SPIM

Thèse de Doctorat

école doctorale **sciences pour l'ingénieur et microtechniques**

UFC UNIVERSITÉ DE FRANCHE-COMTÉ

Security supervision of mobile ad hoc networks

a lightweight, robust and reliable intrusion detection system

Mouhannad ALATTAR

SPIM

Thèse de Doctorat

UFC

école doctorale **sciences pour l'ingénieur et microtechniques**

UNIVERSITÉ DE FRANCHE-COMTÉ

University of Franche-Comté | UFR Sciences, Techniques et Gestion de l'Industrie
FEMTO-ST/DISC | Doctoral school SPIM

Security supervision of mobile ad hoc networks: a lightweight, robust and reliable intrusion detection system

Ph.D. THESIS

will be publically defended on 12 July 2013

to obtain the title of

Ph.D. of Science of the University of Franche-Comté

Doctoral school SPIM - Speciality Computer Science

by

Mouhannad Alattar

Composition of the jury

Supervisor:	Julien Bourgeois	Professor at University of Franche-Comté
President :	Pascal LORENZ	Professeur at University of Haute Alsace
Reviewers :	Serge Chaumette	Professor at University of Bordeaux 1
	Pascale Minet	Researcher qualified to advise Ph.D students, INRIA Rocquencourt
Co-Supervisor :	Françoise Sailhan	Associate professor at CNAM

femto-st
sciences & technologies

Dedication

To the Syrian people who are struggling and suffering a lot in order to get their freedom ...

Acknowledgements

I would like to thank Prof. Pascal LORENZ (University of Haute Alsace) to preside the jury of my PhD defense. I am very thankful to Dr. Pascale MINET (Researcher at INRIA Paris Rocquencourt) and Prof. Serge CHAUMETTE (University of Bordeaux 1) who accepted being the reviewers of this thesis. The discussion that I did with them helped me to have valuable ideas about re-using my work in other interesting domaines.

I am indebted to my supervisor Prof. Julien BOURGEOIS (University of Franche-Comté) and co-supervisor Dr. Françoise SAILHAN (CNAM, Paris). Prof. Julien provided funding for my PhD studies and helped me in many research and administrative issues. He is a very kind, polite, and energetic person. Throughout my thesis, Prof. Julien gave me valuable working advices, motivated me to collaborate with other researchers in different universities, and encouraged me to continue working harder and pass a lot of challenges. During my thesis, I really enjoyed working with Dr. Françoise SAILHAN, and I hope that we will continue working together. She is a friendly and intelligent person. She spent a lot of time and efforts to enhance my skills as a researcher, especially in writing articles. Dr. Françoise helped me overcoming several difficulties during my thesis, even the personal ones.

I am thankful to the members of OMNI team in FEMTO-ST/DISC department for their full support. Prof. François SPIES, Dr. Philippe CANALDA, Dr. Pascale CHATONNAY, Dr. Christelle BLOCH, Dr. Eugen DEDU, Dr. Hakim MABED and others are very kind persons. I had interesting discussions with them, I collaborated with them in many lectures, and I did enjoyable sporting and social activities with them. During my thesis, I had the chance to meet special PhD students coming from different countries. It was a very nice and valuable experience to discuss and work with them. I would like to thank them all, especially: Dr. Mais HAJ RACHID, Dr. Wassim RAMADAN, Dr. Bogdan CORNEA, Dr. Matteo CYPRIANI. I am thankful to my best lab-mates: Dr. Raheel HASSAN, Dr. Wahabou ABDOU, and Osama ABU OUN who have helped and supported me a lot. I would like to thank my friends in the campus: Imane SAYAH, Soumia LARDJANE, Nejib MAIRECH, Ahmed ALKHATIB, Mostafa AKGUL, Asmae HAMZAOUI, and Halim BYDA. They largely helped and encouraged me especially during the last period of my thesis. I would thank all my friends and colleagues in Syria who encouraged me during the past years.

I am deeply indebted to my mother (Asma) and my father (Abdulatif) who have encouraged me to continue my studies and overcome all the challenges. In addition to provide funding for my first trip from Syria to France, they demonstrated patience and love. My mother and my father are always the basic reason behind all the successes that I did and I would do. I am very thankful to my brothers (Talal and Molham) and sisters (Kenda and Kenana) who have always loved me, encouraged me, and supported me.

