standardized format, (ii) MIB data provides to monitor simultaneously network, host, and application level.

It is one of the few IDSs on MANETs that use misuse-based detection. Since MANETs are very recent systems not much approach work has been carried out to determine the signatures of MANET attacks. Using Finite State Machines (FSM) they describe two attack signatures: one is for a network level attack against OLSR; and the other one is an application level attack.

Their system is implemented using Java2 and evaluated on a MANET of only 6 nodes. They show that the two attacks described in the paper are detected. Performance and scalability are planned areas of work. However, it is supposed to be scalable, since the detection is not distributed all over the network and happened generally in a small number of nodes which are close to each other. Adding anomaly detection module to broaden the spectrum of detected attacks is identified as future work.

An outline of the proposed IDSs is given in Table 3.5. This shows the contribution/novelty each IDS brings and the MANET issues it does not address. However, security and limited resources issues are not shown in the table for each IDS separately, since all proposed systems usually make assumptions about these issues, or pay no attention to them.

3.4 Detection of Misbehaving Nodes

Nodes in MANETs rely on other nodes to forward their packets. However, these intermediate nodes can misbehave by dropping or modifying these packets. Several proposed techniques to detect such misbehaviors are given below.

3.4.1 Watchdog and Pathrater [62]

This is the primary work in detecting misbehaving nodes –nodes that do not carry out what they are assigned to do– and mitigating their effects. Since ad hoc networks maximize total network throughput based on cooperativeness of all nodes for routing and forwarding, misbehaving nodes can be critical for the performance of the network as stated in [62]. In this paper, watchdog and pathrater mechanisms on DSR are proposed to improve throughput of the network in the presence of misbehaving nodes. Nodes can misbehave because they can be overloaded, selfish (wanting to save their own resources), malicious, or simply malfunctioning [62].

The watchdog's work is to detect misbehaving nodes by listening to nodes in promiscuous mode. When a node forwards a packet, the watchdog mechanism of that node

IDS	**Contribution**	**Other Manet IDS Issues**
Distributed and Cooperative IDS	first distributed and cooperative IDS considers mobility	considers only local mobility
IDS Using Cross-Feature Analysis	uses of cross-feature analysis constructs anomaly-based detection model automatically defines rules to detect attack(er)s	high computational cost considers only local mobility does not consider cluster-heads' capabilities
Zone-Based IDS	uses of zone-based architecture defines MIDMEF considers mobility based on changes of node's neighbours	causes detection and response latency even when there is enough evidence on local nodes
General Cooperative ID Architecture	uses multiple-layered clustering	high-cost maintenance of the architecture under high mobility
IDS Using Multiple Sensors	uses mobile agents for a scalable and bandwidth-efficient system	may not monitor each node on the network due to the hop attribute of clusters
Specification-Based IDS for AODV	first application of specification-based detection technique to MANETs	communication overhead (among monitor nodes) under high mobility
DEMEM	introduces ID messages between IDS agents to help detection	may not detect some kind of distributed and collaborative attacks
Case-Based Agents for Packet-Level ID	uses case-based approach and anomaly-based detection techniques together	has difficulties in updating case archives in a distributed environment
IDS Architecture with Stationary Database	explores use of a stationary secure database to keep patterns of normal user behaviors and attack signatures	has a central point –one point of failure
IDS Model Integrating Different Techniques	uses anomaly-based and specification-based detection techniques together	carries out only local detection, may not detect distributed attacks
A Modular IDS Architecture	defines two attack signatures	not scalable

Table 3.5: Outline of the Proposed IDSs on MANETs

monitors the next node to confirm that it also forwards the packet properly. It keeps sent packets in a buffer. When the packets are actually forwarded by next nodes, they are removed from the buffer. If the packets remain in the buffer longer than some timeout period, the watchdog increments the failure count of the node implicated. When the failure count of a node exceeds a threshold, the node is identified as a misbehaving node and a notification is sent to the source node. It is stated that watchdog can also detect replay attacks to some extent. However, since it uses promiscuous listening, it is stated that it might not detect misbehaving nodes in the existence of ambiguous collisions, receiver collisions, nodes that control their transmission power to deceive a listener into believing a message has truly been sent, and nodes that falsely report other nodes as misbehaving. It cannot detect partial dropping attacks and collaborative attacks involving at least two consecutive malicious nodes in a route [62].

Pathrater finds the most reliable path by using link reliability data and misbehaving nodes' information from the watchdog. In DSR, there can be many paths from source to destination, but the shortest path is selected. By using pathrater, the most reliable path is selected instead of the shortest path in the presence of misbehaving nodes. The SRR (send extra route request) extension to DSR can be added to find new paths when all paths include misbehaving nodes. Pathrater gives ratings to each node and provides a path metric based on the ratings of the nodes on the path. The authors state that ratings of the nodes should be rearranged to prevent permanently excluding temporary misbehaving nodes from routing and forwarding.

Watchdog and Pathrater with/without SRR are evaluated on the ns simulator with four different mobility levels by using throughput, overhead and false positive rates as metrics. The results show that watchdog and pathrater increase the throughput by 17% in the presence of 40% misbehaving nodes under moderate mobility with 9%-17% overhead. Under extreme mobility, they increase throughput by 27% with 12%-24% overhead.

The approach detects misbehaving nodes efficiently by using simple techniques without a priori trust relationship information. Moreover, it increases the throughput of the network in the existence of misbehaving nodes, and does so with low overhead. On the other hand, it cannot detect collaborative attacks and partial dropping attacks. Additionally, it is applicable only to source routing protocols, because the watchdog needs to know where the packet is going to be forwarded by the next node. Applying the watchdog mechanism to other protocols requires adaptation. DSR needs modification for the SRR extension when misbehaving nodes exist on all paths. Finally, it rewards and reinforces malicious nodes in their behavior by forwarding their packets while they do not forward for other nodes [21].

3.4.2 Nodes Bearing Grudges [21]

This is an interesting approach for detecting and responding to misbehaving nodes, inspired by the biology concept of reciprocal altruism. It detects misbehaving nodes and responds by not forwarding their packets. The aim of this approach is given as increasing fairness, robustness and cooperation in MANETs.

Each node is responsible for monitoring the behavior of its next hop neighbors and detecting misbehaving nodes. There is a trust architecture and an FSM in each node with four main components: the monitor, the reputation system, the path manager, and the trust manager.

The monitor (neighborhood watch) keeps a copy of recently sent packets. It can compare them with the packets forwarded by the next hop node and can detect routing and forwarding misbehaviors as deviations from normal expected behaviour. The types of misbehavior that can be detected by this system are stated to be: no forwarding, unusual traffic attraction, route salvaging, lack of error messages, unusually frequent route updates, and silent route change [21]. When a misbehaving behavior is detected, a reputation system is called for rating the misbehaving node.

The reputation system (node rating) keeps a local rating list and/or black list which can be exchanged with friends. The rating of a node can change when there is enough evidence, and is based on the frequency of misbehavior occurrence [62]. The rate function also uses weights depending on the source detecting misbehavior. One's own experience by promisuous monitoring has the highest weight, while observations (types of misbehaviour in the routing protocol) have relatively smaller weights and reported experiences from other nodes have weight based on the trust level of these nodes. The reputation system uses only negative experience, research on positive changes and timeouts still needs attention. A path manager is called to take action when sufficient evidence of misbehavior is obtained.

The trust level of nodes is managed by the Trust Manager which is distributed and adaptive. It is also responsible for forwarding alarm messages and filtering incoming messages from other nodes. The trust of a node plays a significant role when exchanging routing information with that node, using it for routing or forwarding, and accepting its forwarding requests.

Path manager may respond to a request from misbehaving nodes in a variety of ways, such as ignoring the request, not replying back to the node, responding to any request for a route that includes misbehaving nodes by sending alerts to the source node, re-ranking paths, and deleting paths including misbehaving nodes [21].

ALARM messages are an extension to DSR and are used to distribute warning information. An ALARM message contains the type of protocol violation, the number of occurrences observed, whether the message was self-originated by the sender, the address of the reporting node, the address of the observed node, and the destination address [21]. When an ALARM received, it is sent to Trust Manager to evaluate its trust level.

Assessment of this approach uses the GlomoSim simulator for evaluation and performance analysis is in progress. Moreover, the use of Game Theory for analytical evaluation is being investigated. One aim of the evaluation is to find the relation between the number of nodes in the network, the number of malicious nodes that can be tolerated, and the number of friend nodes that are needed for detection. In addition, they are planning to analyze the scalability, the cost/benefit ratio, the increase in the number of bits per unit of time forwarded to the correct destination minus any bits lost or retransmitted, and overheads for achieving security (an important consideration for MANETs). Research is needed to determine the degree to which mobility affects how easily promiscuous monitoring may be carried out. The effects of mobility on promiscuous monitoring (which can increase collusions) could be analyzed. Since, it uses a threshold mechanism, the effects of different threshold values for different mobility levels could usefully be assessed.

3.4.3 LiPaD: Lightweight Packet Drop Detection for Ad Hoc Networks [15]

Anjum and Talpade have proposed a practical approach for detecting packet dropping attacks [15]. In this approach every node counts the packets that it receives and forwards and periodically reports these counts to a coordinator node. Promiscuous monitoring is not used since it depends on the link layer characteristics and the link layer encryption approach [15]. That's why every node is responsible for monitoring its packets in LiPaD. The algorithm executed in each node is very simple, which is good for resource-constrained nodes. On the other hand, the network bandwidth consumption can be huge, since every node sends reports of each flow defined by source IP and destination IP to the coordinator node. They suggest compressing and aggregating the reports of multiple flows instead of sending each flow in a packet. However, it still affects network traffic, especially in networks with hundreds of nodes. There will be a heavy computation load on the coordinator node (which analyzes all nodes' reports). The coordinator node needs to be a powerful device and must also be secure as it can be the target of the attacks to disable the detection mechanism. For example, it can be target of DoS attacks (by overloading the coordinator node with reports).

Since the coordinator node analyzes the same flow through the reports from all nodes in the route, it can detect liar nodes that pass the wrong information about the statistics of their packets to the coordinator node [15]. If all the nodes on the route are cooperative and malicious, LiPaD cannot detect packet dropping attacks on this route. It is stated that LiPaD detects selective forwarding attacks. It determines a threshold value for permissible packet loss. The coordinator node also implements rewards and punishments depending on the behavior of the nodes.

It is assumed that IDS messages are encrypted and that nodes use a delivery mechanism for IDS messages to prevent them being dropped.

LiPaD is simulated on a network with 30 nodes using the OP-net simulator. It demonstrates that LiPaD detects malicious packet-dropping nodes even in the presence of non-malicious natural link-loss. On the other hand, the performance of LiPaD needs to be evaluated under high mobility and frequent link-loss. Evaluation of LiPaD performance under increased network traffic and node mobility is needed.

3.4.4 Intrusion Detection and Response for MANET [72]

Parker *et al.* extend snooping based methods to detect misbehavior across routing protocols. A node listens to all nodes in its transmission range, not just the packets forwarded by one of its next nodes (as in watchdog [62]). To detect a malicious node in this approach, it is stated that the node must be in the proximity of a good node and act maliciously. It detects dropping and modification attacks which exceed the value in the threshold table for the particular attack class. However, a node moving out of range of the monitoring node before it forwards packets can be assumed to be carrying out a dropping attack. This issue will be addressed in future by the authors. Also, this approach cannot detect misrouting attacks, since it does not know the next hop of a packet that it monitors.

The intrusion detection protocol can give either a local or global response. In a local response, misbehaving nodes in the Bad Node table are isolated. It is emphasized that it is more effective in more dense networks, since more nodes detect intrusive behavior and prevent malicious nodes from utilizing network resources. In the global response, the maliciousness of a node is determined by a vote by all nodes in a cluster. If the majority of the nodes agree that the node is intrusive, an alert will be broadcast. Voting is initiated by cluster heads. Cluster heads can be malicious but the likelihood of malicious nodes being elected as cluster heads is relatively small.

The approach is simulated using the GlomoSim simulator. The effect of node density (both malicious and normal nodes) on false positives is stressed. The response mechanism also affects the rate of false positives. It is claimed that global response reduces false positives due to rapid isolation of the intrusive nodes from the network.

Another approach for detecting dropping attacks on MANETs is presented in [33]. The algorithm only differentiates dropping attacks from the faults due to broken links. Malicious behaviour is defined as the dropping of data packets starting at some random time and going on from that time onwards. The idea behind the algorithm is based on associating the route error messages of the DSR routing protocol with TCP timeouts. In the DSR protocol, a route error control message is sent back to the source node, if an intermediate node cannot forward the packet to the next hop. TCP timeout occurs when the sender does not receive an acknowledgement within a specific interval. All route error messages on a per flow basis are collected at the source node. When a TCP timeout occurs at this node, it is controlled if there are any route error messages for this flow within the detection interval or not. If there are, they are associated with broken link, otherwise with malicious dropping. Obviously, choosing the detection interval is very important in terms of effective detection and low false positives.

All reviewed IDSs are summarized in Table 3.6.

IDS	**Input Data**	**Detection Method**	**Architecture, Grouping**	**Inter-operability**	**Decision Making**	**Respone Mechanism**	**IDS Security**	**Simulation**
Distributed and Cooperative IDS	local data: node's movement and changes of routing table	statistical anomaly-based detection	distributed & cooperative, -	network packets	local & collaborative	active response on attacked system	prone to DoS and spoofed intrusion attacks	ns-2
IDS Using Cross-Feature Analysis	traffic and non-traffic features	anomaly-based detection	hierarchical, clusters	network packets	independent	active response on attacked system	security of cluster-heads	ns-2
Zone-Based IDS	changes in routing table	markov chain anomaly-based detection	hierarchical, zones	network packets	independent	alarms and response	IDS agents: assumed to be secure	GlomoSim
General Cooperative ID Architecture	host data and network packets	all	hierarchical, multiple-layer clusters	network packets	collaborative	alarms and response	security of cluster-heads	-
IDS Using Multiple Sensors	user, system and network level data	anomaly-based detection	hierarchical, clusters	mobile agents	independent	active response on attacked system	attacks against mobile agents	-
Specification-Based IDS for AODV	routing packets of AODV	specification-based	selective distributed cooperative, nodes in monitors' radio range	network packets	collaborative	alarms	assumed to be secure	-

IDS	Input Data	Detection Method	Architecture, Grouping	Inter-operability	Decision Making	Respone Mechanism	IDS Security	Simulation
DEMEM	routing packets of OLSR	specification-based	distributed & cooperative, -	network packets	collaborative	active response and recovery on attacked system	messages and node identities are assumed to be authenticated	GlomoSim
Case-Based Agents for IDS	user, system and network level data	anomaly-based and misuse-based detection	hierarchical, clusters	mobile agents	independent	active response on attacked system	security of mobile agents	no details
IDS Architecture with Stationary Database	host and network audit data	anomaly-based and misuse-based detection	distributed & cooperative, -	network packets	collaborative	active response on attacked system	one point of failure	-
IDS Model Integrating Different Techniques	network packets and routing table	specification-based and anomaly-based detection	stand-alone	-	local	-	compromised nodes	MobiEmu
Watchdog and Pathrater	network packets	techniques use packet snooping	distributed & cooperative, one-hop away nodes	network packets	local	active response on attacked system	nodes which send false reports to source node	ns
Nodes Bearing Grudges	network packets	techniques use packet snooping	distributed & cooperative, one-hop away nodes	network packets	collaborative with trusted nodes	active response	node identities are assumed to be authenticated	GlomoSim

IDS	Input Data	Detection Method	Architecture, Grouping	Inter-operability	Decision Making	Respone Mechanism	IDS Security	Simulation
LiPaD	network packets	using statistics	distributed & cooperative, -	network packets	independent	active response (reward and punishment)	one point of failure, can be target of DoS and cooperative attacks	OP-net
Intrusion Detection and Response	network packets	techniques use packet snooping	distributed & cooperative, clusters	network packets	collaborative	active response	security of cluster heads	GlomoSim
A Modular IDS Architecture	MIB data	misuse-based detection	distributed & cooperative, -	mobile agents	collaborative	response	assumed to have security services for mobile agents	network with 6 nodes
Our Approach	network packets and routing table	misuse-based detection	local and distributed & cooperative, one-hop away nodes	network packets	local and cooperative	alarms	assumed to be secure	ns-2

Table 3.6: Summary of the Reviewed IDSs on MANETs

3.5 Discussion of Applicability of Proposed IDSs to MANETs

Proposed IDSs for MANETs vary significantly, *e.g.* in terms of their detection technique, architecture, decision making and response mechanisms. All systems have advantages and disadvantages. On the other hand, every proposed system should be considered in its own context. For example, a system using a misuse-based technique is generally not suited to the very nature of MANETs, since attack databases cannot easily be updated without a central point. On the other hand, it can fit a military network which has a central location during peace-time.

Mobility, node capabilities, and network infrastructure are usually the main features examined for proposed MANET IDSs. For highly mobile networks IDSs using anomaly-detection techniques may suffer high false positive rates. Furthermore, an IDS architecture that is easy to set up should be preferred for these networks, *e.g.* IDS agents who collaborate with one-hop away nodes. Besides mobility, node capabilities should also be considered. Simple detection techniques can be more appropriate for nodes with limited resources. Trying to make the techniques simpler can be another approach. For instance, the approach in [96] uses a reduced feature set without significantly decreasing detection rate. Obviously, network infrastructure plays an important role in IDS selection. A hierarchical IDS architecture should be preferred to a multi-layered infrastructure, and distributed and cooperative architecture should be preferred for flat infrastructures [13]. Networks with central points make misuse-based and anomaly-based detection techniques easier to use by maintaining the signature database and user behaviors and analyzing them at these points. There may be an opportunity to use these techniques together in order to increase the effectiveness of the system.

For highly secure networks, the security of IDS and IDS traffic should be considered. For example, use of mobile agents can be avoided. Moreover, IDSs that are able to detect both known and unknown attacks should be preferred. That security requirements of the system can change in different situations (*e.g.* peace time and war time requirements of a military network may differ) should be borne in mind while designing an IDS. For low bandwidth networks communication between IDS agents should be minimized.

None of the proposed systems are necessarily the best solution taking into account different applications. Every organization should choose the appropriate IDS for its network. Moreover, it can change the IDS according to its own requirements and characteristics. For example, it can change architecture of chosen IDS or put different intrusion detection techniques together. Therefore, defining requirements and determining characteristics of the network are very important factors in determining the most appropriate IDS solution.

3.6 Future Research

MANETs are a new type of distributed network whose properties are complex and ill-understood. Intrusion detection on these complex systems is still an immature research area. There are far fewer proposed IDSs for MANETs than for conventional networks. Researchers can focus on either introducing new IDSs to handle MANET specific features or can adapt existing systems. Hybrid approaches may also prove of significant use.

As stated earlier, intrusion detection in MANETs poses special problems. Table 3.5 shows each proposed IDS reviewed in this chapter, identifying any novel contributions together with an indication of notable issues they do not address. They usually emphasize just a few specific MANET concerns. The range of MANET issues should be considered during design to ensure effective and efficient intrusion detection suited to the environment at hand.

We make the following observations about the proposed IDSs:

- The systems generally cover restricted sets of attacks.
- The systems usually target a specific protocol.
- Some proposed IDS systems do not take into account mobility of the network.
- Inadequate acknowledgement is given to the resource constraints that many nodes are likely to be subject to, and to the likelihood of nodes with different capabilities.
- Several network architectures proposed do not fit well with the dynamic nature of MANETs.
- A more extensive evaluation of many of the systems would seem appropriate.

The proposed systems seek to address the *lack of central points* issue on MANETs by proposing distributed and cooperative IDS architectures. Such architectures raise questions about security, communication and management aspects. Suitability of the architecture to the environment is an important consideration in designing IDS. An architecture should not introduce new weaknesses/overheads to IDS. For instance, some of the proposed architectures like cluster-based approaches are costly to build and maintain for high mobility networks. Some have critical points of failure.

Appropriate weight should be attached to *mobility*, especially for anomaly-based IDSs. The false positive rate may be greatly affected by mobility level. The system should be aware of its mobility and current network topology. So, features having information about mobility should be included to the intrusion detection system being designed. How we get information about the mobility of the network and what features of the nodes or the network are related to mobility should be investigated.

Communication between IDS agents should be minimized due to constrained bandwidth of *wireless links*. This is one of the goals of the approach described in [92]. Other proposed systems usually do not pay attention to this issue. MANET Intrusion Detection Message Exchange Format (MIDMEF) is consistent with IDMEF and is defined in [89].

Since the nodes are the only data sources on the network, many nodes might be needed to carry out monitoring, detection and providing data to other nodes cooperatively in order to detect some attacks (*e.g.* distributed attacks). However, nodes can have different computational capabilities. Moreover, some of them cannot be powerful enough for executing complex or large intrusion detection algorithms. There would appear to be insufficient research on the *limited resources* issue. Researchers can consider developing different algorithms for different nodes based on their resources and/or computational capabilities. Besides this, more resource intensive detection algorithms can be applied in order to monitor critical nodes as proposed in [51]. The approach in [96] aims to reduce the number of input features used for IDS for the nodes with limited capability. It is shown that a subset of the features used in [43] reduced by the Markov Blanket algorithm produces almost the same detection results as the approach uses all features.

Testing IDS is an open research area for both MANETs and conventional networks. Some of the proposed systems in MANETs have not yet been implemented. Some of them are tested only on very small simulated networks and with few attack scenarios. IDSs should be tested under different mobility levels and with different network topologies. Defining testing criteria for IDSs and preparing test datasets need research.

3.7 Conclusion

MANETs are a new technology, increasingly used in many applications. These networks are more vulnerable to attacks than wired networks. Since they have different characteristics, conventional security techniques are not directly applicable to them. Researchers currently focus on developing new prevention, detection and response mechanism for MANETs.

In this chapter, we have given a survey of research on IDS for MANETs. Many MANET IDSs have been proposed, with different intrusion detection techniques, architectures, and response mechanisms. We have focused on the contribution/novelty each brings and have identified the specific MANET issues each does not address. Proposed systems generally emphasise only a few MANET issues. MANETs have most of the problems of wired networks and many more besides. As a consequence intrusion detection for MANETs remains a complex and challenging topic for security researchers.

In this thesis we investigate the use of artificial intelligence techniques in order to explore this complex design space. Evolutionary computation techniques introduced in the subsequent chapter are employed to evolve both effective and efficient intrusion detection programs. The suitability of evolved programs to MANETs is also considered during the IDS design. The following features of MANETs are taken into account: mobility, limited resources (power, bandwidth) and lack of central points. The proposed approach and the evaluation results are presented in the subsequent chapters.

Chapter 4

Evolution of An Intrusion Detection System in MANETs

This chapter introduces the problem of intrusion detection on MANETs. It then presents an introduction to the techniques that form the core of our proposed solution. The evolutionary computation techniques Genetic Programming (GP) and Grammatical Evolution (GE) are explained in Section 4.2.1 and Section 4.2.2 respectively. The applications of evolutionary computation techniques to intrusion detection in the literature are also summarized. A rationale is provided for why evolutionary computation is worthy of investigation for the development of intrusion detection on MANETs. Finally the application of each technique (GP and GE) to intrusion detection in MANETs is detailed in Section 4.3.

4.1 Threat Model

MANETs require new approaches to routing and have become the target of new attacks which exploit this cooperative nature of routing protocols. Because collaborative routing is so crucial to MANET operations, we have decided to focus the work in this thesis on the detection of attacks on the protocols that implement collaborative routing. Specifically we will focus on AODV as an exemplar protocol in this research. The basic operations of AODV and the attacks on AODV we aim to detect are explained in detail in the subsequent sections.

4.1.1 Ad-Hoc On Demand Routing Protocol (AODV)

There have been many routing protocols proposed to suit the different needs of MANETs. Unfortunately most of these routing protocols do not consider security. One of the most popular ones is the Ad hoc On-Demand Vector (AODV) routing protocol. In this research the Ad hoc On-Demand Vector routing protocol is employed. The operations of AODV are now described in order to allow a better understanding of the routing attacks explained subsequently.

AODV is a reactive routing protocol, discovering routes only when they are needed. “It offers quick adaptation to dynamic link conditions, low processing and memory overhead, low network utilization, and determines unicast routes to destinations within ad hoc network” [73]. It is claimed that AODV can handle low, moderate, and relatively high mobile rates, together with a variety of data traffic loadings [73]. However, it makes no provisions for security.

There are three main types of messages in AODV: Route Request (RREQ), Route Reply (RREP), and Route Error (RERR) messages. When a node wants to communicate with another node in the network and does not have a fresh route to this destination, it starts the route discovery process by broadcasting a RREQ message for the destination node into the network. Intermediate nodes that receive this request either send a RREP to the source node if they have a fresh route to the destination node and the “destination only” flag is not set, or forward the RREQ message to other nodes. A fresh route is a valid route entry whose sequence number is equal to or greater than that contained in the RREQ message. If the request packet has been previously forwarded by this intermediate node, it is silently dropped. When the destination node receives a RREQ for itself, it sends back a RREP message on the reverse route. The requesting node and the nodes receiving RREP messages on the route update their routing tables with the new route.

Wireless mobile networks can have frequent link breakages due to the mobility of nodes in the network or simply due to transmission errors. “AODV allows mobile nodes to respond to link breakages and changes in a timely manner” [73]. The methods for a node to control its connectivity to its active next hops on AODV are:

- link layer notification using control packets such as link layer acknowledgement messages (*e.g.* ACK or RTS-CTS);
- passive acknowledgement: notification by listening on the channel to determine if the next node forwards the packet or not; and
- receiving any packet from the next node or sending some request packets to the next node, such as RREQ or ICMP Echo Request, or Hello messages which are periodic control messages sent only to one hop neighbours.

Assume that a link breakage to the next hop is detected by the absence of Hello messages in the allowed time interval (or with any of the methods above). The node, who detected the link breakage, invalidates the routes (using this link) in his routing table and notifies other nodes affected by the link breakage by sending RERR messages. If the link breakage occurs on an active route, a local repair mechanism can be initiated. In this mechanism new RREQ messages are broadcast to the destination by nodes on the existing route who detect the link breakage.

4.1.2 Attacks on AODV

The three attacks (targeting at AODV) whose detection is the subject of this research are introduced below.

Dropping Attack

In a dropping attack malicious node(s) drop data packets not destined for themselves with the aim of disrupting the network connection. Selfish nodes also drop data packets not destined for themselves to preserve their resources. Dropping attacks result in reduced network performance by causing the retransmission of data packets, the discovery of new routes to the destination, and the like. Furthermore, they can prevent the end-to-end communications between nodes if the dropping node is at a critical point. In this paper, a dropping attack pattern is defined as dropping data packets not destined for the node itself for a given time interval. Since malicious nodes need to be on a routing path to drop data packets, they have little reason to drop routing protocol control packets such as RREQ, RREP, and RERR messages used in route discovery and maintenance mechanisms of AODV. So, it is assumed that malicious nodes do not drop routing protocol control packets.

While packet losses usually occur due to congestion in wired networks, there can be other causes on MANETs. Major causes of packet losses on MANETs are given below [59]:

- wireless link transmission errors
- mobility
- congestion

Transmission errors depend on the physical characteristics of the channel and the terrain, and they can not be eliminated or reduced by improving routing protocols [59]. Packet losses due to mobility are the result of one of the main characteristics of MANETs.

Mobility of the nodes changes network topology and frequently makes existing routes inactive. Situations like buffer overflows, broken links, and no route to the destination can occur due to mobility and cause packets to be dropped. Lastly, packet losses due to congestion occur when the demands exceeds the capacity of a communication link [59].

Mobility is given as the major cause of packet losses on AODV [59]. It is shown that more than 60% of total packet loss on AODV is due to mobility. We mainly aim to differentiate packet dropping due to malicious behaviour from packet dropping due to mobility in this research. The attacker drops all packets received in the last attack period(=3 seconds) in the simulation. Our attack simulation is much more challenging than other approaches in the literature which drop all packets continuously. That is the reason we keep the attack period reasonably small on purpose.

Route Request Flooding Attack (Ad Hoc Flooding Attack)

Network topology changes frequently on MANETs due to mobility. Moreover link breakages are very common in wireless networks. These may result in making existing routes inactive and discovering new routes by route request packets. Route request messages are sent only when nodes need a new route on reactive routing protocols such as AODV. Evidently, mobility may increase the number of route request packets on the network. In the flooding attack scenario, the attacker exploits this property of the route discovery mechanism by broadcasting a lot of route request messages for randomly selected nodes. The attacker aims to consume the resources of the nodes and the network. We believe that high mobility makes it difficult to distinguish a flooding attack from benign behaviour on a network, since it may also cause a high number of route request packets in the network. In the simulation, the attacker broadcasts 20 route request packets in a row as in [68].

Route Disruption Attack

In a route disruption attack, the attacker sends route reply messages to the victim node without receiving any route request messages from that node. There is no mechanism that checks route request-reply flow in AODV. Even nodes overhearing a route reply message can update their routing tables if the message carries a fresh route. Instead of sending route replies for random destination nodes, the attacker chooses one of its neighbours as a victim. Since the attacker is the victim node's neighbour, he already knows about the active routes of the victim through the routing control packets broadcasted by him, or by promiscuous monitoring. The attacker sends fresh route reply messages (with higher destination sequence numbers) to this node. Since the route reply messages sent by the attacker are fresher, the victim node updates his routing table with the routes that the attacker advertises. Hence the attacker is able to disrupt the victim's active routes and puts himself into the victim's routes. As stated in [89], one or just a few routing control packets could hardly inflict severe damage to the system. So, in the simulation the attacker sends 5-10 route reply packets to the victim in a time interval (a second). This attack has been extended to 3 and 5 seconds in further simulations where the attacker achieves his goal slowly and makes the detection of his malicious behaviour difficult.

Minimal attack parameters are chosen here for each attack scenario (except the dropping attack) from the literature. It is believed that when these parameters are increased, the consequences of attacks will be more obvious on the network. In this research evolutionary computation techniques are employed to detect these three attacks effectively and efficiently. These techniques are explained thorougly in the next section.

4.2 Introduction to Evolutionary Computation

Evolutionary Computation (EC) is a research area inspired by natural evolution. It is loosely based on the process of Darwinian ***survival of the fittest***, where individuals are competing with each other for survival and reproduction in an environment that can only host limited number of individuals [30]. Evolutionary computation uses this approach to create computer programs for a given problem automatically, where candidate solutions of the problem correspond to the individuals, and the best programs correspond to the fittest individuals in a population in natural evolution. The pseudocode of the general steps in evolutionary computation is given below.

```
initialize population
while termination criterion not satisfied do
  execute and evaluate fitness value of each individual
  apply genetic operators (selection, crossover, mutation, etc.) to the individuals
  create new population
end while
return best-of-run individual
```

EC starts with generating a population of individuals (usually randomly) which are candidate solutions for the target problem. Then, each individual is evaluated and assigned a *fitness value* that indicates how well this candidate solves or comes close to solving the problem at hand. Until a termination criterion is satisfied, new populations are generated iteratively by using selection, crossover, and mutation operators as in natural evolution. These genetic operators are used to provide better solutions in the new population.

The main components of an evolutionary algorithm are outlined as follows:

- Representation of Individuals
- Fitness (Evaluation) Function
- Initialization
- Selection Mechanism (Parent Selection)
- Variation Operators
- Replacement Mechanism (Survivor Selection)
- Termination

Representation of Individuals defines how to link a real world problem to the EC world (the problem-solving space) [30]. There have been many different EC techniques proposed in the literature such as genetic algorithms, genetic programming, and grammatical evolution. They generally differ from each other in the representation of individuals.

In natural evolution, individuals are represented by the set of genes contained in a genome, which is called the *genotype*. However, individuals with the same genotype can look different as a consequence of their interaction with the environment. These observable characteristics of an individual are called *phenotype*.

The *Fitness Function* indicates how well a candidate (individual) solves or comes close to solving the problem at hand. The aim of an evolutionary algorithm is to optimize this function. So individuals in a population are selected or replaced based on this measure. Since it influences the search directly, the choice of the fitness function in EC is very important. In natural evolution the fitness of an individual is how well adapted that individual is to its environment. The fitter individuals in a population have a higher chance to survive or mate.

Initialization identifies how to create the first population. The individuals are usually generated randomly at this stage. Every population other than the first population is generated by applying the evolution operators to the previous (old) population in the evolution.

Variation Operators are used to create new individuals using the selected individuals from the old population. It aims to create better individuals in the next generations by using/modifying the fitter individuals in the current populations as in natural evolution. Application of these operators differentiate the evolutionary computation from random search. The main variation operators are crossover and mutation. Crossover mimics the exchange of DNA under sexual production to generate new individuals. This binary operator generates two child individuals by swapping some part of two parent individuals selected. The parameter *crossover probability* shows how likely this operator will be performed on the individuals selected for mating. Mutation is an unary operator that mimics natural mutation by changing selected individuals to introduce diversity into the population. The parameter *mutation probability* shows how likely each part of an individual's genotype elements will be altered. Reproduction is another unary operator used in EC which copies selected individuals without any modification to the new population. The *reproduction probability* shows how likely this operator will be applied to the individual selected.

The *Selection Mechanism* provides a great opportunity for fitter individuals to survive by picking out individuals from the current population based on their fitness values. There are various selection methods such as roulette-wheel, rank-based, tournament selection. Tournament selection is employed in our experiments. In tournament selection, a group of individuals is chosen randomly from the population and the best individual from this group is selected as parent. *Tournament size* defines the number of the individuals in this group.

Replacement Mechanism In EC the population size is constant most of the time, so the individuals who will survive in the next generation need to be selected. A choice is made among the current population and the new individuals generated by variation operators. This choice is based on the fitness value. In contrast to parent selection which is typically stochastic, survivor selection is often deterministic [30]. There are two main replacement mechanisms: simple, and steady-state. In a *simple* replacement approach, the new individuals (children) replace the current population. In the *steady-state* *approach*, only one individual, which is generally the worst member of the population, is replaced. Hence, in the latter method, the best fitness value of the population steadily increases (or stays still) as the number of generations increases. A common mechanism also is (λ, μ). In this approach μ offsprings are generated and pooled with the λ existing population. The best λ are then selected on the next generation.

Termination defines when the evolution process terminates. New populations are generated iteratively by using genetic operators until a termination criterion such as 'the ideal solution is found' is satisfied. However finding the ideal solution may take a very long time for complex problems. That is the reason the parameter *generations* is typically used in evolutionary algorithms. This parameter refers to the pre-specified number of generations at which the evolution stops, whether the ideal solution is found or not. In natural evolution there is no such termination criterion, because the evolution goes as long as the environment changes.

4.2.1 Genetic Programming

Genetic Programming (GP) introduced by Koza [55] is one of the most widely employed evolutionary techniques in the literature. It is claimed that GP has equaled or exceeded the performance of other machine learning techniques, and also evolved better programs than the best programs written by people [19]. It has been applied to many problems.

Representation:

The individuals are typically represented by a tree structure in GP. A GP tree is built from the functions and the terminals. Terminals are the leaves in a tree which are generally the inputs to the GP, constants, or other functions with no argument. Functions can be mathematical operators, boolean functions, program statements (if, loop), and the like which can be applied to terminals. Figure 4.1 shows an example GP tree of depth 3 which represents the following mathematical expression: (22 - $\frac{X}{11}$)+(7 $\times$ cos(Y)). *Tree depth* defines the maximum size of the individuals (trees) which is the length of the longest path in the tree from the root node.

Initialization:

There are two main methods to initialize the first population in GP: full and grow. The full initialization method creates trees where each terminal node reaches the maximum

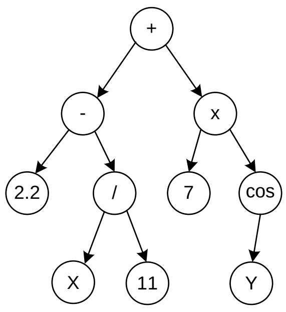


Figure 4.1: An Example GP Tree of Depth 3

tree depth given as a parameter. In this method nodes are selected randomly from the function set until the maximum tree depth is reached, after this level only terminal nodes are selected. In the grow initialization method nodes are selected from the terminal and function sets at any tree level until the maximum depth is reached. So the trees created by this method will not necessarily be full. The *ramped half and half initialization* method which combines both methods is introduced to increase the population diversity in GP. In this method the population is divided equally into groups with different maximum tree depth size between two to the maximum parameter given. Half of the trees of each group are initialized with the grow method, and the other half with the full method.

Variation Operators:

GP employs point mutation by default as shown in Figure 4.2. In this research strongly-typed point mutation is employed. In strongly-typed point mutation a subtree is selected randomly and exchanged with a tree created with the same constrainsts.

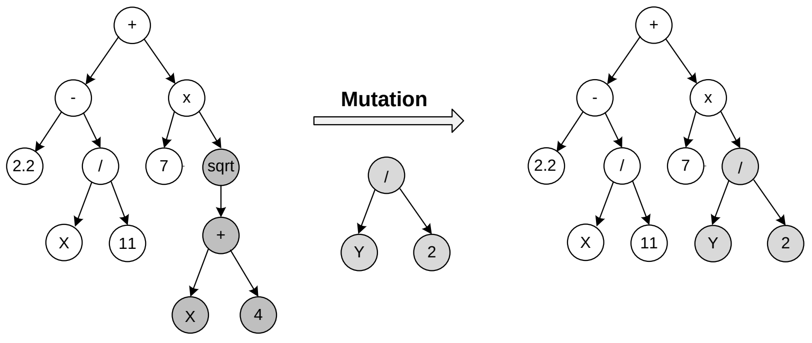


Figure 4.2: Mutation Operator on Genetic Programming

In GP crossover is implemented as a swapping of subtrees of two individuals. Subtree crossover operator is illustrated in Figure 4.3. This research uses a strongly-typed subtree crossover operator, where only subtrees having the same constraints (return type, etc.) are exchanged.

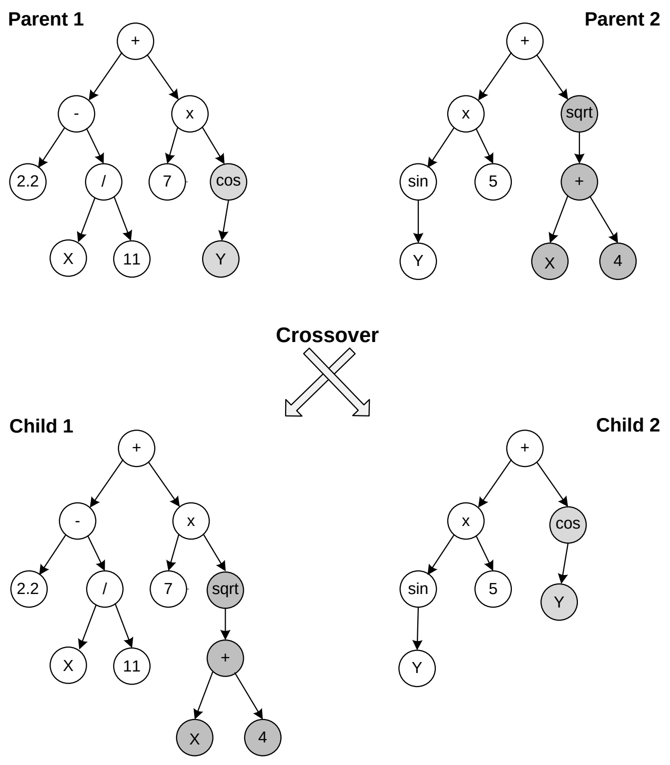


Figure 4.3: Crossover Operator on Genetic Programming

Replacement Mechanism:

GP employs a simple replacement mechanism by default. The new population replaces the old population at each generation.

4.2.2 Grammatical Evolution

Grammatical Evolution (GE) is a technique that allows us to generate programs in an arbitrary language by evolving programs written in a BNF grammar [79]. GE is not the first technique using grammars, but it presents a unique way of using grammars in an evolution process [69]. The core idea of GE relates to how simple integer sequences can be interpreted as programs and this is now described below.

Representation:

GE evolves programs written in a BNF grammar. BNF (Backus-Naur Form) is a formal way to describe a language by defining a set of rules. A BNF system is described as a quadruple: T, N, P, S. T is a set of terminal symbols, which are concrete terms in the grammar. N is a set of non-terminal symbols, which are place-holders used in the generation of terminals by using the set of production rules P. P provides mapping from non-terminal symbols to sequences of terminal or non-terminal symbols. S is the start symbol where mapping starts. A BNF grammar for a symbolic regression problem is given in Table 4.1 [70]. The symbols enclosed by brackets (<>) are non-terminals, others are terminals. The productions of a rule assigned by '::=' are separated with '|'.

```
S = <expr>
<expr> ::= <expr><op><expr> | (<expr><op><expr>) |
           <pre-op>(<expr>) | <var>
<op> ::= + | - | / | *
<pre-op>::= sin | cos | exp | log
<var> ::= X | 1.0
```

Table 4.1: BNF Grammar for a Symbolic Regression Problem

In GE genomes are represented by variable-length binary strings. A group of 8 bits in a genome produces a codon value, which is used to choose a rule from a BNF grammar. Assume that the genome of an individual with consecutive codons interpreted as integers is:

220	35	47	68	137	55	144	22	46	178

At any stage the interpretation proceeds by expansion of the leftmost non-terminal. The first codon value is used to choose the production rule from the start rule S. Since S has one production *expr*, 220 is used to choose one of the productions of *expr* according to the formula below:

$$Rule = (codon\ value)\ MOD\ (number\ of\ productions\ of\ the\ non-terminal) \quad (4.1)$$

expr has 4 productions, so (220 MOD 4 = 0) is calculated and the production '<expr><op><expr>' is selected accordingly (production options are numbered starting from 0). Then, the next codon value 35 is used to choose from *expr* rule, since it is the first non-terminal at this point. So, the third (35 MOD 4 = 3) production of the *expr*, '<var>', is selected. The individual becomes '<var><op><expr>', the next codon value 47 is used to choose the first production of *var* rule (47 MOD 2 = 1) to give '1.0<op><expr>'. The process continues until there are no unmapped non-terminal symbols. If there are still non-terminal symbols to be expanded and all integers in a string are used, the interpretation simply returns to the beginning of the genome again and starts re-reading the string. This process is called *wrapping*. However, there is a threshold value for such wrapping. If this value is exceeded and there are still unmapped non-terminals, this individual is assumed invalid and assigned the lowest fitness value.

GE maps a binary string to a BNF grammar (program) as genotype-phenotype mapping in natural evolution. This provides an unconstrained searh of the genotype, while it ensures the validity of the phenotype (program) [18]. Moreover GE supports silent mutation where the mutations on a genotype do not result in a change on the outcome phenotype. The formula given in Equation 4.1 is the interpretation of many different genotypes. For example two different codon values 15 (15 MOD 4 = 3), 39 (39 MOD 4 = 3) in two different genomes can result in choosing the same production of an expression with four production rules. This feature of GE helps the maintenance of genetic diversity within a population [70].

Initialization:

GE employs sensible initialization to create the inital population. Sensible initialization is based on ramped half and half initialization in GP but generates derivation trees of equivalent size [70].

Variation Operators:

GE employs one-point crossover by default. In one-point crossover, two points are selected on each parent randomly and their genetic contents are exchanged starting from these points. This operator is illustrated in Figure 4.4.

As a mutation operator point mutation which mutates each bit of a genome with a given probability is employed in this research. It is possible that more than one bit making up a codon value will be changed. However, for low probability of individual bit mutation, this is clearly unlikely.

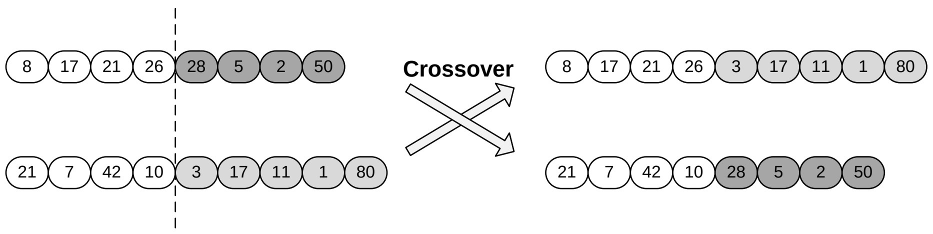


Figure 4.4: Crossover Operator on Grammatical Evolution

Replacement Mechanism:

GE is proposed as a steady-state approach by default to eliminate generated invalid individuals in evolution. These individuals are assigned to the lowest fitness value in a population and might result in slowing down the evolution process. These individiuls are likely to survive in the next generations with a simple selection approach. However they might be eliminated with a steady-state approach which generally replaces the worst individuals of an population with the new individuals. The positive effect of steady-state approach on GE has also been observed in our results.