Abstract

Mobile Ad hoc NETworks (referred to as MANETs) continue increasing their presence in our every day life. They are becoming a cornerstone in many domains and are used in military, science, logisitic and next-generation applications. However, these networks mostly operate over open environments and are therefore vulnerable to a large body of threats. Traditional ways of securing networks relying on preventive techniques, e.g. firewall and encryption, are not sufficient and should henceforth be coupled with a reactive security solution, e.g. Intrusion Detection System (IDS). Designing an IDS for MANETs is quite challenging because such IDS must not only ensure a high detection accuracy but also has to take into account the limited resources (e.g. battery life and bandwidth) and the dynamic nature of these networks. Moreover, the IDS should be protected against attacks and/or falsifications. In this thesis, we respond to these requirements by proposing a **l**ightweight and robust **i**ntrusion **d**etection system for ad hoc **r**outing protocols (LIDR). We first explore the attacks that threaten MANETs, focusing on the attacks targeting the Optimized Link State Routing (OLSR) protocol. We then describe LIDR that offers a high rate of attack detection, while limiting efficiently the resources consumption. Indeed, contrary to the existing systems that monitor the packets going through the host, our system parses and analyzes logs in order to identify patterns of misuse. We also introduce two metrics, the levels of suspicion and of gravity, which are used as to lower the communications and the processing on each host. To ensure the best management of the available resources, we also use the confidence interval as a measure of detection reliability. This statistical metric allows our IDS to: (i) identify the redundant evidences, hence the waste of resources resulting from gathering and processing them is avoided, and (ii) correctly make critical detection-related decisions. In order to enhance the robustness of our IDS, we couple it with an entropy-based trust model that assigns, based on the unlawful participations in the detection, low trustworthiness to the misbehaving nodes. Thanks to the estimated trustworthiness, our IDS reduces the bad effects of the falsified feedback provided by the misbehaving nodes. The proposed trust model is risk-aware such that the higher the risk of an attack the higher (resp. the lower) the trust in the nodes that helped in detecting (resp. hiding) it.

The proposed IDS and the coupled models have been experimented on different scenarios of mobility and density. The results show that LIDR offers a high detection rate along with a remarkable maintenance of the available resources. Moreover, it presents a significant robustness against the falsified detection-related evidences.

Keywords: Security, intrusion detection, ad hoc routing, resources maintaining, detection reliability

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

Over the last decade, the wireless industry has enjoyed unprecedented growth and success. Wireless communication continue invading every locations on our planet, even underwater areas. They are now central for personal use as well as for business and computer science. Moreover, the wireless mobile devices, laptops, smart phones and tablets, become essential in the daily life of millions of people. Such devices are equipped with wireless transceivers. Thus, these devices, even if they have different characteristics and goals, can share their resources in a network and constitute a Mobile *Ad hoc* NETwork (MANET).

A MANET is a self-organizing, self-healing, and autonomous wireless network. It is formed on-the-fly without the aid of fixed infrastructures (e.g., routers, radio base stations) or a central administration. A MANET is usually composed of mobile and resource-limited devices which are connected by bandwidth-constrained wireless links; the union of which forms a random topology. In MANET, two devices can communicate whether they are in the same radio coverage area or not. For the latter scenario, intermediate nodes are used to forward the packets from the source to the destination. Thus, each device in a MANET operates as a host and as a router, and needs a routing protocol.

Thanks to its special characteristics, i.e, infrastructure-less, self-organizing and low cost of deployment, MANET is highly recommended for providing collaborative communications and computing in highly dynamic and unpredictable environments (e.g., battle fields, rescue missions). Moreover, it offers a non-expensive, robust alternative or extension of infrastructure-based networks (e.g., airports, stadiums). However, the absence of administrative facility and fixed central points imposes more challenges and responsibilities on the devices that compose the network. In particular, each device should detect the presence of other devices in its radio ranges and further their types and attributes, cooperate with other devices in terms of forwarding the packets in the network, and guarantee its security as well as the security of the whole network through the cooperation with others.