4.2.3 Related Work

Applications of evolutionary computation techniques to intrusion detection have employed generally either genetic programming (GP) or genetic algorithms (GA) so far. One of the first GP applications to intrusion detection was given in 1995 by Crosbie and Stafford [28]. The main idea in that research is to train autonomous agents based on the input features and the functions given to detect intrusive behaviours. If a malicious activity (believed to be detected easily) is mis-classified during the evolution process, it is penalised heavily in the fitness function. There are also promising applications of genetic algorithms proposed for misuse-based intrusion detection given in [57][63].

Two recent approaches use GP and evaluate its performance on the KDD-99 data set [4], which is the most widely used benchmark evaluation data for intrusion detection on wired networks. In [12], the output program evolved by GP is small, simple and uses just a few input features where "most machine learning paradigms (artificial neural networks, support vector machines, decision trees) examine all input features to detect intrusions [12]". The results of the evaluation show that the approach is lightweight and effective, satisfying the main goals of an intrusion detection algorithm. The GP techniques used in that research are compared with some other machine learning techniques (Support Vector Machines and Decision Trees) on intrusion detection in [11].

The results show that genetic programming technique outperforms other techniques and is a lightweight approach which uses far fewer input features. In [84] linear GP is efficiently trained on a large data set by using the RSS-DSS (Random Subset Selection-Dynamic Subset Selection) algorithm. This approach also uses a small set of the features.

Grammatical evolution has been proposed recently for intrusion detection on wired networks [100]. It is applied to the KDD-99 data set [4] and evolves programs to detect different class of attacks such as DoS, and probes. While the classification accuracy is higher for the DoS attack class, the U2R (user-to-root) and R2L (remote-to-local) attacks show low detection rates. The grammar, the fitness function, the training data, and the features all might affect the results. In summary, the application of GE to intrusion detection in wired networks is in its early stages and improvements are very likely possible.

There is little research on applying evolutionary computation techniques to few problems in sensor networks. Evolutionary computation techniques have not beed applied to intrusion detection problem in sensor networks so far. Researchers generally focus on how to adapt GP to the new environment, wireless sensor networks. In [97], the Distributed Genetic Programming Framework (DGPF) which automatically discovers distributed algorithms for given problems was introduced for sensor networks. The election problem (to select one node out of a group of nodes, for example to act as a communication relay [97]) is solved by using this framework and, a multi-objective optimization technique is also employed on this problem by considering non-functional fitness functions such as code size, memory size, and transmission count for this resource-constrained environment. Another research aims to adapt the GP system to work in a low-power environment such as wireless sensor networks by using a distributed parallel genetic algorithm, limiting individual sizes and the like [48].

4.2.4 Why Evolutionary Computation?

All intrusion detection techniques have their strengths and weaknesses. It is hard to say one technique is better than others. Different techniques are often used together for an effective intrusion detection in conventional networks. Intrusion detection architectures combining different techniques are also proposed for MANETs. A great deal of research has been done on specification-based intrusion detection for different MANETs' routing protocols. Its combination with anomaly-based intrusion detection techniques has also been suggested since specification-based techniques can not detect DoS attacks.

Even though using anomaly-based and specification-based techniques together with misuse-based approaches are proposed, there has been little research in finding the signatures of known attacks against MANETs. In this research this issue has been addressed by using techniques from artificial intelligence to find intrusion detection rules automatically.

MANETs are a new type of distributed network whose properties are complex. It is hard to distinguish attacks from normal activities under a dynamic environment such as MANETs. It is far from clear whether the human perception of what makes a good intrusion detection algorithm in these contexts really is the best possisevil senble. Moreover resource-constrained nodes require different tradeoffs to be made between intrusion detection ability of programs and their resource usage. Humans are not particularly adept at selecting good choices when complex tradeoffs have to be made. In this research evolutionary computation techniques are proposed to discover automatically complex properties of MANETs. Although various artificial intelligence technqiues have been proposed for intrusion detection, EC is one of the most promising approaches. It makes fewer assumptions about the solution space. Of course, it is possible to use other heuristic computation techniques to derive intrusion detection systems. However, our chosen EC approaches are very flexible. IDS programs derived using GP or GE lend themselves to some degree to manual analysis. We can often see what the program is doing. Many alternative approaches, such as neural networks, often produce results that are very hard to understand. Furthermore recent research [11][84] shows that the programs evolved by EC are lightweight and use far fewer features compared to some other machine learning techniques. The representation of IDS problems in an evolutionary computation framework is also quite easy. These characteristics are among the main motivations behind using EC in this research. Futhermore multi-objective evolutionary algorithms allow us to optimize multiple objectives simultaneously. So they can be used to discover detection programs that are both effective (*i.e.* detect intrusions without a high false positive rate) but also efficient (in particular, suited to deployment on constrained resource platforms). These features make EC very attractive for the development of intrusion detection programs suitable for MANETs.

4.3 Evolving Intrusion Detection Rules

In this section we detail how to apply evolutionary computation techniques to derive intrusion detection programs for MANETs.

4.3.1 Feature Selection

"Features" are characteristics of our system whose measurements provide the inputs to our evolved decision algorithms. They provide the basic data any such evolved algorithms can use to reach a result. The choice of which characteristics can be used for these purposes is very important. They must contain sufficient information to allow the fundamentals to be developed.

Features	**Explanation**
neighbours	no. of neighbours
added_neighbours	no. of added neighbours
removed_neighbours	no. of removed neighbours
active_routes	no. of active routes
repaired_routes	no. of routes under repair
invalidated_routes	no. of invalidated routes
addedroutes_disc	no. of added routes bsevil seny route discovery mechanism
addedroutes_notice	no. of added routes by overhearing
updated_routes	no. of updated routes (modifying hop count, sequence number)
added_repairedroutes	no. of added routes under repair
invroutes_timeout	no. of invalidated routes due to expiry
invroutes_other	no. of invalidated routes due to other reasons
avg_hopcount	average no. of hop counts of active routes
recv_rreqPs	no. of received route request packets destined to this node
recvF_rreqPs	no. of received route request packets to be forwarded by this node
send_rreqPs	no. of broadcasted route request packets from this node
frw_rreqPs	no. of forwarded route request packets from this node
recv_rrepPs	no. of received route reply packets destined to this node
recvF_rrepPs	no. of received route reply packets to be forwarded by this node
send_rrepPs	no. of initiated route reply packets from this node
frw_rrepPs	no. of forwarded route reply packets from this node
recvB_rerrPs	no. of received broadcast route error packets (to be forwarded or not)
send_rerrPs	no. of broadcasted route error packets from this node
recv_aodvPs	no. of received total routing protocol packets
recvF_aodvPs	no. of received total routing protocol packets to be forwarded
send_aodvPs	no. of initiated total routing protocol packets from this node
frw_aodvPs	no. of forwarded total routing protocol packets by this node
dropped_dataPs	no. of data packets not forwarded by the next node

Table 4.2: The Features

In this research we have sought to allow a considerable degree of expressiveness. Table 4.2 shows the features maintained at each node by the routing protocol. (Some additional information is also stored, e.g. some configuration constants, but these are of little use for our purposes and so are omitted from the table.) We provide a rich set of features and expect our technqieus to select judiciously from them. It would be difficult to confidently supply apriori a narrower set of features. The features used in this research can be categorized into two main groups: mobility-related and packet-related features. Mobility-related features help reflect the mobility model of a node or the network. How to consider mobility when designing an IDS is analyzed in [88] and the link change rate in Equation 4.2 is proposed to reflect different mobility levels. It is shown that link change rate reflects the mobility model of the network better than the generally used mobile speed measure. Furthermore the physical movements only give an idea about local mobility, not the network's mobility. Hence the features related to nodes' physical movements (speed, velocity and direction) are not used in this research. The feature link change rate is not used directly either. The evolutionary computation algorithm is allowed to discover mobility level by using the features and the mathematical functions given as input. It could evolve the same formula given in Equation 4.2. Some of these mobility features give information about the mobility directly such as changes in the number of neighbours (*e.g. added_neighbours*, *removed_neighbours*). Others can be the results of mobility such as changes in the routing table (*e.g. addedroutes_notice*). For example the active routes can be inactive due to mobility and so new routes need to be discovered. Hence a significant increase in the number of added routes can result from high mobility. It is believed that these features can help to reflect not only the local mobility but also the network mobility or the mobility around a node.

$$\frac{\mid N_1 \setminus N_2 \mid + \mid N_2 \setminus N_1 \mid}{\mid t_2 - t_1 \mid}, \tag{4.2}$$

where N_1 is the neighbor set of the node at time t_1 and N_2 is the neighbor set of the node at time t_2.

Packet-related features include information about the frequency of the routing protocol control packets (RREQ, RREP, RERR) sent, received, forwarded in a time interval, and the routing table updates respectively.

Some of the features in Table 4.2 are specific to some attacks. The average hop count (*avg_hopcount*) feature is used only for detection of route disruption attacks since this feature can reflect the abnormal changes in a victim node. Similarly an increase in the number of data packets not forwarded by the next node (*dropped_dataPs*) can be a sign for a dropping attack and employed only for detection of this attack.

These features are gathered periodically by each node. All features are local to a node, so no communication with other nodes is needed to gather them. The other option is to collect data when an event (such as a control routing packet is received) occurs. However, such event-driven feature collection is very expensive in this dynamic environment. Because routing protocols aim to meet network needs continually in MANETs (especially in on-demand routing protocols), they may result in a lot of routing protocol control packets on the network. That is why a periodic approach, where the features are collected every second is chosen over an event-driven approach in this research.

4.3.2 Application of Genetic Programming to Intrusion Detection in MANETs

In GP, a problem is defined with functions and terminals (features) which are the parts of a GP tree, and the fitness function. The strongly-typed GP (STGP) [66] is employed here. STGP enforces data type constraints. For example a genetic function could be forced to use specific data types (as input and output). The operators applied to these variables, (mathematical, relational, and comparison operators) are given in the Table 4.3.

Objective	Find a computer program to detect ad hoc flooding and route disruption attacks against MANETs routing protocols
Function set	The binary operators +, -, ×, /, pow, min, max, mod, percent The unary operators sin, cos, log, ln, sqrt, abs, exp, floor, ceil The comparison operators <, <=, ==, !=, >, >= The relational operators and, or
Terminal set	The feature set given in Table 4.2
Population Size	100
Generations	1000
Crossover Probability	0.9
Reproduction Probability	0.1
Tournament Size	7

Table 4.3: The GP Parameter Settings

The fitness function is very important in evolutionary computation, since it evaluates how good the individual is. In our experiments we use a fitness function based on the the main metrics used to evaluate an IDS (*i.e.* detection rate, false positive rate) as shown in the equation below. The detection rate shows the ratio of correctly detected intrusions to the total intrusions on the network. The false positive rate shows the ratio of normal activities that are incorrectly marked as intrusions to the total normal activities on the network. An acceptable low rate of false alarms is as important as true alarms in intrusion detection. A high false positive rate will cause a good deal of time to be wasted and will likely destroy confidence in the IDS.

$$\begin{aligned} detection\ rate &= \frac{correctly\ detected\ attacks}{total\ attacks} && (4.3) \\ false\ positive\ rate &= \frac{normal\ activities\ incorrectly\ detected\ as\ attacks}{total\ normal\ activities} && (4.4) \\ Fitness &= detection\ rate - false\ positive\ rate && (4.5) \end{aligned}$$

The toolkit *ECJ 18* [2] is used for the GP implementation in the experiments. The GP parameters used in this research are given in Table 4.3. The parameters except the population size and the generations are the default parameters of the toolkit. The population size is decreased to 100, the generation size is decreased to 1000 in order to decrease the algorithm's execution time. Obviously parameter choices could affect the performance of the system. However, our significant use of default settings will make future comparisons easier. We can make no claim to optimality of these choices. Indeed, finding optimal parameter choices is a significant challenge encountered when EC techniques are applied to almost any problem. Our choice of parameters is a pragmatic one. We aim to show that good results can be achieved with the specified sets of parameters. With very significant computational resources, better results could likely be obtained. (In practice, if IDS designers were to use our technique to evolve programs for their specific networks, a degree of parametric experimentation would be recommended. Again, no claim to optimality could then be made, but experimentation might continue until the designers were happy with the performance of the programs that were evolved.)

4.3.3 Application of Grammatical Evolution to Intrusion Detection in MANETs

In GE, a problem is defined with a grammar and a fitness function. In this research the grammar in Table 4.4 is used to evolve programs in order to detect the attacks (ad hoc flooding, route disruption, dropping) on MANETs and raise an alarm.

This grammar returns an 'if' statement. The variables and the functions used in this grammar are the same as those used in GP. Even though more complicated grammars including 'if-else' statements, loops are employed at first, it is observed that simplified

```
S = <code>
<code> ::= if(<cond>) {raise_alarm()}
<cond> ::= <cond><set-op><cond> | <expr><relop><expr>
<expr> ::= <expr><op><expr> | (<expr> <op><expr>) |
           <pre-op>(<expr>) | <pre-op2>(<expr>) | <var>
<op>     ::= + | - | / | ×
<pre-op> ::= sin | cos | log | ln | sqrt | abs | exp | ceil | floor
<pre-op2>::= max | min | pow | percent
<rel-op>  ::= < | ≤ | > | ≥ | == | !=
<set-op>  ::= and | or
<var>     ::= The features given in Table 4.2
```

Table 4.4: The BNF Grammar Used for the Problem

programs make the evolution process much easier. Moreover the grammar in Table 4.4 shows a good performance on detecting attacks as shown in the results section. The fitness function in Equation 4.5 is used in the GE algorithm as well.

The *libGE* [5] library is used for the GE implementation in this research. The GE parameters used in the experiments are given in Table 4.5.

Programs are evolved using GP and GE with the parameters, the inputs and the fitness function introduced in this chapter. A simulated network under medium mobility and traffic is used in the evolution process to evaluate the candidate solutions on. The "best" solutions obtained by the evolutionary run are subsequently evaluated on simulated networks with varying mobility and traffic patterns.

Objective:	Find computer programs to detect ad hoc flooding, route disruption and dropping attacks on MANETs
Non-Terminal Operators:	The binary operators +,-,×, /, pow, min, max, mod, percent
	The unary operators sin, cos, log, ln, sqrt, abs, exp, floor, ceil
	The comparison operators <, <=, ==, !=, >, >=
	The relational operators and, or
Terminal Operators:	The feature set given in Table 4.2
Fitness cases:	The given sample of network data marked malicious or non-malicious
Raw Fitness:	The detection rate over the fitness cases subtract the false positive rate over the fitness cases
Standardised Fitness:	Same as raw fitness
Parameters	Population Size = 100
	Generations = 2000
	Crossover Probablity = 0.9
	Mutation Probability = 0.01
	Steady State

Table 4.5: The GE Tableau for Detecting Known Attacks on MANETs

Chapter 5

Performance Evaluation of Evolutionary Computation on Intrusion Detection

This chapter presents the evaluation results. Firstly the simulation environment is introduced in Section 5.1. Then the performance of evolved programs using GE and GP on simulated networks with various mobility and traffic patterns are demostrated and discussed in Section 5.2 and Section 5.3. The effects of variations in attack scenarios on the results are also discussed. The performance of evolved programs are also compared with the hand-coded programs in Section 5.4. Lastly the performance of GP and GE for evolving intrusion detection programs in MANETs is compared fairly by using a Design of Experiments methodology explained in Section 5.5.

5.1 Simulation Model

In this research the networks are simulated by *ns-2* [7] to evaluate the performance of evolved programs by using GP and GE. Mobility patterns of the nodes on the network are created using *BonnMotion* [1]. The Random Waypoint mobility model is employed here. In this model, each node moves from its current location to a random new location with random speed and pause time within determined speed/pause time limits [23]. The parameters of the network simulation are given in Table 5.1.

Different network scenarios are created with different mobility levels and traffic loads. 50 nodes are placed in a 1000m by 500m grid. TCP traffic is used for communication. The maximum number of connections is set to either 20 or 30 to simulate different traffic loads. The maximum speed of nodes is set to 20 m/sec and the pause time between movements is set to 40, 20, and 5 sec to simulate low, medium, and high mobility respectively. AODV is chosen as the routing protocol and AODV periodic hello messages are used for local link connectivity. The simulations run 5000 seconds to

network dimensions	1000x500
number of nodes	50
packet traffic	TCP with 20 and 30 connections
speed	0-20 m/sec, pause time 40, 20, 5 sec to simulate low, medium and high mobility respectively
routing protocol	AODV
radio propagation	two-ray ground model with 250m transmission range
local link connectivity	AODV periodic hello messages
simulation time	5000 sec (training), 2000 sec (testing)

Table 5.1: The Parameters of Network Simulation

create training data and 2000 seconds to create testing data.

The detection programs are evolved using the training data collected from a network under medium mobility with 30 TCP connections. The same network with attacks and without attacks are used together for training to reduce false positives. The best result of ten runs is chosen for each attack type and then evaluated on different network scenarios.

Separate programs are evolved for each attack by offline training. Intrusion detection programs are distributed to each node on the network. Each node gathers the features defined in Table 4.2 at each time interval. We assume that attacks are detected by the nodes that the attacks affect directly.

- In dropping attacks, the monitor nodes are the ones who forward packets to the malicious node.
- In flooding attacks, the nodes who are flooded by route request messages detect the attack.
- In route disruption attacks, the victim node is assumed to detect malicious change in its routing table.

The performance of evolved programs using GP and GE, and hand-coded programs are evaluated on the simulated networks and discussed in the subsequent sections.

5.2 The Performance of Grammatical Evolution

The GE algorithm is run ten times on the training data and the best result of ten runs is evaluated on simulated networks. Table 5.2 shows the performance of the evolved program for each attack on the network under medium mobility with 30 TCP connections (the same traffic and mobility levels that the training network has). The false positive rates on the same network under no attack and on the stable network (no mobility) under no attack are also given in the table.

The results show that the evolved program for the dropping attack has a perfect detection rate. However it results in a high false positive rate. As it is stated before, packet losses occur frequently on MANETs due to congestion, wireless link transmission errors, and mobility. So, the features beyond the routing protocol are also needed to detect malicious dropping effectively. We believe that the system could be improved by adding these features to the evolution. Furthermore the attack takes a few seconds in the simulation here since the main aim is to differentiate malicious packet dropping from benign packet dropping. In the literature many approaches define this attack as when the attacker drop packets continuously (or for a long time) and thus makes the attack very obvious to detect. The dropping attack scenario used in this research is clearly more challenging than given in those approaches.

Attacks	**Detection**	**False Positive**	**False Positive without attack (with mobility)**	**False Positive without attack (no mobility)**
Dropping Attack	100%	8.46%	8.81%	10.30%
Flooding Attack	99.86%	2.00%	2.19%	5.11%
Route Disruption	100%	0.83%	0.81%	0.59%

Table 5.2: The Performance of GE on a Network with Medium Mobility/Traffic

In the results given in Table 5.2 the false positive rate of the detection program for the ad hoc flooding attack on the stable network is higher than for the mobile network. Since the programs are evolved under medium mobility, this is quite expected. Furthermore lots of route request messages are noticed in the stable network. The nodes who cannot communicate with their destination nodes (the network could be partitioned) broadcast route request messages frequently to find routes. As a consequence of lots of route request packets under no mobility the system results in high false positive rate.

It it observed that the false positive rate on the network under attack is slightly more than the network without any attacks for the detection of route disruption attack in Table 5.2. It might be the consequence of the attack whose effect takes longer on the network than it is assumed.

For each attack, the performance of the evolved programs is also tested on the network under different mobility and traffic levels. The results for each attack are given in Table 5.3.

Network Scenarios	**Flooding Attack**		**Route Disruption Attack**	
	DR	FPR	DR	FPR
low mobility low traffic	99.81%	0.29%	100%	0.41%
low mobility medium traffic	98.54%	1.72%	100%	0.88%
medium mobility low traffic	99.86%	0.36%	100%	0.40%
medium mobility medium traffic	99.86%	2.00%	100%	0.83%
high mobility low traffic	99.96%	0.66%	100%	0.44%
high mobility medium traffic	98.66%	1.73%	100%	0.76%

Table 5.3: The Performance of GE on Simulated Networks

The results demonstrate that programs evolved using a grammatical evolution approach show good performance on intrusion detection in mobile ad hoc networks. The detection rate is high for both ad hoc flooding attack and route disruption attack, since the results of these attacks are quite evident on the network. However they are not easily differentiated from normal network operations under high mobility and high traffic, but still achieve a false positive rate under 2%.

The false positive rate changes due to the mobility and the traffic load. However they are not the only factors affecting the false positive rate. Other factors such as network topology, traffic and mobility patterns also affect the results. For instance, it is expected that the number of route request packets will be much higher in high mobility networks, since mobility could frequently make existing routes inactive. In Figure 5.1 the number of route request packets on a normal network and on a network under ad hoc flooding attack are presented. It shows little difference in the numbers of route request packets among the networks with different mobility levels. The network under low/medium mobility may also broadcast a lot of route request packets due to its topology, its mobility and traffic patterns to build and preserve its active routes. In conclusion mobility is not the only major factor affecting the number of route request packets on the network. Hence the performance of evolved programs here is affected by those factors as well.

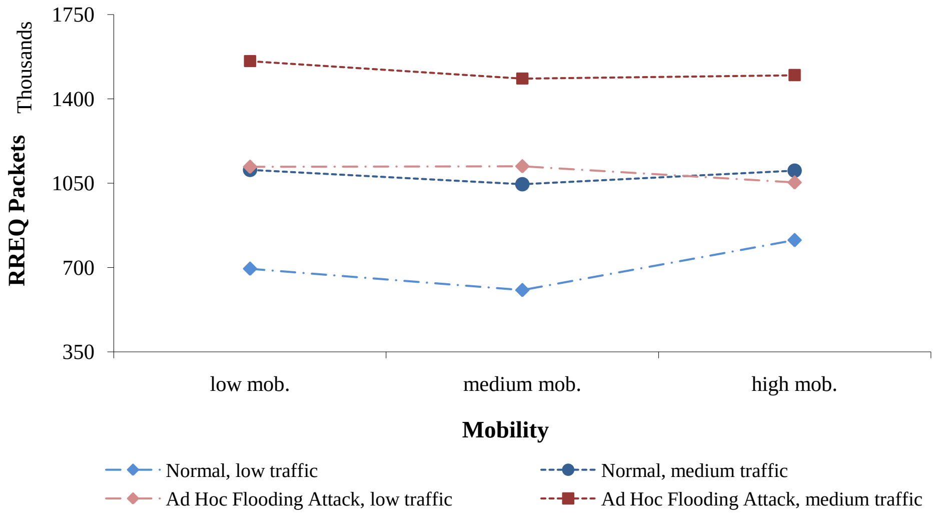


Figure 5.1: Route Request Packets on Simulated Networks

Another factor affecting the performance of GE could be the fitness function. Since the number of normal events is much higher than the number of malicious events in the network, a small improvement in increasing the number of attacks detected could improve the fitness function much more than a small improvement in decreasing the number of false positives. The results also support this idea by evolving programs with high detection rates. This issue could be addressed by improving the fitness function such as assigning different weights to detection rate and false positive rate, or adapting a multi-objective fitness approach. In the subsequent chapters a multi-objective fitness function, which aims to optimize these two objectives simultaneously, is employed. However, here we have chosen the fitness function defined in Section 4.3.2 due to its simplicity.

For each attack type, the evolved programs and the memory usage of each program (detect an attack and call the response function) are given in Table 5.4. Memory usage of each program is calculated according to the formula below:

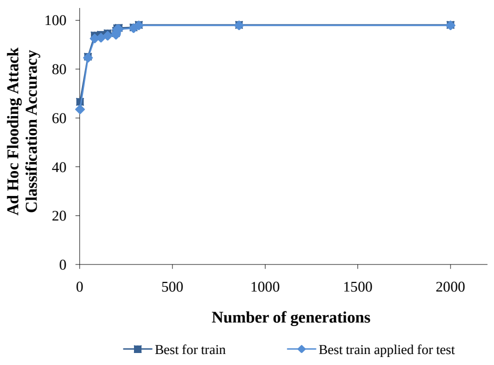
$$memory = program\ binary\ size + variables * sizeof(float) \tag{5.1}$$

Lastly the relation between accuracy of evolved programs for each attack and number of generations in GE algorithm is examined and demonstrated in Figure 5.2. A high fitness value is achieved in 100 generations and afterwards. Furthermore the best program

Attack Type	**Evolved Program**	**Memory Usage**
Flooding	(send_rrepPs + exp(frw_aodvPs - updated_routes * pow(frw_rreqPs, added_repairedroutes))) > no_neighbours	≃7.52kB
Route Disruption	log(exp(invroutes_other)) + recv_aodvPs - percent(exp(send_aodvPs), send_rreqPs) + repaired_routes > recvB_rerrPs	≃7.79kB

Table 5.4: The Intrusion Detection Programs –Best Individuals– Evolved by GE

show a performance on the testing data (network) almost as good as on the training data.



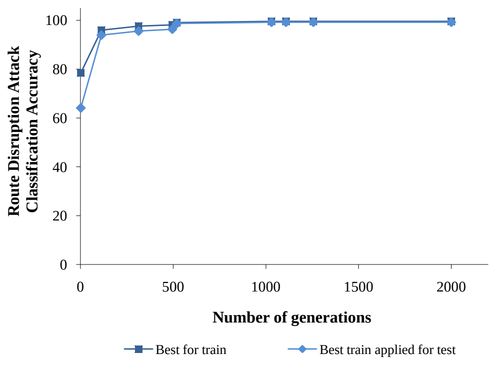


Figure 5.2: GE: Relation Between Classification Accuracy and Number of Generations

5.2.1 Variations in Route Disruption Attack

In this section we investigate how variations in an attack affect the performance of evolved programs. The route disruption attack has been extended to 3 and 5 seconds in the simulations. The attacker sends the same amount of route reply packets but in different time intervals. The attacker achieves his goal but more slowly. He aims to hide his malicious behaviour from the intrusion detection system. So the route disruption attack is implemented in 1, 3, 5 seconds on the same networks and trained separately by collecting data for each of 1, 3, 5 seconds respectively. In each case, the GE framework is applied ten times and the best performing resulting program is returned.

The results for the best individuals are presented in Figure 5.3. It is clear that the false positive rate is increased proportional to the attack distribution time. The best individual with the highest detection rate (100%) is chosen here to compare attack

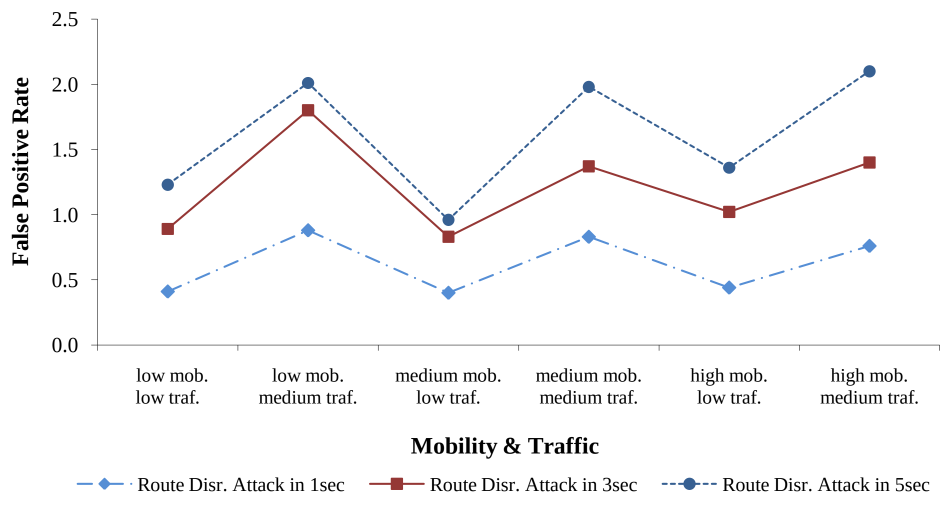


Figure 5.3: The Performance of Evolved Programs on Route Disruption Attack

patterns with each other easily. There are individuals who have lower false positive rate but also a lower detection rate among the evolved programs. Detection rate and false positive rates are effectively traded off.

5.3 The Performance of Genetic Programming

In Table 5.5 the performance of the evolved program (the best individual of the ten runs of the GP algorithm) is shown for each attack type on networks with varying mobility and traffic patterns.

Some conclusions can be drawn from these results. Apparently, route disruption attacks seem to be easier to detect than ad hoc flooding attacks. In all cases but one the detection rate is 100% and the false positive rate is less than 1%. Note that in the case with medium mobility and low traffic perfect detection is not reached, but the false positive rate is low (0.46%). It seems reasonable to suppose that a 100% detection rate can be achieved with a small increase in the false positive rate as observed in our further experiments. The results for ad hoc flooding attacks are slightly “worse”, attaining in almost all cases detection rates higher than 99% while keeping the false positive rate reasonably low. Note that in both attacks the main difficulty seems to come from the traffic load: regardless of the mobility patern, the false positive rate for medium traffic is higher than for low traffic. This is a common characteristic of any detection technique which does not achieve a perfect detection, as the higher the traffic to be analysed, the

Network Scenarios	**Flooding Attack**		**Route Disruption Attack**	
	DR	FPR	DR	FPR
low mobility low traffic	99.81%	0.34%	100%	0.51%
low mobility medium traffic	99.24%	1.94%	100%	0.99%
medium mobility low traffic	99.95%	0.36%	97.06%	0.46%
medium mobility medium traffic	99.89%	1.88%	100%	0.88%
high mobility low traffic	99.79%	0.66%	100%	0.52%
high mobility medium traffic	98.62%	1.83%	100%	0.84%

Table 5.5: The Performance of GP on Simulated Networks

higher the false positives.

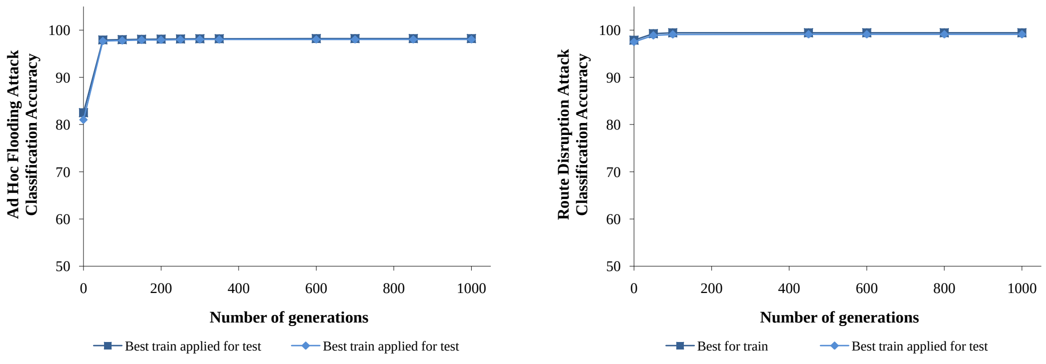


Figure 5.4: GP: Relation Between Classification Accuracy and Number of Generations

The relation between accuracy of evolved programs for each attack and number of generations in GP algorithm is shown in Figure 5.4. While a high fitness value for detection of ad hoc flooding attack is achieved in 100 generations and onwards, it is obtained in early generations for detection of route disruption attack. It is observed that the route disruption attack can be detected by a simple program since it directly violates the routing protocol. Note that the evolution process is carried out offline. Computational complexity of the evolution process is an orthogonal issue to that of computational resource requirements of any evolved program when deployed.

5.4 The Performance of Manual Detection

The performance of hand-coded programs for ad hoc flooding and route disruption attacks is analysed in this section to see if an evolved program could discover MANETs' complex relations and achieve a better performing solution.

In manual detection of an ad hoc flooding attack a threshold-based signature is typically used which simply considers the excessive amount of forwarded route request packets by a node (*frw_rreqPs* > *threshold*). Threshold-based signatures to detect resource depletion attacks in MANETs have already been employed in other approaches [94]. The performance of the signature is evaluated with different threshold values on a network with medium mobility and traffic, and demonstrated in Figure 5.5. It is shown that the fitness value (100-Fitness) has its optimal value at the threshold value three, then it starts increasing. The manual signature (using the threshold value three) is evaluated on networks with varying mobility and traffic patters and demonstrated in Table 5.6. The results show that the manual detection achieves almost a perfect detection rate. However it does not perform well on differentiating benign flooding from malicious flooding, and results in non-negligible false positive rate for networks under high traffic. On the other hand GP and GE decrease the false positive rate with a small amount of decrease in the detection rate. The false positive rate is decreased to almost half that of the manual detection does. The difference becomes more remarkable with the distributed and cooperative intrusion detection programs evolved in Chapter 7. Overall, GP and GE outperform the manual detection for the ad hoc flooding attack by some considerable margin.

An intuitive signature which checks if the number of received route reply packets are consistent with (less than) the number of sent route request packets by a node is employed in the manual detection of route disruption attack (*init_rreqPs* < *recv_rrepPs*). A similar approach has been used in some specification-based IDSs proposed for MANETs [90] which monitor request-reply flow of each routing packet. The results are demostrated in Table 5.6. Programs evolved by GP and GE already show that the route disruption is a simple attack detected by small and relatively simple programs. The highest detection rate and a false positive rate less than 1% could be achieved by manual detection as well. GE decreases the false positive rate a bit more (up to 0.12%), while GP shows almost the same performance as manual detection. A node which

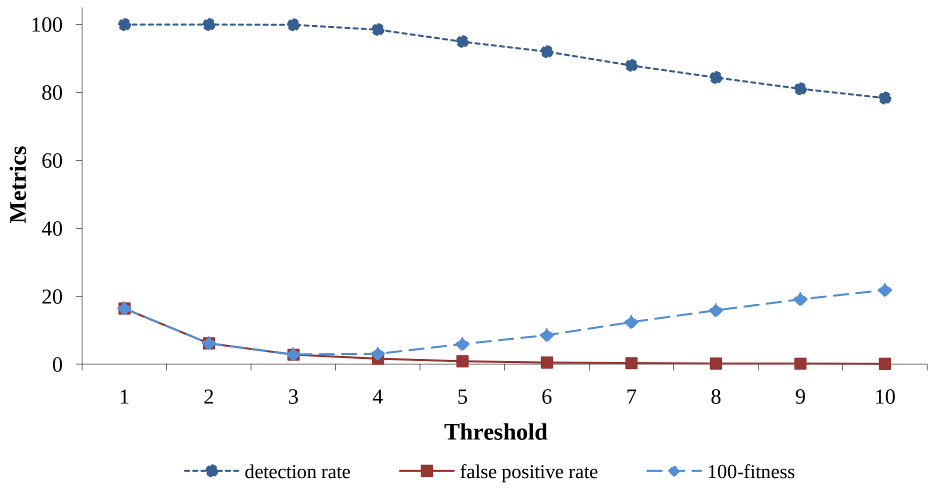


Figure 5.5: The Performance of Manual Detection of Ad Hoc Flooding Attack on Different Threshold Values

compares the number of route reply packets sent and received for each node on the network separately can achieve a better detection performance, but this approach is not very effective in terms of memory. In this research the hand-coded signatures used only the features in Table 4.2 to allow a fair comparison. In conclusion GE shows a slightly better performance in detection of the route disruption attack than manual detection.

There are also other advantages of our approach over other misuse-based and hand-coded approaches proposed in the literature. For example nodes monitor every packet and keep them in its memory for a while in many approaches [62][94]. In addition they usually do it by using promiscuous monitoring which is expensive in terms of power usage. Furthermore, we consider the effect of mobility on attacks in this research. While other approaches generally monitor if a node behaves properly or not without taking into account other factors affecting its behaviour.

Two variants of evolutionary computation techniques, namely Genetic Programming (GP) and Grammatical Evolution (GE), have been evaluated to design intrusion detection programs for known attacks against MANETs so far. It is shown that GP and GE are good at discovering complex relations on MANETs. In these experiments the default parameters (of ECJ [2] and libGE [5]) are used for each technique. However each techniques can show different performance under different parameter settings. In the subsequent section each technique is analysed at their approximate optimal parameters for an unbiased evaluation.

Network Scenarios	**Flooding Attack**		**Route Disruption Attack**	
	DR	FPR	DR	FPR
low mobility low traffic	99.95%	0.62%	100%	0.51%
low mobility medium traffic	99.92%	3.19%	100%	1.00%
medium mobility low traffic	100%	0.98%	100%	0.47%
medium mobility medium traffic	99.96%	3.07%	100%	0.90%
high mobility low traffic	99.91%	0.99%	100%	0.53%
high mobility medium traffic	99.80%	3.39%	100%	0.85%

Table 5.6: The Performance of Manual Detection on Simulated Networks

5.5 The Evaluation of GP and GE on Intrusion Detection

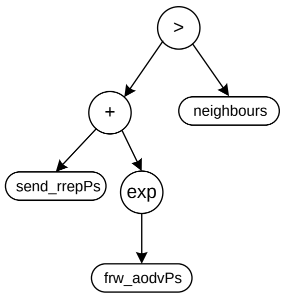
In this section a fair comparison between GP and GE on intrusion detection in MANETs is aimed. Approximations to the optimal parameters for each technique are identified at first and then a fair comparison of these techniques under their optimal parameter settings are made. In order to achieve these goals the steps of a simple Design of Experiments (DoE) methodology in [99] are followed in this research.

Genetic Programming and Grammatical Evolution differ from each other in three fundamental ways [70]. The main difference is the representation of individuals. While individuals are represented as trees in GP, GE employs linear genomes. Secondly, GE performs mapping from the genotype to the phenotype (program) which provides degenerate code as in biological genetic systems. Lastly, GE uses a grammar to dictate legal structures in the phenotypic space [70]. We aim to show if these differences make a statistically significant difference on the performance of detection programs evolved by using these techniques separately.

The experiment results so far show that route disruption attack can be detected easily since it violates the specifications of the routing protocol directly. However the ad hoc flooding attack is a more complex attack and not easily differentiable from normal behaviour of the system due to the dynamic nature of MANETs. Since this seems a challenging problem it can serve as a useful test case for a comparative evaluation of GE and GP.

5.5.1 The Testbed

Firstly a common platform is built for each evolutionary computation algorithm before tuning the parameters. Each algorithm uses the feature set in Table 4.2, the fitness function given in Section 4.3.2, the same training and testing data simulated by ns-2 [7] and BonnMotion [1]. The functions used in order to define the problem in a GP tree are the same as the ones in the GE grammar. Strongly-typed GP is employed here to enforce the rules of the GE grammar in the GP. It facilitates the production of suitably structured code. For example, only relational functions which return Booleans can be placed at the first level of the GP tree by requiring the tree to return a Boolean type. Then relational functions are placed into an if-statement while the individual is translated to a C program. Consequently, it provides a program start with an if-statement as in the GE grammar. Figure 5.6 shows a C statement expanded from the GE grammar and the GP tree corresponding to it.



if((send_rrepPs + exp(frw_aodvPs)) > neighbours)

Figure 5.6: A GP Tree and Corresponding C statement

The size of an individual is constrained by tree depth in GP, and by genome size and wrapping in GE. Maximum tree depth size is one of the important parameters in GP. Maximum genome size in GE, corresponding to maximum tree depth in GP, is also defined in our experiments to evolve individuals in the same size range in each technique. Maximum genome size is computed by building the full tree with maximum tree depth and defining the grammar which builds this tree intuitively. Since our grammar is not complicated, it is estimated easily. However both algorithms behave differently when the size of an individual exceeds the predefined parameter or an invalid individual is created in general. While GP copies the parents of the invalid individual instead of itself, GE assigns the lowest fitness value to the individual.

Koza's ramped half and half initialization in GP and sensible initialization in GE are used to create an initial population. GE sensible initialization is based on Ramped Half and Half Initialization in GP but generates derivation trees of equivalent size [70].

Tournament selection is used to select individuals for recombination. However the libGE library [5] uses an improved version of the tournament selection. That is why it was re-implemented in a standard way. A simple genetic algorithm was used in both techniques to select individuals for replacement. Furthermore elitist approaches were employed in each technique by keeping the best individual of each population.

The details of other parameters used/tuned in our experiments and the methodology to compare GP and GE techniques are given in the subsequent section.

5.5.2 The Design of Experiments

In this methodology the parameters to identify approximations to the optimal parameters for each technique are tuned firstly. However, finding the optimal parameters for an algorithm requires a large-scale experimentation consuming potentially vast computing resources. So the approach which assumes "if an equivalent amount of effort is spent in applying this method to each of the algorithms, it is reasonable to expect the approximations to be similarly close to the absolute optimum for each algorithm and so the comparisons to be fair" [99] is taken here. Therefore we start with finding approximations to the optimal parameters for each technique.

Four independent parameters are used in these experiments: crossover and mutation probabilities, population size, and tournament size. Since it is excessive to run experiments at each possible parameter setting, a three-level full factorial design is used, where each parameter is considered at three levels (referred as low, intermediate and high). Table 5.7 shows the value range of each parameter in these experiments.

The ranges of each parameter are chose large enough to cover all practical values. For crossover probability the highest value is chosen as 0.9 since leaving some part of old population survive to next generation is believed to be good [82]. Mutation helps to avoid being trapped in local extremes by introducing diversity into the system. However,

Parameter	**Value Range**
x_1: crossover probability	[0.1,0.9]
x_2: mutation probability	[0.01,0.5]
x_3: tournament size	[2,9]
x_4: population size	[50,1000]
x_5: max. number of generation	$100.000/x_4$

Table 5.7: The Parameters and Their Ranges

mutation should be used sparingly (In extremis, use of a very high mutation rate cuases the process to degenerate into a random search). A maximum rate of 0.5 will be used here. The pilot study also showed that high mutation rates did not give good results. The parameter for termination criteria of an evolutionary algorithm, namely *generations*, depends on the parameter *population size* in this table to ensure the same number of individuals are created overall in each algorithm. Each algorithm runs twice (with different seeds) for each parameter setting. Hence each algorithm is run totally $3^4 \times 2$ times to estimate the β coefficients in the second linear model below which describes the relationship between the performance of the algorithm, y, and the parameter settings, x$_i$. y shows the fitness value (detection rate - false positive rate) of the evolved intrusion detection programs evaluated on simulated networks. ϵ represents the noise which is the difference between the predicted mean performance of the algorithm and its observed performance.

$$y \ = \beta_0 \ + \sum_i \beta_i x_i \ + \ \sum_i \sum_{j>i} \beta_{ij} x_i x_j \ \sum_i \beta_{ii} x_i^2 \ + \ \epsilon \tag{5.2}$$

The coefficents in the linear model are estimated by using a standard linear regression. Then we apply quadratic programming in order to locate approximations to the optimal parameters. Quadratic programming is an optimization problem which aims to optimize a quadratic function of several variables subject to linear constraints on these variables [8]. Table 5.8 shows the results, the parameter settings which give the best performance for each algorithm.

Finally each algorithm is run one hundred times at the parameter settings in Table 5.8 and compared by applying a statistical hypothesis test of equality. The results show that the mean of GP runs (fitness values) is greater than the mean of GE runs with 95% confidence. However better results by GE which uses steady-state approach have been observed in the experiments presented in Chapter 4. This approach is proposed

Parameter	GP	GE
crossover probability	0.1	0.9
mutation probability	0.37	0.5
tournament size	8	7
population size	50	1000
max. number of generation	2000	100

Table 5.8: The Approximate Optimal Parameters for Each Algorithm with Simple Approach

to reduce the effect of invalid individuals in GE. That is why the same methodology is applied to compare two techniques with the same parameter values in Table 5.7, but with a steady-state approach. The standard steady-state approach which replaces the worst individual of the preceeding population is employed. Even if this individual is better than the new individual, it will be replaced regardless of its better score. The parameter settings which give the best performance for each algorithm are given in Table 5.9.

Parameter	GP	GE
crossover probability	0.1	0.9
mutation probability	0.01	0.5
tournament size	50	2
population size	9	514
max. number of generation	2000	194

Table 5.9: The Approximate Optimal Parameters for Each Algorithm with Steady-State Approach

After running each algorithm one hundred times with the parameter settings in Table 5.9, hypothesis testing is applied for comparison. The results show that if a steady-state approach is employed, the mean of GE runs (fitness values) is greater than the mean of GP runs with 90% confidence. It is reasonable to say that it was invalid individuals who affect the performance of GE (slow down the evolution) in the simple approach. Overall when the results of the best GP version (with simple approach) is compared with the results of the best GE version (with steady-state approach), GP shows a better performance.

The performance of programs evolved with approximate optimal parameters on simulated networks (under medium mobility and medium traffic) is shown in Table

5.10. The results obtained in the previous sections are also represented in the table. It is observed that programs evolved with approximate optimal parameters decreases the false positive rate with a small decrease in the detection rate.