Security is an essential condition for ensuring MANET's functions, e.g., packet forwarding and routing, especially under an adversarial environment: a misbehaving (or a compromised)

device could drop data packets passing through it or disrupt route calculation by, for example, injecting incorrect topological information in the network or obstructing the propagation of the routing control traffic. But, securing this type of networks is particularly challenging because they rely on an open radio-based medium of communication. Hence, the adversaries within the same radio range can launch their attacks without the need of a physical access into the network. In addition, MANET is by nature cooperative and therefore there is a lack of centralized management and security enforcement points e.g., switches and routers, from which preventive strategies can be launched. Moreover, the dynamic topology and the limited resources (e.g., battery life, computing capabilities, and bandwidth) complicate the deployment of security solutions in MANETs. Thus, traditional ways of securing the networks relying on e.g., firewall and cryptographic protection, should be enriched with reactive mechanisms, such as Intrusion Detection System (IDS for short), which constitute a second line of defense [10].

Intrusion detection system becomes a critical component of the security strategy in MANETs. However, the very decentralized and dynamic nature, the lack of fixed infrastructures, and the limited resources, make the deployment of a proper IDS for MANET extremely difficult. This challenge makes the intrusion detection in MANET a very interesting area of research. Generally speaking, an IDS for MANET should be cooperative and distributed so as to deal with the dynamic and the cooperative nature of such networks. In addition, an IDS performs three key functions: (i) evidence gathering so that information about the activities in the network and the devices' behaviors is gathered and analyzed, (ii) diagnosis that correlates these evidences so as to detect the attacks, and (iii) responding that takes the proper measures to stop or reduce the consequences of a detected attack. The majority of proposed IDSs depends on sniffing the traffic in order to extract the information and the evidences used during the diagnosis. The responding mechanisms could consist in avoiding the cooperation with the attackers, reinitializing the communication channels in the network, isolating or excluding the intruders and the non-cooperative devices, alerting the end user, and many other mechanisms. Since the diagnosis (or the detection method) constitutes the core of the IDS, a large portion of the literatures focuses on it. Therefore, there is a large number of proposed detection methods, which have profited from diverse sciences such as statistical and/or probabilistic classification, data mining, neural networks, graph theory, information theory, social engineering, etc. We note that the majority of proposed IDSs for MANET focus on providing a high detection accuracy and forget to take into account the criticality of the nodes performance. In fact, the amount of available resources, e.g., battery life, computing capabilities, and bandwidth, are restricted in MANET. Thus, an IDS that requires excessive calculation and/or radio transmission is not suitable for such networks even though it provides a high detection rate. Unfortunately, too few proposed IDSs consider the necessity of resources scarcity. Moreover, they aim at reducing the resources consumption uniquely in the detection method. Until now, no solution, to the best of our knowledge, considers reducing the consumption of resources during the evidences gathering phase.

The reliability of detection is another problem that faces IDS in MANET. Since this type of network uses an open medium of communications and the devices can join or leave freely, misbehaving devices are more likely to be found. These misbehaving (or compromised) devices aim at falsifying the intrusion detection through different ways: they may refuse to cooperate during the detection, or they may inject incorrect evidences that lead to either accusing well

behaved device(s) or protecting a misbehaving device(s). Moreover, the dynamic topology and the mobility of devices may lead to different security and topological points of view between well behaved devices. Hence, they would provide inconsistent evidences for the IDS. Such inconsistent and/or falsified evidences make the results of the employed IDS under question. Therefore, there is a need for not only a mechanism that prevents the falsified evidences, but also a measure of reliability that presents to what extend the results of the detection are reliable. Such measure helps also in making the critical decisions in the IDS such as concluding that a device is an intruder, and further launching a proper responding mechanism. To the best of our knowledge, such factor of reliability has never been proposed for the intrusion detection in MANET.

Overall, there is a need to use an IDS in MANET so as to detect and react against the security threats which succeed in exceeding the conventional security solutions. This IDS should take into account the following issues:

- the adaptability to the cooperative and the dynamic nature of MANET,
- the necessity of providing a high detection rate along with a limited number of false alarms,
- the criticality of maintaining the resources along the different phases of detection,
- the necessity to being robust against the attacks and misbehaving devices,
- the necessity of measuring the reliability of the detection conclusion, thus the consequences of incorrect detection-related decisions are avoided.

This thesis aims at addressing the aforementioned issues by designing a resources-aware and robust intrusion detection system for MANETs. We are interested in detecting the attacks targeting the *ad hoc* routing protocol since routing is one of the most vital functions in MANETs, and hence disrupting the correct operation of the routing protocol could cause the most devastating damages. Our IDS is build on a distributed architecture so that all the devices participate in detecting the attacks. It also considers many aspects like detection accuracy, maintaining the resources, enhancing the robustness against the misbehaving devices, and the reliability of detection. The proposed IDS is oriented to work with OLSR protocol. However, it can be adapted for other *ad hoc* routing protocols.