Algorithm	**Detection Rate**	**False Positive Rate**
GP	99.89%	1.88%
GP with Optimal Parameters	98.24%	1.12%
GE	99.86%	2.00%
GE with Optimal Parameters	98.59%	1.58%

Table 5.10: The Performance of Programs Evolved with Approximate Optimal Parameters on Simulated Networks

In this chapter it is shown that evolutionary computation techniques discover complex relations of MANETs such as mobility. The main hypothesis of this research is supported with the evaluation results. Furthermore it is shown that GP and GE both show good results in intrusion detection in MANETs. They show their optimal performances under different parameter settings. GE performance is clearly affected by invalid individuals. That is the reason steady-state approach is more suitable for GE.

Chapter 6

Trade-offs in Intrusion Detection in MANETs

This chapter presents a new approach to intrusion detection in MANETs. Section 6.1 explains the need for a power-aware intrusion detection system in this resource-constrained environment. Section 6.1.1 introduces the simulation environment to estimate the energy consumption of programs. The energy consumption of programs evolved using GP is analyzed in Section 6.3 and it is observed that different trade-offs can be made between the classification accuracy and the energy consumption of the evolved programs. In order to discover these tradeoffs multi-objective evolutionary computation, explained in Section 6.2, is employed. Finally the evaluation results are given and discussed in Section 6.4.

6.1 Introduction

Nodes on MANETs can vary from hand held devices such as PDAs, cell phones, and the like to laptops that have different resource and computational capacities. Moreover they usually run on battery power to provide mobility. They are generally constained in terms of computational capabilities, memory, communication bandwidth, and especially power. In the case of sensor networks, the power issue may become acute. Some microcontrollers used in wireless nodes (especially for sensor networks) and their capabilities are given in Table 6.1 [76]. The first element in the table is extremely constrained, even unable to support the de-facto operating system for sensor nodes TinyOS [76]. However there are more powerful devices such as the last two elements in the table, which are widely used in hand-held devices (PDAs). Other devices shown in Table 6.1 are defined as normal devices which are resource-constrained but powerful enough to host a complex application. They are the most common devices used in sensor networks [76].

The variety of mobile nodes often with scarce resources affects the proper working of intrusion detection systems running on these nodes. For instance, IDS agents might not

Model	Frequency (MHz)	Word size (bit)	RAM memory	Inst. memory	Power:awake (mA)	Power:sleep
PIC12F675	4	8	64 KB	1.4 KB	2.5	1nA
PIC18F6720	20	8	4 KB	128 KB	2.2	1 μA
MSP430F14x	4	16	2 KB	60 KB	1.5	1.6 μA
MSP430F16x	8	16	10 KB	48 KB	2	1.1 μA
ATmega128L	8	8	4 KB	128 KB	8	15 μA
PXA271	13(416)	32	256 KB	32 MB	31-44	390 μA
ARM920T	180	32	512 KB	4 MB	40-100	40 μA

Table 6.1: Example Microcontrollers in Wireless Networks [76]

be able to process every packet/alert due to limited resources. This is why efficiency is as important as effectiveness for intrusion detection in mobile networks. Hence the detection algorithm must take into account limited resources.

Two different power-saving strategies have been used in MANETs routing [64]: local and global. In local strategies a node can save his energy by controlling its transmission power enough to reach the receiving node, or switching to a sleep state when it is idle. Global strategies aim to maximize the network lifetime by balancing the energy consumption across the network [64]. The network lifetime is defined as the duration of time until the first node fails due to the battery depletion [53] and affects the network performance. In most of the routing protocols for MANETs the shortest path is the only metric used to choose a path between end-points. In [81] power required for transmitting/receiving a packet on a route is also taken into account. The approach in [64] also uses the remaining energy on nodes in order to maximize the network lifetime. It is stated that if remaining energy is not considered then more nodes will suffer battery depletion earlier than otherwise. In [61] the remaining energy (the ratio of the full-charge battery capacity to the remaining battery capacity of nodes) is considered with their transmission power for a power-aware source routing protocol. The approach in [25] also uses the remaining energy for routing in a static wireless network and shows that the traffic should be routed through a path which balances the energy consumption in the network instead of minimizing the required power on this path.

Currently proposed intrusion detection approaches for MANETs generally do not put emphasis on power issues. Some approaches propose a hierarchial intrusion detection architecture where the network is divided into manageable small groups such as clusters and zones. In this architecture some nodes bear more IDS responsibility than other nodes in a group. For example while all nodes are responsible for local intrusion detection in a cluster, clusterheads carry out global intrusion detection (network-based).

In some approaches [87][49] cluster heads are chosen based on some criteria such as connectivity, energy remaining, and the like. However the selection mechanisms proposed for cluster-heads are neither investigated in detail nor implemented in those approaches. In other approaches such choices are made randomly for security reasons [42][105]. Using central management points to carry out computationally intensive tasks like data mining [83] is another method proposed in intrusion detection on MANETs.

Limited resources (especially power) in intrusion detection on MANETs is only considered in a few approaches. In [53] monitoring nodes are selected based on their connectivity and battery power with a voting mechanism. The weakness of this approach as stated in the paper [53] is that many control messages are sent for voting, since monitoring nodes need to be updated frequently. In this approach only monitoring nodes carry out intrusion detection, so a monitoring node listens to all traffic in its transmission range (not only that destined for itself), which is expensive in terms of power. In [85] another power-aware approach which determines network monitors based on available power in nodes is proposed. While all nodes carry out host monitoring, network monitoring is distributed to the nodes powerful enough.

In this research we investigate evolving intrusion detection programs which take into consideration the capability of nodes running these programs as well as their classification accuracies. We aim to explore trade-offs between these functional and non-functional properties of programs by using multi-objective evolutionary computation techniques. Since power is one of the most crucial resources on mobile networks, both classification accuracy and energy consumption of programs are considered as objectives during the evolution process. This approach is different to proposed IDSs in the literature which distribute some functional tasks among nodes usually based on randomness or consider only the capabilities of nodes carrying out global detection. In this research the capabilities of each node on the network, regardless of whether they carry out local or global detection, are taken into account while designing an IDS for MANETs. Programs demonstrating different trade-offs among these objectives are evolved in this research. So a node might choose one of these programs based on its remaining battery power and play a part in intrusion detection.

6.1.1 Power Simulation

Wattch [20] (integrated with SimpleScalar [9]) is used to estimate the energy consumption of programs evolved in this research. Wattch is an architectural simulator for analyzing and optimizing microprocessor power dissipation for specific architectures. It is claimed it achieves accuracy within 10% of their estimates as verified using industry tools [20].

Wattch estimates CPU power consumption using the power model based on dynamic power consumption (P_d) which is the main source of power consumption in CMOS microprocessors. The power model is defined as follows:

$$P_d = CV_{dd}^2 af, \tag{6.1}$$

where C is given as the load capacitance, V_{dd} as the supply voltage, a as the activity factor, f as the clock frequency. Where V_{dd} and f are dependant on the process technogoloy, C is estimated. The activity a represents how often clock ticks lead to switching activity on average. It is an estimated value (between 0 and 1) based on the benchmark programs executed or is assumed to be 0.5 as a default.

In this research Sim-Wattch, an integration of Wattch with the SimpleScalar architectural simulator, is used. SimpleScalar measures the dynamic characteristics of the hardware model (such as which units are accessed per cycle) and the performance of the software running on it [9]. Here the execution of a program is simulated on the PISA (the Portable Instruction Set Architecture) architecture and its energy consumption is estimated.

6.2 Multi-Objective Evolutionary Computation

Multi-Objective Optimisation (MOO) aims to optimise two or more, often conflicting objectives simultaneously. The Multi-Objective Optimization Problem is defined in [?] as "the problem of finding a vector of decision variables which satisfies constraints and optimizes a vector function whose elements represent the objective functions. The term *optimize* is defined as finding such a solution which would give the values of all the objective functions acceptable to the decision maker since the objectives are usually in conflict with each other".

The solution to multi-objective optimisation generally is not unique, it is the set of optimal solutions called the Pareto set. An objective vector x is said to dominate another objective vector y (x $\succ$ y) if no criterion of x is greater than the corresponding component of y and at least one criterion is less (lesser values are preferable). Formally:

$$x \succ y : if\ x_i \leq y_i\ for\ each\ i\ and\ x_i < y_i\ for\ some\ i \tag{6.2}$$

The Pareto front comprises the solutions that are not dominated by any other individual. In other words, it includes the optimal solutions (non-dominated) which represent different trade-offs among objectives. Figure 6.1 shows an example of non-dominated solutions lying on the Pareto front and a solution dominated by both point A and point B.

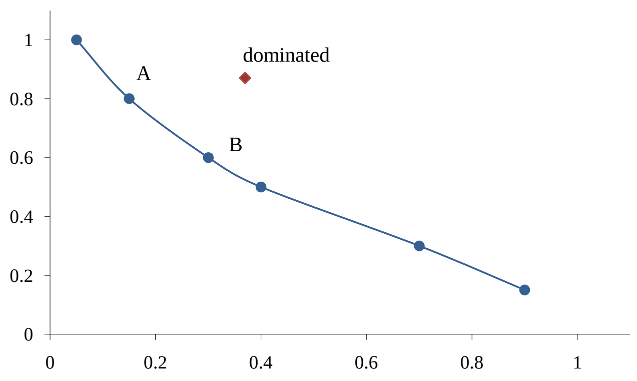


Figure 6.1: An Example of Pareto Front

Multi-objective evolutionary computation (MOEC) allows us to combine multi-objective optimisation with evolutionary search. The use of evolutionary algorithms to solve problems with multiple objectives has been motivated mainly because of the population-based nature of evolutionary algorithms which allow the generation of several elements of the Pareto optimal set in a single run [27].

6.2.1 Strength Pareto Evolutionary Algorithm (SPEA2)

There are two methods to define a multi-objective optimization problem in evolutionary computation: weight-based and pareto-based. In weight based approaches a single fitness function is defined and the relations among objectives are indicated by weights. However it is difficult to determine the weights. Furthermore this approach does not allow us to compare different objectives and trade-offs among them. It summarises many different trade-offs as effectively equally good. That is why SPEA2 [108], which is one of the most popular Pareto-based MOEA algorithms, is employed here. An implementation of SPEA2 which is an extension to ECJ [2] is utilised in this research.

In SPEA2 there is a regular population and a fixed size archive. The archieve contains the non-dominated individuals among all candidate solutions considered so far. The main steps of the algoritm [108] are given in Table 6.2.

Input:	N population size
	$\overline{N}$ archive size
	T maximum number of generations
Output:	A nondominated set
Step 1:	**Initialization** generate an initial population (P_0) and create an empty archive ($\overline{P_0} = \emptyset$).
Step 2:	**Fitness assignment** calculate fitness values of individuals in P_t and $\overline{P_t}$.
Step 3:	**Environment selection** copy all non-dominated individuals in P_t and $\overline{P_t}$ to $\overline{P_{t+1}}$. If size of $\overline{P_{t+1}}$ exceeds $\overline{N}$ then reduce $\overline{P_{t+1}}$ with the truncation operator, otherwise if it is less than $\overline{N}$ then fill $\overline{P_{t+1}}$ with dominated individuals in P_t and $\overline{P_t}$.
Step 4:	**Termination** if termination criteria ($t \geq T$) is satisfied then set A to the non-dominated individuals in $\overline{P_{t+1}}$ and stop.
Step 5:	**Parent Selection** perform tournament selection with replacement on $\overline{P_{t+1}}$ to fill the mating pool.
Step 6:	**Variation** apply variation operators to the mating pool and set P_t to the new population. Increase generation number ($t = t + 1$) and go to *Step 2*.

Table 6.2: The Main Steps of SPEA2 Algorithm

The fitness of an individual i in SPEA2 is defined by two elements: the raw fitness R(i) and the density D(i).

$$F(i) = R(i) - D(i) \tag{6.3}$$

The raw fitness of an individual i is determined by the strengths of its dominators in both archive and population. It is aimed to be minimized. The density is calculated by an adaption of the k-th (k= $\sqrt{N + \overline{N}}$) nearest neighbour method which gives the k-th nearest neighbour (σ_i^k) to the individual.

$$R(i) = \sum_{j \in P_t + \overline{P_t}, j \succ i} S(j) \tag{6.4}$$

$$D(i) = \frac{1}{\sigma_i^k + 2} \tag{6.5}$$

A simple MOO problem is given as an example here. We evolved programs using MOEC techniques to discover trade-offs between two conflict objectives in intrusion detection: detection and false positive rate. The optimal solutions for both metrics might not be discovered by using the fitness function described in Equation 4.3.2. The fitness of an individual can be high due to high detection rate or low false positive rate, or both. For example, the fitness of a program with 90% detection rate and 2% false positive rate (Fitness = 90 - 2) is the same with a program which has the perfect detection rate and 12% false positive rate (Fitness = 100 - 12). Therefore, the fitness of an individual (evolved program) is represented by two separate objectives here: detection rate and false positive rate and, a set of optimal solutions is obtained by using MOEC techniques. Figure 6.2 shows the Pareto fronts for each attack, which demonstrate the optimal solutions at the end of 500 generations. Each chart shows detection rate versus false positive rate which are metrics to be minimized simultaneously. As it is seen there is a clear trade-off between these two objectives: while false alarm decreases, detection rate decreases too. In this research MOEC techniques are employed in order to discover more complex relations (functional and non-functional properties of a program) on intrusion detection in MANETs as presented in the subsequent sections.

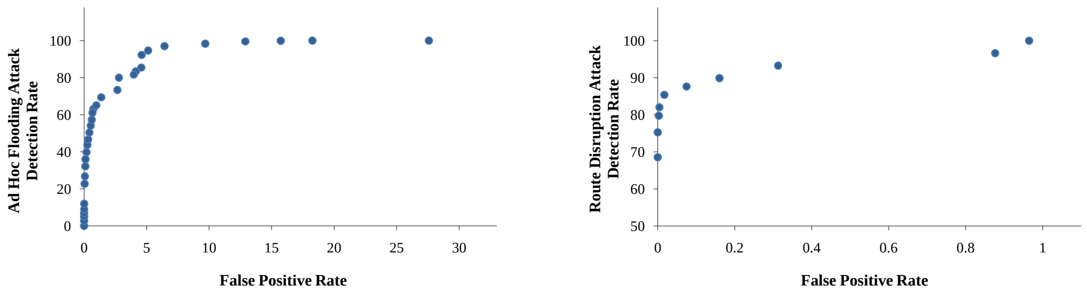


Figure 6.2: Trade-offs Between Detection Rate and False Positive Rate for Attacks Ad Hoc Flooding and Route Disruption

6.3 Analysis of Power Consumption of Evolved Programs

Firstly the power consumption of the programs evolved by using GP to detect ad hoc flooding and route disruption attacks on MANETs and represented in Chapter 5 are analyzed in this section. To evaluate a program's energy consumption, the execution of each program needs to be simulated. For that reason each individual

(GP Tree) is converted to a C program and written to a file. In the transformation process from a GP tree to a C program, the functions used by the individuals and not included in the standard C library (*e.g.* percent function) are defined as macros. After the C file is created, it is compiled and run on the Sim-Wattch to simulate the execution of the program on the PISA architecture and estimate its energy consumption.

The best individuals of ten runs with their energy consumptions are given in Figure 6.3. This figure shows that while classification accuracy is high, energy consumption of the program gets higher as well for ad hoc flooding attacks. On the other hand, this relation is not quite straightforward for route disruption attacks. Analyzing the best individuals evolved for route disruption attack shows that it can be detected by small programs (with simple expressions) which have a tendency to consume lower energy.

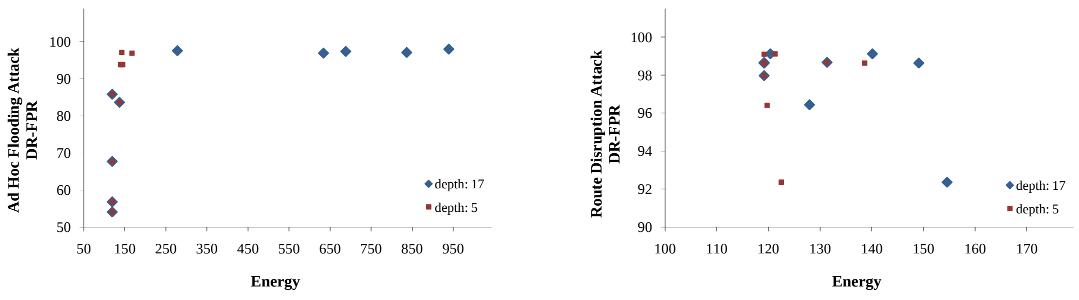


Figure 6.3: Classification Accuracy and Energy Consumption of the Optimal Evolved Programs

Furthermore, since the size of the programs can affect their energy consumption, we conduct experiments to evolve programs with different tree depths (17, 5) (the maximum size of the individuals (trees) evolved in GP). The same GP parameters listed in 4.3 and the same fitness function given in Equation 4.3.2, are used to evolve programs. The effect of program size on evolved programs' detection ability and energy consumption can also be seen in Figure 6.3. As it has been stated before, the evolved programs for route disruption attacks are small. There are programs which can achieve the same performance on detecting this attack in different program sizes in Figure 6.3. However, program size forces the programs to be smaller which can result in less energy consumption. Hence the programs evolved for this attack with the same detection ability but consuming different energy levels can be seen in the figure. The results are more dramatic for the ad hoc flooding attack. Good performance on detection of this attack can be achieved with small-sized programs as well. Nevertheless programs with bigger program size and accordingly higher energy consumption show a slightly better detection performance.

The energy consumption of programs is mainly affected by the program size and the functions used. In GP there is a phenomenon called *bloat* where the size of individuals (program size) in a GP population increases dramatically over the duration of a run, largely due to redundant code [79]. The effect of bloat is also noticed in our experiments with some larger individuals exhibiting redundancy. Fortunately SPEA2 reduces the bloat where a larger individual (with accordingly higher energy consumption) will only survive if it makes an improvement over the existing archive in at least one objective [98]. It should be noted that small programs do not necessarily consume low energy, a program that uses expensive functions (such as multiply) could have higher energy consumption than a program with greater size.

These experiments demonstrate that different trade-offs can be made between classification accuracy and energy consumption of programs, and encourage us to find acceptable trade-offs between these objectives. A multi-objective evolutionary computation technique is employed to discover these trade-offs, as explained in the next section.

6.4 Discovering Trade-offs in Intrusion Detection Programs

Multi-objective evolutionary computation techniques are employed in order to optimise the following three objectives in our experiments: detection rate, false positive rate, and energy consumption of the program. The individual takes part in the evolution process and survives in the next generations based on its performance on these objectives. A multi objective function of an individual is defined below. These three objective are to be maximized simultaneously.

$$f_1 = no.\ of\ attacks\ detected/no.\ of\ attacks \tag{6.6}$$

$$f_2 = 1 - no.\ of\ false\ events/no.\ of\ normal\ events \tag{6.7}$$

$$f_3 = 1/energy\ consumption \tag{6.8}$$

The conceptual schema of the experiments is demonstrated in Figrure 6.4.

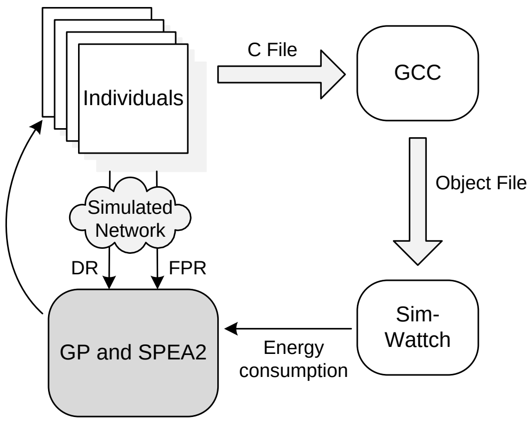


Figure 6.4: Simplified Schema of Experiments

6.4.1 Experiment 1: Attack-specific Intrusion Detection Programs

We firstly evolved intrusion detection programs for ad hoc flooding and route disruption attacks separately by using MOEC techniques. The parameters used are the same as in Table 4.3 except the population size (150) and SPEA2 archive size (100).

Figure 6.5 shows the optimal solutions found for ad hoc flooding attack at the 1000th generation. It shows the conditioning plots of detection rate (DR) versus false positive rate (FPR) which are produced conditional on the energy consumption of programs. Each chart shows the value DR and FPR of programs whose energy consumption fall in one of the intervals shown in the top of the figure. For example the top-right chart shows the programs whose energy consumption falls in the highest energy consumption and the bottom-left chart shows the programs with the lowest energy consumption. These charts show that the programs more close to the optimal solution (high detection rate and low false positive rate) consume higher energy consumption, as expected. In programs with lower energy consumption the false positive rate increases noticably. Overall, the trade-offs falling in the second biggest energy consumption interval (the chart in the top-middle) could be more optimal and acceptable, since it include points which achieve the high classification accuracy with less energy consumption.

For the route disruption attack, programs closer to the optimum solution which have higher detection ability and lower energy consumption are achieved by using MOEC techniques. Moreover we have compared energy consumption of programs which have a high-accuracy detection ability with the programs evolved using GP in Figure 6.3. It is observed that programs with lower energy consumption stands out in the results

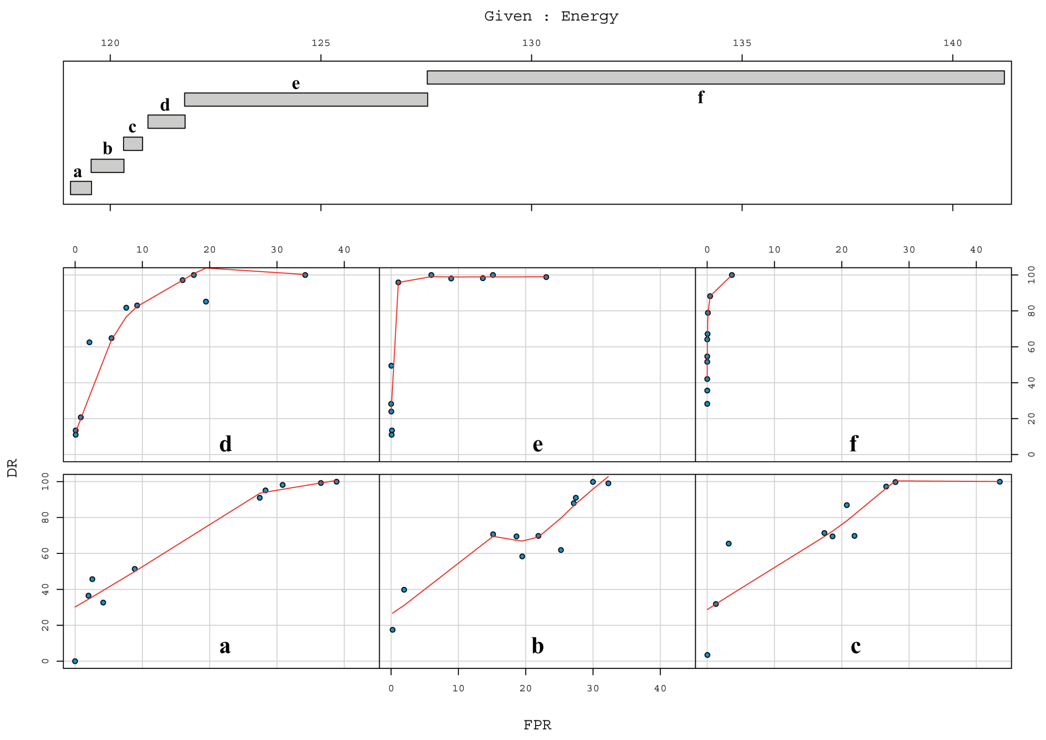


Figure 6.5: Coplots for Programs Evolved for Detection of Ad Hoc Flooding Attack with Three Objectives

obtained by MOEC techniques. Especially for the ad hoc flooding attack, energy consumption is significantly reduced. It could also be the effect of reduced bloat by using SPEA2.

6.4.2 Experiment 2: Multi-attack Intrusion Detection Programs

In this part we evolve programs to detect ad hoc flooding and route disruption attacks together by using MOEC techniques. We aim to investigate if it is better to evolve one program to detect both attacks or evolve two programs each using half the resource usage. The following multi-fitness function which evaluate the performance of program on both attacks, and its energy consumption is employed.

$$f_1 = \frac{(detection\ rate_{flooding} + detection\ rate_{r.disruption})}{2} \tag{6.9}$$

$$f_2 = 1 - \frac{(false\ positive\ rate_{flooding} + false\ positive\ rate_{r.disruption})}{2} \tag{6.10}$$

$$f_3 = 1/energy\ consumption \tag{6.11}$$

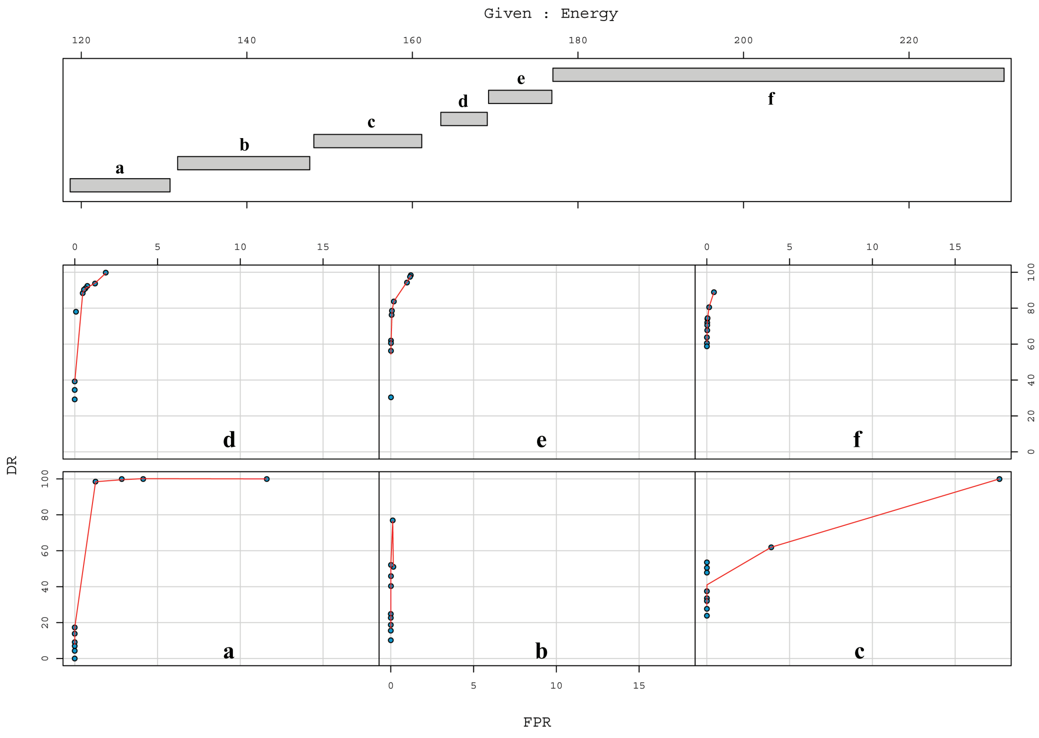


Figure 6.6: Coplots for Programs Evolved for Detection of Both Attacks Together with Three Objectives

Figure 6.6 shows the conditioning plot diagrams for three objectives on detection of both attacks together. The programs with lower false positive rate generally consume higher energy consumption. However good results (closer to the classification accuracy of programs consuming high energy consumption) are also observed in the lower energy consumption intervals. In overall the results demonstrate that a detection program for both attacks can be more energy-efficient than two programs which detect these attacks separately, although it does not show high classification accuracy as the two programs do separately. In the results of thirty runs, there is no program evolved for detecting both attacks which has false positive rate less than 2% (with high detection rate) simultaneously. There is a trade-off to be made based on the requirements of the MANET application used. Some good results are demonstrated in Table 6.3.

Flooding Attack		**Route Disruption Attack**		**Resource (Wattch Units)**
DR	FPR	DR	FPR	
98.80%	1.90%	100%	2.97	≃131
97.71%	1.75%	100%	2.48	≃169
98.80%	1.86%	97.78%	2.90	≃174

Table 6.3: The Performance of Some Programs for Detection of Both Attacks Together

Table 6.4 shows some example programs (with high performance) evolved using MOEC. (There are many other programs on the Pareto front which have different trade-offs.)

Attack Type	**Evolved Program**	**DR**	**FPR**	**Energy Usage**
Flooding	(frw_aodvPs * frw_aodvPs) > (4log(neighbours) + 5updated_routes)	99.79%	1.63%	≃142
Route Disruption	((2updated_routes - 2recv_aodvPs + active_routes) * recv_rrepPs > (recv_aodvPs + updated_routes)	100%	0.85%	≃122
Both	init_rreqPs < (recv_rrepPs + sin(recv_rrepPs + log(log10(frw_aodvPs - updated_routes))) - cos(added_repairedroutes - log10(frw_aodvPs - updated_routes)) + log(frw_aodvPs - updated_routes))	99.40%	2.44%	≃131

Table 6.4: Example Programs Evolved by MOEC for Each Attack

These experiments show that different trade-offs could be made between classification accuracy and energy consumption. The trade-offs are discovered by using a MOEC technique. It is shown how in some circumstances a multiple objective approach provides a more effective means of searching the trade-off space. It is likely that for some types of networks (*e.g.* sensor networks) the ability to make good trade-offs will be particularly important. Programs with almost the same classification accuracy (as obtained by

GP) but with lower energy consumption are evolved by using MOEC techniques. More importantly a set of solutions showing different trade-offs is obtained. So programs showing different trade-offs could be distributed to nodes based on their energy level. A final choice between solutions making different trade-offs rests with the designer.

Programs detecting ad hoc flooding and route disruption attacks together (in a program) are also evolved in this section to see if energy consumption for detection of these two attacks separately could be minimized. The results show that although these programs consume less energy than two separate detection programs do, they do not show as high classification accuracy as the separate targetted programs achieve. It is a decision to be made based on the application of MANET. It is believed that evolving programs to detect similar attacks (showing similar consequences on the network) such as DoS attacks together could give more promising results in terms of energy consumption than evolving separate programs for each attack, since they are more likely to have similar signatures.

This work is unusual in that it trades off security performance (detection and false positive rates) against resources (power). The inherent complexity of MANET operations makes it difficult to see how IDS programs with optimal trade-offs could be obtained by standard system development practices. To conclude, an optimisation based approach seems a natural and effective candidate for the problem.

Chapter 7

Distributed and Cooperative Intrusion Detection on MANETs

The only thing that will redeem mankind is cooperation.

Bertrand Russell

This chapter investigates the evolutionary synthesis of a suitable intrusion detection system for MANETs. An architecture *cooperative detection in neighbourhoods* is investigated using GE and GP techniques in Section 7.2.1 and Section 7.2.2 respectively and evolved programs are compared with the local detection results presented in Chapter 5. The energy and bandwidth consumption of these programs are also considered, and optimal trade-offs between intrusion detection ability and resource usage (energy, bandwidth) of evolved programs are discovered using multi-objective optimization techniques and presented in Section 7.2.3.

7.1 Introduction

So far it is assumed that each node in the network carries out monitoring and detecting of malicious activities. Every node has an intrusion detection agent and detects network attacks locally. Hence the evolved programs are distributed to each node on the network. In this chapter further intrusion detection architectures suited to this distributed and resource-constrained environment are explored.

Intrusion detection architectures on MANETs can be classified into two main groups: local detection, and distributed and coopeative detection. In local detection, every node in the network has an IDS agent and detect attacks on their own without collaborating with other nodes. Hence the evolved programs are distributed to each node. However a node on a MANET can only see a portion of the network: the packets in its radio range and the packets which it sends or receives. So network attacks (network scans, distributed attacks, etc.) cannot be detected with such partial network data on a local node in this environment. Moreover, not every node in the network is capable of performing intrusion detection, due to limited resources.

In a distributed and cooperative architecture each node has IDS agents as in the local detection architecture, but they can also communicate with other nodes to exchange information, to reach decisions and to agree on responses. One of the most cooperative detection architectures proposed in the literature is based on hierarchy among nodes. Distributed IDS agents (nodes) are generally divided into small groups such as one-hop away nodes, neighbouring nodes enabling them to be managed in a more efficient way. For example in one-hop away nodes a node and its immediate neighbour nodes build a group. However a node could be a member of more than one group based on the grouping algorithm in this architecture. In a hierarchical architecture, the network is divided into groups such as clusters, zones where some nodes (cluster heads, interzone nodes etc.) have more responsibility (providing communication with other clusters, zones) than other nodes in the same cluster. It is the same from intrusion detection point of view: each node in the cluster carries out local detection while cluster heads and interzone nodes carry out global detection. The biggest drawback of this architecture is the high cost of building and maintaining the hierarchy in a highly dynamic environment. Message sending and receiving is also very expensive in terms of energy consumption on MANETs. Communication between these IDS agents is provided either by exchanging data directly or by use of mobile agents.

In this research we explore an intrusion detection architecture suitable to the distributed and resource-constrained environments of interest. Two intrusion detection architectures are investigated in this research: local detection and cooperative detection in the neighbourhood. A distributed and cooperative intrusion detection architecture where nodes communicate with their immediate (one-hop away) neighbours to reach decisions is proposed. The following question is explored: "Is it possible to increase the effectiveness of an intrusion detection system by collaboration with the IDS agents in its neighbourhood?". The efficiency of this architecture in terms of bandwidth and energy usage is also explored and compared with that of local detection. The trade-offs between the intrusion detection ability, the enery usage of programs, and the number of neighbours in cooperation are discovered using multi-objective optimization techniques and demonstrated.

7.2 Intrusion Detection Architectures in MANETs

In this section the intrusion detection architecture *cooperative intrusion detection in neighbourhood* is investigated by employing grammatical evolution, genetic programming, and multi-objective optimization techniques and compared with the performance of *local detection*.

7.2.1 Cooperative Detection in Neighbourhood by GE

It is shown that ad hoc flooding attack can be detected effectively by using evolutionary computation techniques in Chapter 5. By nature an ad hoc flooding attack is a distributed DoS attack and floods almost every node in the network with broadcast RREQ packets. When a node believes that someone is attacking (flooding) him, he can support his judgement with the help of his neighbour nodes since he gets RREQ packets through them. Accordingly we believe that cooperative intrusion detection with neighbour nodes can increase the fitness value. So we extend the experiments to evolve a distributed and cooperative detection program in which a node asks for information from its neigbour nodes to reach a decision.

Extending the Grammar for Cooperative Detection

The grammar introduced in Table 4.4 is extended in order to evolve a cooperative detection program for ad hoc flooding attack as below.

The main difference here from the grammar for local detection in Table 4.4 is that it allows using features from neighbouring nodes to make a decision. Each neighbour execute the evolved statement and send its result to the main node. These results from each neighbour are added together at this node. Moreover while the local detection grammar returns an if statement, the cooperative detection grammar returns if or if-else statement as shown below. The logic behind this is that we can evolve programs for each node to detect attacks locally where we can, and ask more information from its neighbour nodes otherwise. Other primitives used in the grammar are the same.

Number of runs(=30) is increased in these experiments. We aim to see the best cooperative detection program evolved in those runs and compare them with the best local detection program evolved in thirty runs. To compare these results fairly, we increase the number of runs. The same parameters except the generations(=4000)

```
S = <code>
<code> ::= if(<cond>) {raise_alarm()} |
           if(<cond>) {raise_alarm()} else {<code>}
<cond> ::= <cond><set-op><cond> | <expr-list><relop><expr-list>
<expr-list> ::= <expr><op><expr> |
           the sum of (<expr> <op><expr>) from each neighbour
<expr> ::= <expr><op><expr> | (<expr> <op><expr>) |
           <pre-op>(<expr>) | <pre-op2>(<expr>) | <var> |
           <neighvar> | <const>
<op> ::= + | - | / | *
<pre-op>::= sin | cos | log | ln | sqrt | abs | exp | ceil | floor
<pre-op2>::= max | min | pow | percent
<rel-op>::= < | ≤ | > | ≥ | == | !=
<set-op> ::= and | or
<var> ::= The features given in Table 4.2
<neighvar> ::= The features obtained from neighbours
<const> ::= 0.<digit><digit>
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Table 7.1: The BNF Grammar Used for Cooperative Detection in Neighbourhood

in Table 4.5 are used. Since the grammar is more complex, we increase the number of generations until termination. The performance of the best program evolved is demonstrated in Table 7.2. Detection rate and false positive rates are used to evaluate its performance. It is also compared with the best local detection program evolved in Chapter 5. The results show that cooperative detection algorithm can achieve lower a false positive ratio (0.06%-0.31%) compared to systems employing local detection only.

However, all the best evolved programs of each run return an if statement which uses information only from neighbour nodes. However, a program that detects attacks locally where it can is more desirable to preserve limited bandwidth. Furthermore, message sending and receiving is very expensive in terms of energy consumption. It is observed that the grammatical evolution algorithm tends to evolve simplified programs. For that reason GE algorithm is forced to evolve a program in the structure *Cooperative Detection 2* by changing the grammar.

GE is run thirty times using a new grammar (*Cooperative Detection 2*). The results show that the false positive ratio is significantly reduced. Furthermore the communnication workload between nodes is reduced approximately 70% compared to the best

Algorithm 1 Local Detection

```
if condition satistifed then
  raise alarm
end if
```

Algorithm 2 Cooperative Detection-1

```
if condition satistifed then
  raise alarm
else
  Local Detection
  or
  Cooperative Detection
end if
```

Algorithm 3 Cooperative Detection-2