1.2 Contribution of this Thesis

Our contribution concerns the design and the evaluation of a resources-aware and robust intrusion detection system for MANET. It includes the following points:

- **Designing and implementing a log-based and resources-aware IDS**. As a first step towards this target, we classify and model the known attacks targeting the OLSR protocol. The modeling is done thanks to an improved temporal model that is derived from the one introduced in [11]. This improved model enables describing the relationship between the attack's actions and their related consequences. Based on the modeled attacks, we build intrusion signatures; an intrusion signature is thought as a partially ordered sequence of events that characterizes a misbehaving activity. After that, we illustrate the distributed

and the cooperative architecture of our IDS. This IDS aims at reducing the consumption of resources during two phases: evidences gathering and intrusion diagnosis. During the evidences gathering, our IDS analyses the local routing logs instead of sniffing the traffic in the network. This analysis consists in matching the suspicious routing-related operations with the predefined intrusion signatures so as to extract evidences of attacks. During the diagnosis, the evidences of attacks are classified according to their gravity. The goal of this classification is to activate an in-depth diagnosis only if a sufficient level of suspicion exists. Thus, the number and the duration of in-depth diagnoses, which are costly in terms of resources consumption, are restricted. In fact, an in-depth diagnosis includes asking other node(s) to gather and/or to analyze more evidences. The performance of our IDS in terms of detection accuracy as well as resources consumption are further evaluated and compared to other known IDSs. The performance evaluation is done in a simulated MANET coupled with virtual machines. This is the first time that such an evaluation environment is used for experiments on intrusion detection.

- **Coupling a risk-aware trust model with our IDS**. The aim of this coupling is to enhance the robustness of our IDS against the false evidences provided by misbehaving node(s). We propose an entropy-based trust model that increases (resp. decreases) the trustworthiness of a node each time this latter provides a correct (resp. an incorrect) diagnosis-related evidence. During the diagnosis, any evidence is first weighted according to the trustworthiness of its source before being used. We further associate the trustworthiness of a node with the risk of the attacks that this node helps in detecting (resp. hiding) them. This association raises the preventive nature of our IDS along with the level of danger in the network.

- **Employing the confidence interval as a measure of the detection reliability**. Having a measure of the reliability helps in making critical detection-related decisions like the activation of a responding mechanism. Moreover, it determines exactly whether gathering more evidences will not change the diagnostic result (or the change is negligible). Thus, our IDS is able to precisely determine when to stop (or at minimum reduce) gathering more evidences since they are redundant. Consequently, the resources consumption triggered by the detection is significantly dropped.

1.3 Plan of the Dissertation

The thesis is organized in five chapters which are further divided into multiple sections. Hereafter, an overview of those chapters are provided:

Chapter 2 presents a state of the art in the domains that inspire this thesis. This chapter starts by providing a detailed overview of MANET (its origin, special characteristics, domains of applications, and future developments). Then, details are given for the security issues in this type of networks and a review of the existing intrusion detection systems in MANET is presented. We focus on the detection methods, addressed attacks and the performance of the existing IDSs. These latter are further classified into several categories according to their detection method. Moreover, a comparison between the IDSs is also provided. This chapter terminates with a

highlight of the common drawbacks of the current IDSs for MANET that should be resolved.

Chapter 3 introduces our IDS (its structure, detection method, implementation, and performance evaluation). This chapter starts by providing a detailed overview of the OLSR protocol. Then a detailed description of the known attacks targeting OLSR is presented. These attacks are further categorized and modeled. Afterwards, we illustrate the architecture of our IDS with a detailed description of the key components. We then illustrate how our IDS uses the logs so as to extract evidences without consuming too much resources. The in-depth intrusion diagnosis of our system is further explained. The link spoofing attack is then used to provide a real example of the various operations and phases of our IDS. This includes defining the link spoofing signature, and illustrating the corresponding algorithm of diagnosis. The last part of this chapter presents the first series of performance evaluation of our IDS. We introduce the evaluation environment wherein the simulated nodes using NS3 are coupled with an operating-system-level virtual machines. The performance is evaluated in terms of detection accuracy and of resources consumption. It further studies the impact of mobility and density on our IDS. This part concludes with a table that compares the performance of our IDS with the performance of the IDSs already listed in the previous chapter.