```
if local condition satistifed then
  {suspect locally that there might be a malicious activity happening}
  if condition satisfied then
    {reach on a decision based on the further information coming from neighbour
    nodes}
    raise alarm
  end if
end if
```

evolved program with the grammar *Cooperative Detection 1* where a node asks for information from its neighbours at each time interval. However the best program evolved below by the grammar *Cooperative Detection 2* consults its neighbour nodes only when it believes there is a chance of malicious activity on the network.

We here presented the performance of the program with the minimum false positive rate. The program decreases the false positive rate with a small decrease in detection rate. Different programs with different trade-offs between detection rate and false positive rate are also seen in the results. For example another good evolved program results in a lower false positive rate than local detection without decreasing the detection rate. However the false positive rate of the program is a slightly higher than the best program evolved with the grammar *Cooperative Detection 1*. On the other hand the program whose results presented in Table 7.2 achieves lower false positive rate than the best program evolved with the grammar *Cooperative Detection 1*, but with a small decrease in the detection rate.

In conclusion, cooperative intrusion detection programs which achieve lower false positive rates than local detection are evolved by using GE. Furthermore the interaction between IDS agents is investigated and is reduced enormously by improving the BNF grammar of the problem. GE provides a great flexibility in changing the representation

Network	Local Detection		Coop. Detection-1		Coop. Detection-2	
	DR	FPR	DR	FPR	DR	FPR
low mobility low traffic	99.81%	0.29%	99.62%	0.23%	99.29%	0.20%
low mobility medium traffic	98.54%	1.72%	99.48%	1.58%	98.65%	1.43%
medium mobility low traffic	99.86%	0.36%	99.86%	0.22%	99.86%	0.28%
medium mobility medium traffic	99.86%	2.00%	99.61%	1.69%	99.44%	1.32%
high mobility low traffic	99.96%	0.66%	99.49%	0.60%	99.32%	0.47%
high mobility medium traffic	98.66%	1.73%	98.15%	1.42%	97.61%	1.32%

Table 7.2: Comparison of Local and Cooperative Intrusion Detection Programs Evolved by GE for Detection of Ad Hoc Flooding Attack

of a problem by changing the BNF grammar easily. The best programs evolved with each grammar are shown in Table 7.3.

Detection Type	Evolved Program
Local Detection	if((send_rrepPs + exp(frw_aodvPs - updated_routes * pow(frw_rreqPs, added_repairedroutes))) > no_neighbours)
Coop. Detection-1	if((abs(floor(frw_aodvPs) - send_rreqPs) - log(no_neighbours)) > (updated_routes)) //by neighbour nodes
Coop. Detection-2	if(invroutes_timeout < frw_rreqPs && added_neighbours < frw_aodvPs) if((frw_aodvPs - updated_routes) > (max(exp(0.93) + 0.21 - addedroutes_notice, 0.20)) //by neighbour nodes

Table 7.3: The Programs –Best Individuals– Evolved by GE for Detection of Ad Hoc Flooding Attack

7.2.2 Cooperative Detection in Neighbourhood by GP

The local and cooperative intrusion detection programs are also evolved by using GP. The parameters in Table 7.4 are employed. Strongly-typed GP is employed here to return programs in an "if" structure. To evolve cooperative intrusion detection programs, the features of neighbour nodes are also added to the terminal list given in Table 4.2. The performance of the best cooperative detection program evolved is compared with the performance of the best local detection program evolved and demonstrated in Table 7.5.

Objective	Find a computer program to detect ad hoc flooding attack on MANETs cooperatively
Function set	The binary operators +,-,*, /, pow, min, max, mod, percent The unary operators sin, cos, log, ln, sqrt, abs, exp, floor, ceil The comparison operators <, <=, ==, !=, >, >= The relational operators and, or
Terminal set	The feature set
Populations Size	100
Generations	1000
Crossover Probability	0.8
Reproduction Probability	0.2
Tournament Size	7

Table 7.4: The GP Parameter Settings

The results show that cooperative detection algorithm both increases detection rate and decreases false positive rate. The best program achieves a very high detection rate for each network except one (under high mobility and medium traffic). The false positive ratio is also improved a little. The fitness value of local detection is decreased up to 0.6% in some networks with cooperative detection. It is a good improvement based on the fitness values (between 0.37 and 2.38) achieved by GP.

7.2.3 Investigating the Resource Usage of Cooperative Detection Programs

So far, the question "is it possible to increase the effectiveness of an intrusion detection system by collaboration with the IDS agents in its neighbourhood?" has been addressed and it is shown that cooperative intrusion detection with neighbour IDS agents can achieve better fitness values than local detection only does. However suitability of these evolved programs to MANETs should also be considered for this resource-constrained environment. Consequently the energy usage of cooperative intrusion detection programs are now evaluated and compared with the energy usage of local detection programs in this section. The trade-offs between intrusion detection ability and resource consumption of programs (in terms of energy and bandwidth) are also discovered by us-

Network Scenarios	**Local Detection**			**Cooperative Detection**		
	DR	FPR	Fitness	DR	FPR	Fitness
low mobility low traffic	99.47%	0.27%	0.80	99.95%	0.32%	0.37
low mobility medium traffic	99.70%	1.54%	1.84	99.92%	1.39%	1.47
medium mobility low traffic	99.62%	0.23%	0.61	99.72%	0.14%	0.43
medium mobility medium traffic	99.45%	1.56%	2.11	99.61%	1.45%	1.84
high mobility low traffic	99.20%	0.50%	1.29	99.91%	0.61%	0.70
high mobility medium traffic	99.02%	1.41%	2.38	99.10%	1.37%	2.26

Table 7.5: Comparison of Local and Cooperative Intrusion Detection Programs Evolved by GP for Detection of Ad Hoc Flooding Attack

ing multi-objective optimization techniques. Reducing bandwidth usage is investigated by reducing the number of immediate neighbour nodes that cooperate to reach a decision.

In cooperative intrusion detection, each neighbour runs the program and sends his result to the node which aggregates information from neighbour nodes and reaches a decision. In cooperative intrusion detection each node might not need to get/process local information as much as local detection does, since the decision is based on the collected information from each neighbour node. While the information on a node itself might not be able to detect attacks locally, it helps to detect attacks effectively with other neighbours' local information. On the other hand, local detection tries hard to detect attacks with the local information only. These could affect the program's size and energy consumption. The bloat is another affect on program size and energy consumption. In order to analyze energy consumption of evolved programs fairly and discover trade-offs between detection ability and energy consumption of programs SPEA2 is employed.

The SPEA2 algorithm is run to evolve power and bandwidth efficient intrusion detection programs. The algorithm aims to maximize the following objectives simultaneously.

$$f_1 = no.\ of\ attacks\ detected / no.\ of\ attacks \tag{7.1}$$
$$f_2 = 1 - no.\ of\ false\ events / no.\ of\ normal\ events \tag{7.2}$$
$$f_3 = 1 / energy\ consumption \tag{7.3}$$
$$f_4 = 1\ -\ \%\ of\ neighbour\ nodes\ in\ cooperation \tag{7.4}$$

The number of neighbour nodes in cooperation (f_4) is also added to the fitness function in order to minimize communication between IDS agents. Communication between nodes consumes both bandwidth and power. Accordingly we seek to reduce it as far as is practical. A simple way of achieving this is to seek to reduce the number of neighbouring nodes a node collaborates with to reach an IDS decision. In [31] energy consumption in ad hoc networks is modelled for the four states (transmit, receive, sleep, and idle) of the network interface. The cost to a node to send or receive a packet on the network layer is modeled by the linear equation below. The cost associated with channel acquisition (m and b) is assumed to be fixed and given in Table 7.6 [31]. The cost of a sending/receiving packet is proportional to the size of the packet as given in the equation below.

$$Cost = m\ \times\ size\ +\ b \tag{7.5}$$

The total cost of a packet is evaluated by summing the cost of sending packet and the cost of receiving packet (by all receivers). So if it is a point-to-point traffic, the total cost is calculated as follows.

$$Cost = m_{send}\ \times\ size\ +\ b_{send} \tag{7.6}$$
$$+\ m_{recv}\ \times\ size\ +\ b_{recv} \tag{7.7}$$

msend	1.89 mW.s/byte
bsend	246 mW.s
mrecv	0.494 mW.s/byte
brecv	56.1 mW.s

Table 7.6: Fixed Costs in the Power Model

The packet size (the output of evolved programs) in our experiments is quite small, generally it is one or two floating point numbers. However each neighbour node participates in detection by sending their local information. More neighbours means more communication and more energy consumption. So reducing the number of neighbours who send their local information for detection is an objective. Therefore the number of neighbours (%) in cooperation to reach IDS decisions is added to the multi-fitness function (as shown in Equation 7.4.). The trade-offs among the number of neighbours in cooperation, detection ability and power usage of evolved programs are to be discovered.

In mobile networks radio communication has very high power consumption. A node consumes an amount of power(=Rx receiving power) even in its idle state in order to monitor the channel. While transmission power(Tx) is higher, it depends on the transmission range. A node should maintain its transmission power at a level sufficient to reach receiving nodes, or switch to a sleep state to save its power locally. Researchers generally focus on reducing the communication power since this is the main cause of battery depletion. In this research even though an intrusion detection program is likely to consume a small amount of power compared to the communication power, it runs continuously. When the communication is inevitable in mobile networks, controlling the power usage of programs running on a node is another approach to save battery power locally. The degree to which a node engages in "passive" monitoring may also be a factor. With largely passive operation, the relative importance of power consumption of programs increases. These are the main reasons that we consider the power consumed by intrusion detection programs besides its communication power in this research. However, we are aware that further integration of these conpepts is also possible. For example, we could reduce the rate at which neihgbouring information is requested (and hence save power) but this would inevitably give rise to a lower detection capability. (That is to say there is a trade-off to be made here.)

SPEA2 was run thirty times. Each run produced a Pareto optimal set of non-dominated solutions. Figure 7.1 shows the distribution of the union of those thirty sets. (Within the union some individual solutions from one contributing set may dominate or be dominated by an individual solutions from another contributing set). We really want solutions that work well on all three criteria. Consider, for example, the set of solutions with x>=0.8, y>=0.7 and z>=0.5. Solutions in this set clearly perform well on all three axes. In some respects we may consider them "excellent", or an acceptable outcome from a run of our technique. The black points in the figure show the acceptable solutions found by our technique. The average number of "acceptable" solutions in each run is 2.8333 (with a standard deviation of 2.0356). The number of runs which produced no "acceptable" solution is 6. Of course, different definitions of acceptable are clearly possible.

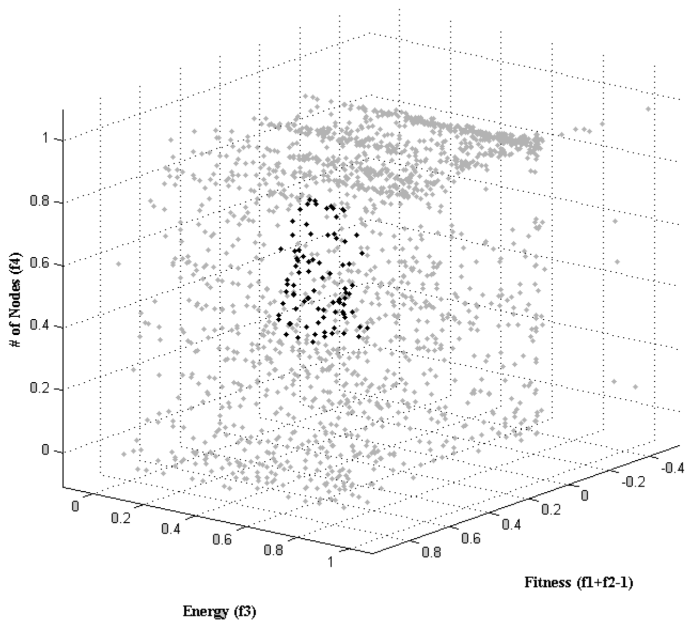


Figure 7.1: Union of the Non-dominated Solutions from Each Run

MOEC returns a set of solutions and one solution has four values showing different trade-offs between the multi-objectives described in Equation 7.1-7.4. Since analyzing the multi-dimensonal data is difficult, we show the relation between each pair of multi-objectives separately here. Only programs with high classification accuracy (where the fitness value of detection rate and (100-false positive rate) is bigger than 98%) are selected here and evaluated on a network under medium mobility and medium traffic. Firsly, the relation between the fitness values of these programs and the percentage of neighbour nodes asked for information to achieve these fitness values is shown in Figure 7.2. In the figure when the number of neighbours participating in the detection increases, the effectiveness of the programs increases as well. The correlation between these values is analyzed for each run and the average correlation coefficient is calculated as 0.3829$\mp$0.0963. This value (with the p-value 0.0142$\mp$0.0226) shows that this two value is correlated, however it is not a very strong relationship. There are other influences and the relationship between these values is not fixed. Since the aim is to find different trade-offs among four objectives, the value of one objective is affected by other objectives as well.

The relation between energy usage of programs and the percentage of neighbour nodes in cooperation is shown in Figure 7.3. There is no statistifically significant correlation

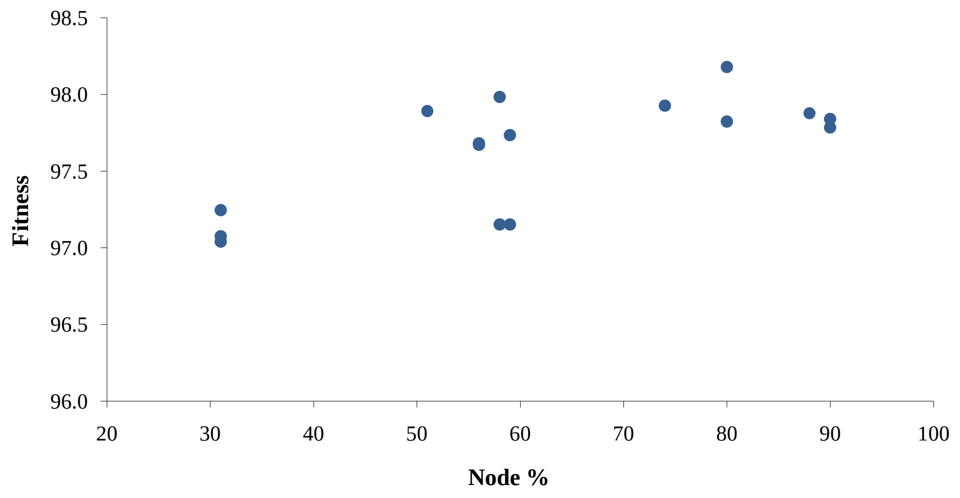


Figure 7.2: Fitness vs. Percentage of Neighbour Nodes in Cooperation

found between these two elements. Energy usage is affected by the fitness values as shown in Figure 7.4. Programs with higher classification accuracy tend to consume more energy as shown. We here present few solutions produced in one run. These values are weakly correlated in some runs where the average correlation coefficient is 0.3070∓0.0131 (with the p-value 0.0167∓0.0038). It means while the energy increases, the fitness increases too in the output non-dominated solutions. However the fitness is also affected by other objectives. Moreover it is observed that the energy consumption of these programs is lower than that of the programs given in the previous sections which only considers classification accuracy as the fitness function.

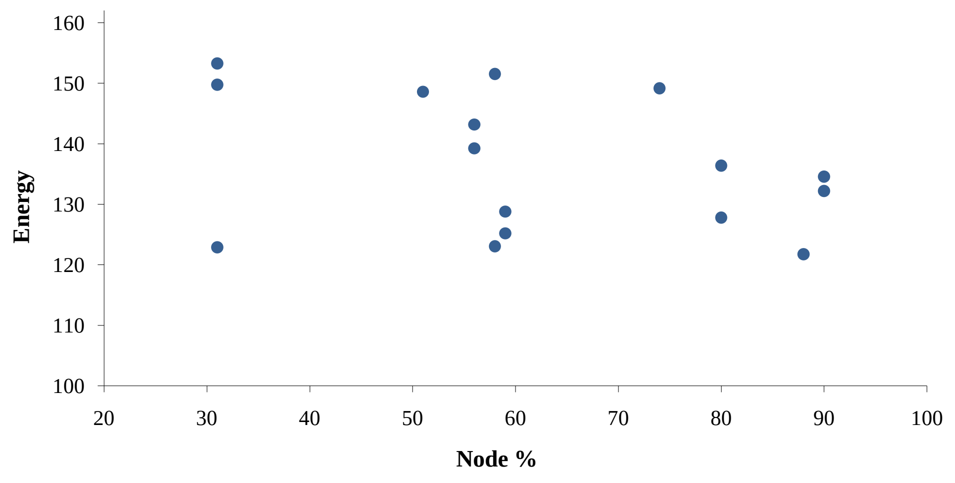


Figure 7.3: Energy Consumption vs. Percentage of Neighbour Nodes in Cooperation

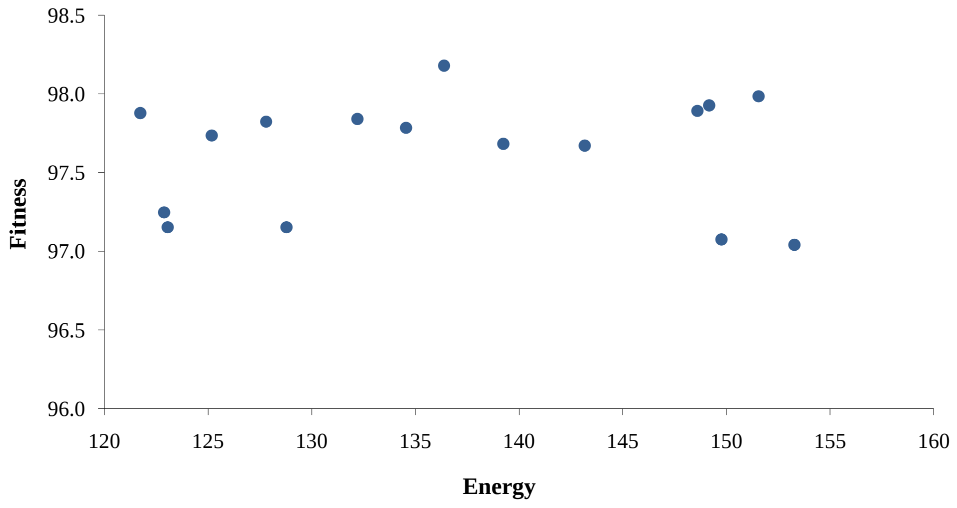


Figure 7.4: Fitness vs. Energy Consumption

The performance of some of these programs evolved is demonstrated in Table 7.7. The best classification accuracy is achieved by program 4 which uses 80% of neighbour nodes to reach a decision. Where program 1 consumes higher energy than the program 4, but decreases the number of nodes participating in intrusion detection very significantly. Its detection ability (fitness value) is as almost the same as that of local detection, but it only uses half of its neighbour nodes. It decreases the false positive rate with a small decrease in the detection rate. The energy consumption of evolved programs is generally almost the same as for the power-aware local detection programs' evolved in Chapter 6. Different trade-offs among the classification accuracy, the energy consumption and the number of neighbour nodes in cooperation are clearly seen in these results. Here we are using the number of collaborating nodes as a proxy for resource consumption incurred by collaboration. Thus, we need to send requests to our neighbours and they must respond. This incurs both broadcast and reception costs but also information retrieval costs within each of the neighbouring nodes. It is clear that we should seek to reduce the number of collaborating neighbours as much as it is practical.

The performance of evolved programs is demonstrated on networks only under medium mobility and medium traffic as shown in Table 7.7. On simulated networks under high mobility the detection rate is decreased down to 97% with a decrease in the false positive rate down to 1%.

The programs evolved are demonstrated in Table 7.8.

This chapter demonstrates the potential use of evolutionary computation techniques

Program No.	**Detection Rate**	**False Positive Rate**	**Nodes**	**Energy**
1	99.21%	1.32%	51%	$\simeq 149$
2	99.53%	1.54%	58%	$\simeq 151$
3	99.65%	1.72%	74%	$\simeq 149$
4	99.41%	1.23%	80%	$\simeq 137$

Table 7.7: The Performance of Some Cooperative Programs Evolved by MOEC

Program No.	**Evolved Program**
1	(cos(log(cos(log(log(frw_rreqPs))) - frw_rreqPs))) < (log(log(log(frw_rreqPs) - frw_rreqPs)))
2	(cos(log(repaired_routes + (log(frw_rreqPs) / frw_rreqPs)))) < (log(log(frw_rreqPs)))
3	(frw_rreqPs) > (min(exp(exp(exp(0.00627320552581323))), exp(exp(log10(exp(min(exp(0.00627320552581323),active_routes)))))))
4	(cos(log(frw_rrepPs - frw_rreqPs))) < (log(log(log(addedroutes_notice) - (log(frw_rrepPs - 2frw_rreqPs)) - frw_rreqPs))))

Table 7.8: Example Programs Evolved by MOEC for Detection of Ad Hoc Flooding Attack Cooperatively

to discover complex properties of MANETs (such as limited power and limited bandwidth) and to generate a suitable intrusion detection approach applicable to this new environment. It is shown that cooperative intrusion detection with neighbour nodes increase the effectiveness of the system. The energy consumption of these programs is almost the same as that of the power-aware local detection programs evolved in Chapter 6. However cooperative intrusion detection requires communication between nodes. Even though the size of the packets sent for intrusion detection is small, message sending and receiving still consume an amount of energy. For that reason the number of neigbour nodes taking a role in intrusion detection is aimed to be minimized by using MOEC techniques. The results show that the performance of the local detection programs can be achieved by using a distributed and cooperative detection program. Some neighbouring nodes (50% of neighbouring nodes asked for information) participate in cooperative intrusion detection. If the number of nodes in cooperation increases, the classification accuracy increases as well. Different trade-offs that can be applied according to the application are presented here. We believe this is the first work to consider constrained resources in the case of IDS for MANETs.

Chapter 8

Conclusion

This chapter completes the thesis by summarizing the research and reviewing the contributions. The thesis hypothesises presented in Chapter 1 are also revisited to show how the work done in this research support them. Finally areas of future work are discussed.

8.1 Summary of Experimentation

The main problems of existing approaches for intrusion detection in MANETs have been addressed in Chapter 3. The main issues can be be summarized as follows:

- The traditional way of monitoring network traffic at the traffic concentration points is no longer suitable for MANETs. Network data on MANETs is distributed to all nodes in the network.
- Detection of malicious activies is a difficult research problem even on stable networks (with fixed infrastructures). The dynamic topology of MANETs makes intrusion detection harder.
- MANET nodes usually have limited resources and bandwidth, and existing solutions might not be suitable for this environment.

Researchers have generally focused on the first two issues so far. However consideration resource-constraints is vital. In this thesis the limited resources of nodes are also taken into consideration.

This research investigates the use of evolutionary computation techniques, specifically GP and GE, to evolve intrusion detection programs for MANETs. Since MANETs have become the target of new attacks which exploit the cooperative nature of routing protocols, we have aimed to detect specific attacks targetting routing protocols. It is the first application of evolutionary computation to intrusion detection in MANETs. Furthermore it is one of the few misuse-based approaches proposed in the literature which can be used to complement other approaches.

This thesis consists of the detailed descriptions of how to apply evolutionary computation techniques to the detection of the following attacks: ad hoc flooding, route disruption (and its variations), and dropping attacks. As given in *hypothesis 1: evolutionary computation will be able to discover complex properties of mobile ad hoc networks and evolve intrusion detection programs suitable for this new environment.* To evaluate this hypothesis we performed a variety of experiments.

The performance of programs evolved using GP and GE is evaluated on simulated networks with varying mobility and traffic patterns. Both techniques show a good performance for detecting ad hoc flooding and route disruption attacks. The programs provide close to perfect detection with false positive rate less than 2%. The factors (*e.g.* mobility, traffic, topology) affecting the false positive rate are also discussed. However programs evolved to detect dropping attacks cause a high rate of false positives which cannot be managed easily. The performance of these programs could be increased with the use of data from lower layers in the protocol stack since lots of packet losses could occur (before packets are sent to the routing layer) due to congestion, and wireless link transmission errors other than those due to mobility. The approximate optimal parameters for GP and GE were also explored and the algorithms compared fairly at these parameter settings.

The performance of hand-coded programs using the same features given as input to GP and GE algorithms is also presented and compared with the programs evolved automatically in this research. The results show that manual detection achieves almost a perfect detection rate on ad hoc flooding attacks. However it does not perform well on differentiating benign flooding from malicious flooding. On the other hand GP and GE derived programs reduce the false positive rate by almost as half as the manual detection does with a small amount of decrease in the detection rate. For route disruption attack GE slightly decreases the false positive rate. It is shown that this attack could be detected by small and relatively simple programs, since it violates the routing protocol specifications directly. Overall, GP and GE considerably outperform the manual detection for the ad hoc flooding attack and route disruption attacks.

Any proposed approaches (no matter how effective they are) should be suitable for MANETs. Since battery power is the critical resource in mobile nodes, energy consumption of evolved programs using GP are analyzed in Chapter 6. While the classification accuracy of a program is high, energy consumption of the program gets higher as well for ad hoc flooding attacks. It was also shown that route disruption attacks could be detected by small programs which consume lower energy. Furthermore, the effect of program size on energy consumption of evolved programs is analyzed. These results prompt us to find trade-offs between classification accuracy and energy consumption of evolved programs.

Multi-objective evolutionary computation is employed to discover the relations among detection rate, false positive rate and energy consumption of evolved programs. A set of solutions which show different trade-offs among these objectives is obtained for each attack. The results show that an increase in false positive rate is correlated with a decrease in energy consumption of detection programs for ad hoc flooding attacks. Since route disruption is a simpler attack, programs closer to the optimum solution which have high detection ability with low energy consumption are produced by MOEC techniques. For both attacks programs with lower energy consumption than programs evolved with the single objective (classification accuracy) stand out in the results. Finally, programs are evolved to detect two attacks together by using MOEC techniques to answer the following research question: "Is it better to evolve one program to detect both attacks or evolve two programs each using half the resource usage?". In the results there are programs which are more energy-efficient than two programs which detect these attacks separately, however they do not show as high classification accuracy as the two programs do separately. It is a trade-off to be considered by the designer.

Finally a suitable intrusion detection architecture for MANETs is sought in Chapter 7. The work explores how evolutionary computation techniques could evolve to improve its detection performance and its suitability to MANET environment. An architecture (*coopeative detection in neighbourhood*) is investigated to address the following research question: "Is it possible to increase the effectiveness of an intrusion detection system by collaboration with the IDS agents in its neighbourhood?". Distributed and cooperative detection programs are evolved using GP and GE techniques to detect ad hoc flooding attacks. In GE the BNF grammar is extended to allow the use of data from neighbour nodes. Evolved distributed and cooperative programs are shown to generate lower false positives than the local detection does. With the flexibility provided by the grammar in GE the communication among nodes is also reduced enormously. Distributed and cooperative programs evolved using GP also show a slightly better performance (high detection rate, low false positive rate) than the local detection programs do. We also explore the efficiency of this architecture. The trade-offs between the intrusion detection ability, the enery usage of programs, and the number of neighbours in cooperation are discovered using multi-objective optimization techniques and interesting results have been obtained. The results show that the performance of the local detection programs can be achieved by using half of neighbouring nodes of a node which cooperates with each other to make decisions. If the number of nodes in cooperation increases, the classification accuracy increases as well. Different trade-offs which can be applied according to the application are presented here. *Hypothesis 2: multi objective evolutionary computation will allow us to discover trade-offs between functional (intrusion detection ability) and non-functional (power and bandwidth usage) properties of intrusion detection programs* is supported with these results.

8.2 Thesis Contributions

The main contributions of this research are outlined as follows:

Evolutionary computation techniques for intrusion detection in MANETs: This research investigates the use of a promising technique from artificial intelligence to synthesise the most appropriate intrusion detection programs for this challenging network type. Evolutionary computation techniques essentially "grow" intrusion detection programs by evaluating populations of potential programs and subjecting them to a variety of genetically inspired operators. This thesis shows that GP and GE can be used to evolve efficient detectors for known attacks against routing protocol on MANETs. To the best of our knowledge it is the first application of evolutionary computation techniques to intrusion detection in MANETs.

A misuse-based approach: This research presents evolved programs using GP and GE for the detection of some specific attacks against routing protocols: ad hoc flooding and route disruption attacks. It aims to evolve the characteristics of these attacks automatically by differentiating malicious behaviour from normal behaviour of the network or the node. It is one of the few misuse-based intrusion detection approaches proposed for MANETs, while they are widely used for conventional networks due to their low false positive ratios.

This approach shows similarity with anomaly-based intrusion detection systems by aiming to differentiate malicious behaviour from normal behaviour. In these systems the normal behaviours of the system, the user, etc. are defined in a general way and, any deviations from the defined normal profile is considered malicious. The idea that malicious behaviour has different characteristics from normal behaviour is the core of intrusion detection systems. In misuse-based systems this idea is applied by defining signatures which show different characteristics of known attacks or system vulnerabilities than normal behavior. The latter approach is employed in this research.

Efficiency: This research proposes a novel approach by discovering different trade-offs between functional and non-functional properties of programs. Our main contribution in this thesis is to evolve a set of programs for each attack offering different trade-offs between intrusion detection ability and energy usage. Moreover, we investigate if it is better to evolve separate programs for each attack or one program to detect both attacks. Our techniques can be used to generate solution sets with the best (or near best) trade-offs possible. A final choice between solutions making different tradeoffs rests with the designer.

Intrusion Detection Architecture: A suitable intrusion detection architecture is investigated in this thesis. Even though the same architecture has been investigated before in other approaches, it is a novel approach in terms of proposing how to choose monitoring nodes in MANETs by considering limited resources. Beside energy consumption of programs, energy consumption by message sending and receiving in the cooperative detection is taken into account. The interaction between IDS agents and the number of nodes participating in detection are reduced by using multi-objective optimization techniques.

To conclude in this thesis we demonstrate the potential use of evolutionary computation techniques to discover complex properties of MANETs and to propose a suitable intrusion detection on this new environment respectively. Target audiences are artificial intelligence and mobile ad hoc networks community. The properties of MANETs taken into consideration in this research could be summarized as mobility, lack of concentration points, limited power and limited bandwidth. The mobility issue is considered in the design stage by adding features reflecting mobility levels of the network (directly or indirectly) to the model. To overcome lack of concentration points which are used to monitor all network traffic in conventional networks, a distributed and coopeative architecture is proposed to get information beyond local data. Limited resource issues are taken into account by applying a multi-objective optimization technique which aims to optimize power and bandwidth usage of IDS agents as well as their detection ability.

The work presented in this thesis crosses sub-disciplines of computer science. We have endeavoured to present our work to different communities. Thus, publications have appeared in fora of interest to the ad hoc networks and wireless networks community as well as fora familiar to the AI/evolutionary computation community.

8.3 Future Research

The potential areas for future research are summarized below:

Applying evolutionary computation techniques to other areas: In this researh, we show how to apply evolutionary computation techniques to the problem of intrusion detection in MANETs and how to explore different trade-offs between criteria in such resource-limited networks. The work proposed in this research could be adapted easily to other areas such as wireless sensor networks. It is likely that for wireless sensor networks the ability to make good trade-offs will be particularly important since these networks are more resource-constrained than MANETs.

Exploration of new attacks: More research is needed to analyze and discover MANET attacks. Programs could be evolved for the detection of new attacks by using the techniques proposed in this thesis. Similar attacks which have similar effects on a network could be categorized (*e.g.* DoS attacks) and programs to detect these attacks together could be evolved. It is believed that these attacks could have common signatures and evolving programs to detect these attacks together could be more efficient than detecting them separately. Futhermore distributed attacks which cannot be detected locally and have a distributed effect on the network (*e.g.* network scans) could be explored by using cooperative intrusion detection approach proposed in this research. Our approach could evolve effective programs to detect attacks which seems normal to a node but malicious if detected cooperatively.

Improving our approach: In this research we have mainly focused on the data collection and detection components of an IDS. We have dealt only with the raising of alarms in the response component. However there is a tendency in the research of intrusion detection to prevent future attacks by active responses. In MANETs few researchers have employed reputation systems. Those systems maintain a list of misbehaving nodes and exclude these nodes from network operations. Our approach could be improved by developing an aggregation mechanism for alarms produced by each node and adding an active response module. Moreover it could be integrated with other intrusion detection techniques in order to increase effectiveness of the overall system.

More research could be done to investigate a suitable intrusion detection architecture for MANETs. For example, more sophisticated monitoring node assignment algorithms could be developed that take into account other factors such as nodes' coverage areas, the amount of traffic passing on them, and the like. In addition, it is not yet established how many nodes in a MANET need also participate in IDS. Thus, we could explore whether a small number of nodes with high computational ability is better than many nodes with lower computational budgets.

To conclude we believe that artificial intelligence based approaches to program synthesis such as grammatical evolution, genetic programming, and multi-objective evolutionary computation, are of significant potential benefit for the evolution of IDS programs for challenging complex environments such as MANETs and we encourage the research community to explore their use.

References

[1] BonnMotion: A mobility scenario generation and analysis tool. http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewaal/BonnMotion/. 99, 110

[2] Ecj18: A Java-based evolutionary computation research system. http://cs.gmu.edu/ eclab/projects/ecj/. 95, 108, 119

[3] Intrusion detection message exchange format (IDMEF). http://www.ietf.org/html.charters/OLD/idwg-charter.html. 42, 52

[4] KDD cup 1999 intrusion detection data set. http://kdd.ics.uci.edu/databases/kddcup99/. 89, 90

[5] libGE: A C++ library for grammatical evolution. http://bds.ul.ie/libGE. 96, 108, 111

[6] Mobile agent. http://en.wikipedia.org/wiki/Mobile_agent. 46

[7] ns-2: The network simulator. http://www.isi.edu/nsnam/ns. 99, 110

[8] Quadratic programming. http://en.wikipedia.org/wiki/Quadratic_programming. 112

[9] SimpleScalar. http://www.simplescalar.com/. 117, 118

[10] Snort. http://www.snort.org/. 59

[11] A. Abraham and C. Grosan. Evolving intrusion detection systems. In *Genetic Systems Programming: Theory and Experiences*, volume 13, pages 57–79. Springer, 2006. 89, 91

[12] A. Abraham, C. Grosan, and C. Martiv-Vide. Evolutionary design of intrusion detection programs. *International Journal of Network Security*, 4:328–339, 2007. 89

[13] T. Anantvalee and J. Wu. A survey on intrusion detection in mobile ad hoc networks. In *Wireless Network Security*, pages 159–180. Springer, 2007. 45, 46, 73

[14] F. Anjum, D. Suhbadrabandhu, and S. Sarkar. Signature based intrusion detection for wireless ad hoc networks: a comparative study of various routing protocols. In *Proceedings of the 58th IEEE Vehicular Technology Conference*. 60

[15] F. Anjum and R. Talpade. Lipad: Lightweight packet drop detection for ad hoc networks. In *Proceedings of the 60th IEEE Vehicular Technology Conference*, pages 1233–1237. IEEE, 2004. 8, 67, 68

[16] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens. An on demand secure routing protocol resilient to byzantine failures. In *Proceedings of the ACM Workshop on Wireless Security*, 2002. 37

[17] S. Axelsson. Intrusion detection systems: A survey and taxonomy. Technical Report 99–15, Department of Computer Engineering, Chalmers University of Technology, 2000. 41, 43

[18] W. Banzhaf. Genotype-phenotype mapping and neutral variation –a case study in genetic programming. In *Proceedings of the International Conference on Evolutionary Computation: The Third Conference on Parallel Problem Solving from Nature, LNCS 866*, pages 322–332. Springer, 1994. 88

[19] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francome. *Genetic Programming: An Introduction on the Automatic Evolution of Computer Programs and Its Applications*. Morgan Kaufman Publishers, 1998. 84

[20] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th International Symposiyum on Computer Architecture (ISCA-27)*, 2000. 117, 118

[21] S. Buchegger and J. Le Boudec. Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks. In *Proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pages 403–410. IEEE Computer Society, January 2002. 7, 66, 67

[22] S. Buchegger, C. Tissieres, and J.-Y. Le Boudec. A test-bed for misbehaviour detection in mobile ad-hoc networks –how much can watchdogs really do? In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, 2003. 34

[23] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2:483–502, 2002. 99

[24] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41, 2009. 40

[25] J.H. Chang and L. Tassiulas. Energy conserving routing in wireless ad-hoc networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 22–31, 2000. 116

[26] H. Chivers and J.A. Clark. Smart dust, friend or foe? –replacing identity with configuration trust. *Computer Networks*, 46(5):723–740, 2004. 32

[27] C. A. C. Coello, G. B. Lamont, and D. A. V. LinkVeldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer US, 2002. 119

[28] M. Crosbie and G. Stafford. Applying genetic programming to intrusion detection. In *Proceedings of the AAAI Symposium on Genetic Programming*, 1995. 89

[29] D. Denning. An intrusion detection model. *IEEE Transactions on Software Engineering*, 13(2):222–232, 1987. 22, 39

[30] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003. 81, 82, 83

[31] L. M. Feeney. An energy consumption model for performance analysis of routing protocols for mobile ad hoc networks. *Mobile Networks and Applications*, 6:239–249, 2001. 137

[32] A. Fourati and K.A. Agha. An IDS first line of defense for ad hoc networks. In *Proceedings of the Wireless Communications and Networking Conference*, pages 2619–2624, 2007. 59

[33] S. Gavini. Detecting packet-dropping faults in mobile ad-hoc networks. Master's thesis, School of Electrical Engineering and Computer Science, Washington State University, 2004. 69

[34] R. Guha, O. Kachirski, D.G. Schwartz, S. Stoecklin, and E. Yilmaz. Case-based agents for packet-level intrusion detection in ad hoc networks. In *Proceedings of the 17th International Symposium on Computer and Information Sciences*, 2002. 7, 59

[35] Z.J. Haas, J. Deng, B. Liang, P. Papadimitratos, and S. Sajama. Wireless ad hoc network. In *Encyclopedia of Telecommunications*. Wiley-Interscience, 2002. 28

[36] E. Hansson, J. Gronkvist, K. Persson, and D. Nardquist. Specification-based intrusion detection combined with cryptography methods for mobile ad hoc networks. Technical report, FOI Swedish Defence Research Agency/Command and Control Systems, 2005. 57

[37] H.M. Hassan, M. Mahmoud, and S. El-Kassas. Securing the AODV protocol using specification-based intrusion detection. In *Proceedings of the 2nd ACM International Workshop on Quality of Service and Security for Wireless and Mobile Networks*, pages 33–35, 2006. 57

[38] R. Heady, G. Luger, A. Maccabe, and M. Servilla. The architecture of a network level intrusion detection system. Technical report, Computer Science Department, New Mexico, 1990. 39

[39] C.-Y. Hu, A. Perrig, and D.B. Johnson. Packet leashes: A defence against wormhole attacks in wireless ad hoc networks. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications (INFOCOM)*, 2003. 37

[40] Y.-C. Hu, A. Perrig, and D.B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proceedings of the 8th International Conference on Mobile Computing and Networks*, pages 12–23, 2002. 37

[41] Y.-C. Hu, A. Perrig, and D.B. Johnson. Rushing attacks and defence in wireless ad hoc network routing protocols. In *Proceedings of the ACM Workshop on Wireless Security*, 2003. 36

[42] Y. Huang, Wei Fan, Wenke Lee, and Philip S. Yu. Cross-feature analysis for detection ad-hoc routing anomalies. In *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS)*. 7, 44, 49, 50, 52, 117

[43] Y. Huang and W. Lee. A cooperative intrusion detection system for ad hoc networks. In *Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2003. 7, 49, 50, 52, 75

[44] Y. Huang and Wenke Lee. Attack analysis and detection for ad hoc routing protocols. In *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID'04)*, pages 125–145. Springer, 2004. 7, 41, 61

[45] J.-P. Hubaux, L. Buttyan, and S. Capkun. The quest for security in mobile ad hoc networks. In *Proceedings of the 2nd ACM International Symposium on Mobile Ad hoc Networking and Computing*, pages 146–155, 2001. 29, 33

[46] P. Jacquet, P. MÃijhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. In *Proceedings of the IEEE International In Multi Topic Conference (INMIC)*, pages 62–68, 2001. 28, 58

[47] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996. 28

[48] D. M. Johnson, A. M. Teredesai, and R. T. Saltarelli. Genetic programming in wireless sensor networks. In *Proceedings of the European Conference on Genetic Programming (EUROGP 2005), LNCS 3447*, pages 96–107. Springer, 2005. 90

[49] O. Kachirski and R. Guha. Effective intrusion detection usign multiple sensors in wireless ad hoc networks. In *Proceedings of the 36th IEEE International Conference on System Sciences*, 2003. 7, 46, 47, 49, 54, 55, 117

[50] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Elsevier's Ad Hoc Networks Journal: Special Issue on Sensor Network Applications and Protocols*, 1(2–3):293–315, 2003. 36

[51] A. Karygiannis, E. Antonakakis, and A. Apostolopoulos. Detecting critical nodes for MANET intrusion detection systems. In *Proceedings of the 2nd International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing*, 2006. 75

[52] P. Kazienko and P. Dorosz. Intrusion detection systems (IDS), 2004. http://www.windowsecurity.com/articles/IDS-Part2-Classification-methods-techniques.html. 40

[53] H. Kim, D. Kim, and S. Kim. Lifetime-enhancing selection of monitoring nodes for intrusion detection in mobile ad hoc networks. *International Journal of Electronics and Communications*, 60:248–250, 2006. 116, 117

[54] J. Kong, X. Hong, and M. Gerla. A new set of passive routing attacks in mobile ad hoc networks. In *Proceedings of the IEEE Military Communications Conference (MILCOM)*, 2003. 33

[55] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992. 84

[56] S. Laniepce, J. Demerjian, and A. Mokhtari. Cooperation monitoring issues in ad hoc networks. In *Proceedings of the International Conference on Communications and Mobile Computing*, pages 695–700, 2006. 47

[57] W. Li. Using genetic algorithm for network intrusion detection. In *Proceedings of the United States Department of Energy Cyber Security Training Conference*, 2004. 89

[58] Y. Li and J. Wei. Guidelines on selecting intrusion detection methods in MANET. In *Proceedings of the Information Systems Educators Conference*, 2004. 21, 29

[59] Y. Lu, Y. Zhong, and B. Bhargava. Packet loss in mobile ad hoc networks. Technical Report 03–009, Department of Computer Science, Purdue University, April. 79, 80

[60] E. Lundin and E. Jonsson. Survey of intrusion detection research. Technical Report 02–04, Department of Computer Engineering, Chalmers University of Technology, 2002. 40, 42

[61] M. Maleki, K. Dantu, and M. Pedram. Power-aware source routing protocol for mobile ad hoc networks. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 72–75, 2002. 116

[62] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, pages 255–265, 2000. 7, 63, 65, 66, 68, 108

[63] L. Me. GASSATA, A genetic algorithm as an alternative tool for security audit trails analysis. In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID'98)*, 1998. 89

[64] S. Mehfuz and M. N. Doja. Swarm intelligent power-aware detection of unauthorized and compromised nodes in MANETs. *Journal of Artificial Evolution and Applications*, 2008. 116

[65] R. Molva and P. Michiardi. Security in ad hoc networks. In *Proceedings of the Personal Wireless Communications (PWC'03), LNCS 2775*, pages 756–775. Springer, 2003. 28

[66] D. J. Montana. Strongly typed genetic programming. *Evolutionary Computation*, 3:199–230, 1995. 94

[67] 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 IEEE Workshop on Information Assurance*, pages 60–67, 2003. 32, 35

[68] P. Ning and K. Sun. How to misuse AODV: A case study of insider attacks against mobile ad hoc routing protocols. Technical report, Department of Computer Science, North Caroline State University, 2003. 80

[69] M. O'Neill and C. Ryan. Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5:4:349–358, 2001. 86

[70] M. O'Neill and C. Ryan. *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language.* Springer, 2003. 87, 88, 109, 111

[71] J.-M. Orset, B. Alcalde, and A. Cavalli. An efsm-based intrusion detection system for ad hoc networks. In *Proceedings of the Automated Technology for Verification and Analysis*, pages 400–413, 2005. 59

[72] J. Parker, J. Undercoffer, J. Pinkston, and A. Joshi. On intrusion detection and response for mobile ad hoc networks. In *Proceedings of the 23th IEEE International Performance Computing and Communications Conference*, 2004. 8, 68

[73] C. Perkins, E. Belding-Royer, and S. Dan. Ad hoc on-demand distance vector (AODV) routing, 2003. http://www.ietf.org/rfc/rfc3651.txt. 28, 78

[74] N.J. Puketza, K. Zhang, M. Chung, B. Mukherjee, and R.A. Olsson. A methodology for testing intrusion detection systems. *IEEE Transactions on Software Engineering*, 22:719–729, 1996. 42

[75] R.S. Puttini, J.-M. Percher, L. Me, O. Camp, R. Jr. Sousa, C.J.B. Abbas, and L.J. Garcia-Villalba. A modular architecture for distributed IDS in MANET. In *Proceedings of the Computational Science and Its Applications: LNCS 2669*, pages 91–113, 2003. 7, 40, 62

[76] R. Roman, C. Alcaraz, and J. Lopez. A survey of cryptographic primitives and implementations for hardware-constrained sensor network nodes. *Mobile Networks and Applications*, 12:231–244, 2007. 13, 115, 116

[77] Lippmann R.P., Fried D.J., Graf I., Haines J.W., Kendall K.R., McClung D., Weber D., Webster S.E., Wyschogrod D., Cunningham R.K., and Zissman M.A. Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX)*, volume 2, pages 12–26, 2000. 42

[78] Lippmann R.P., Haines J.W., Fried D.J., Korba J., and Das K. Analysis and results of the 1999 darpa off-line intrusion detection evaluation. In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID'00), LNCS 2212*, pages 162–182. Springer, 2000. 43

[79] C. Ryan, J.J. Colline, and M. O'Neill. Grammatical evolution: Evolving programs for an arbitrary language. In *Proceedings of the 1st European Workshop on Genetic Programming, LNCS 1391*, pages 83–95. Springer, 1998. 86, 123

[80] K. Sanzgiri, B. Dahill, B.N. Levine, C. Shields, and E.M. Belding-Royer. A secure routing protocol for ad hoc networks. In *Proceedings of the 10th IEEE Conference on Network Protocols*, pages 12–23, 2002. 37

[81] S. Singh, M. Woo, and C. S. Raghavendra. Power aware routing in mobile ad hoc networks. In *Proceedings of the Annual International Conference on Mobile Computing and Networking*, pages 181–190, 1998. 116

[82] S.N. Sivanandam and S.N. Deepa. *Introduction to Genetic Algorithms.* Springer, 2008. 111

[83] A.B. Smith. An examination of an intrusion detection architecture for wireless ad hoc networks. In *Proceedings of the 5th National Colloquium for Information System Security Education*, 2001. 7, 44, 60, 117

[84] D. Song, M. I. Heywood, and A. Nur Zincir-Heywood. Training genetic programming on half a million patterns: An example from anomaly detection. *IEEE Transactions on Evolutionary Computation*, 9(3), June 2005. 90, 91

[85] T. Srinivasan, V. Mahadevan, A. Meyyappan, A. Manikandan, M. Nivedita, and N. Pavithra. Hybrid agents for power-aware intrusion detection in highly mobile ad hoc networks. In *Proceedings of the International Conference on Systems and Network Communication.* IEEE Computer Society, 2006. 117

[86] F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Proceedings of the International Workshop on Security Protocols.* Springer, 1999. 30, 36

[87] D. Sterne, P. Balasubramanyam, D. Carman, B. Wilson, R. Talpade, C. Ko, R. Balupari, C.-Y. Tseng, and T. Bowen. A general cooperative intrusion detection architecture for MANETs. In *Proceedings of the 3rd International Workshop on Information Assurance*, pages 57–70, 2005. 7, 53, 117

[88] B. Sun. *Intrusion Detection in Mobile Ad Hoc Networks.* PhD thesis, Computer Science, Texas A&M University, 2004. 52, 93

[89] B. Sun, K. Wu, and U.W. Pooch. Zone-based intrusion detection for mobile ad hoc networks. *International Journal of Ad Hoc and Sensor Wireless Networks*, 2(3), 2003. 7, 32, 51, 52, 53, 75, 81

[90] C.-Y. Tseng, P. Balasubramayan, C. Ko, R. Limprasittiporn, J. Rowe, and K. Lewitt. A specification-based intrusion detection system for AODV. In *Proceedings of the ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, 2003. 7, 41, 56, 107

[91] C.H. Tseng, T. Song, P. Balasubramanyam, C. Ko, and K. Levitt. A specification-based intrusion detection model for OLSR. In *Proceedings of the 8th International Symposium on Recent Advances in Intrusion Detection (RAID'05), LNCS 3858*, pages 330–350. Springer, 2005. 58

[92] C.H. Tseng, S.-H. Wang, Wenke Lee, C. Ko, and K. Lewitt. Demem: Distributed evidence driven message exchange intrusion detection model for MANET. In *Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID'06)*, pages 249–271. Springer, 2006. 7, 44, 57, 59, 75

[93] P. Uppuluri and R. Sekar. Experiences with specification-based intrusion detection. In *Proceedings of the Recent Advances in Intrusion Detection (RAID'01), LNCS 2212*, pages 172–189. Springer, 2001. 41

[94] G. Vigna, S. Gwalani, K. Srinivasan, E. M. Belding-Royer, and R. A. Kemmerer. An intrusion detection tool for AODV-based ad hoc wireless networks. In *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04)*, pages 16–27, Washington, DC, USA, 2004. IEEE Computer Society. 107, 108

[95] M. Wang, L. Lamont, P. Mason, and M. Gorlatova. An effective intrusion detection approach for OLSR MANET protocol. In *Proceedings of the 1st IEEE ICNP Workshop on Secure Network Protocols*, pages 55–60, 2005. 59

[96] X. Wang, T. Lin, and J. Wong. Feature selection in intrusion detection system over mobile ad-hoc network. Technical report, Department of Computer Science, Iowa State University, 2005. 73, 75

[97] T. Weise. Genetic programming for sensor networks. Technical report, Distributed Systems Group, Fachbereich 16: Elektrotechnik/Informatik, University of Kassel, 2006. 90

[98] D. R. White, J. Clark, J. Jacob, and S. Poulding. Evolving software in the presence of resource constraints. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'08)*. Springer, 2008. 123

[99] D. R. White and S. Poulding. A rigorous evaluation of crossover and mutation in genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference (EuroGP'08), LNCS 5481*, pages 220–231. Springer, 2009. 109, 111

[100] D. Wilson and D. Kaur. Knowledge extraction from KDD'99 intrusion data using grammatical evolution. *WSEAS Transactions on Information Science and Applications*, 4:237–244, February 2007. 90

[101] B. Wu, J. Chen, J. Wu, and M. Cardei. *A Survey on Attacks and Countermeasures in Mobile Ad Hoc Networks*, chapter 12. 2006. 36

[102] H. Yang, H. Luo, F. Ye, S. Lu, and L. Zhang. Security in mobile ad hoc networks: Challenges and solutions. *IEEE Wireless Communications*, 11(1):38–47, 2004. 30

[103] P.-W. Yau and C.J. Mitchell. Security vulnerabilities in ad hoc networks. In *Proceedings of the 7th International Symposium on Communications Theory and Applications*, pages 99–104, 2003. 31, 34

[104] 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):83–94, 2005. 36

[105] P. Yi, Y. Zhong, and S. Zhang. A novel intrusion detection method for mobile ad hoc networks. In *Proceedings of the Advances in Grid Computing (EGC'05), LNCS 3470*. 57, 117

[106] Y. Zhang and W. Lee. Intrusion detection in wireless ad hoc networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom'00)*, pages 275–283, 2000. 7, 30, 43, 48, 49

[107] Y. Zhang, W. Lee, and Y. Huang. Intrusion detection techniques for mobile wireless networks. *Wireless Networks Journal (ACM WINET)*, 2(5), September 2003. 7, 48

[108] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. Technical Report 103, Swiss Federal Institute of Technology, 2001. 119, 120