Chapter 4 presents the enhancement of the robustness and the reliability of our IDS. This chapter starts by a description of the trust models in MANET. The entropy-based trust model that is coupled with our IDS is then presented. The properties that may affect the establishment of trust relationships between the nodes are described. We propose a new property that depends on the attack risk in order to enhance the preventive nature of our IDS. This part is ended with a series of experiments that aim at evaluating the performance and the benefits of the proposed trust-model in several cases. The second part of this chapter starts by a detailed description of the inferential statistics and of the confidence interval. We demonstrate the benefits of using the confidence interval in order to estimate the opinion of a large population. Then, a detailed mathematical explanation for estimating such intervals and its application as a measure of reliability are presented. This measure serves to significantly minimize resources consumption. Moreover, it helps in making critical decisions during the intrusion detection. This part concludes with a new series of experiments that illustrate the important benefits that are obtained by using the confidence interval in the intrusion detection.

Chapter 5: The general conclusion illustrates that the goals of this thesis are successfully achieved. Future work and useful propositions to enhance the security issues in MANET are then presented.

CHAPTER 2

INTRUSION DETECTION IN MANET

Contents

2.1 Introduction

In this chapter, we introduce the basic concepts related to wireless *ad hoc* networks and to the intrusion detection in order to provide the reader with the background necessary for this dissertation.

In the last few years, wireless mobile industry has grown exponentially and has changed our daily routines. Wireless mobile networks and devices, e.g., laptops, tablet computers, and smart phones, are becoming very popular since they offer the access to the communication and information any time and any where. The conventional wireless network is usually supported by fixed infrastructures; the mobile devices use a single-hop wireless radio communication to forward their packets to a base station which is connected to a wired network [12]. In contrast, a *Mobile Ad hoc NETwork* (MANET for short) provides a peer-to-peer organization of the communication between the mobile devices (hereafter also called nodes) without the need for any infrastructure or centralized administrator [13]. In fact, each device in MANET communicates directly with other devices in its radio range (i.e., via a single hop path). The communications between the devices, which are not in a direct radio range, are done through intermediate devices (i.e., via multi-hop paths). Thus, each device in MANET operates, both, as a host as well as

a router. Thanks to its independence from fixed infrastructure, MANET is very useful in the environments where deploying the networking infrastructure is difficult and/or costly, in terms of time and resources [14]. For instance, MANET is a proper solution to provide collaborative communications and computing in small areas (e.g., office or building organizations), temporal events (e.g., conferences), and unpredictable environments (e.g., battlefields and disaster recovery areas). MANET is further used as a non-expensive, robust alternative or extension of infrastructure-based networks, especially in hotspot sites such as airports or stadiums. Recently, MANET inspires future networks, e.g., Vehicular *ad hoc* network, Internet of Things[1], underwater sensor networks, to find solutions that guarantee the continuity of communications even in the absence of fixed infrastructure [13]. Overall, MANET continuously increases its presence in our life. Because of the mobility and the absence of, both, fixed infrastructure and central administrator, MANET poses a new design challenge: self-configuration. Indeed, the topology of this network may change randomly and unexpectedly due to the arbitrary mobility of the nodes composing MANET. Hence, every node should be able to handle, by itself, the problems of: detecting the presence of other nodes in the neighborhood, discovering the services available in the network, topology maintenance, IP addressing, and route calculation. Answering these challenges is far from trivial, especially that any proposed protocol or solution should take into account the limited resources (e.g., bandwidth, battery power, processing capabilities) in MANET. Therefore, MANET is considered as one of the most innovative and challenging areas of wireless networking [15].
Similarly to any information system or network, securing MANET is essential in order to guarantee the quality of the provided services. More precisely, there is a need to: (i) guarantee the availability of services and communication provided by MANET, (ii) protect the information exchanged between the mobile devices from malicious and/or unauthorized modifications or even destruction, and (iii) prevent unauthorized disclosure of information/data, and thus ensuring the confidentiality in the network. But, MANET is more vulnerable to threats and misbehaving than infrastructure-based networks. This results from its special characteristics [16]:

- Unreliability of wireless links between the nodes, that results from, both, the mobility and the limited energy supply of those nodes.
- Difficulty of physical protection of the mobile nodes since these latter are easy targets for stealing.
- Dynamic and arbitrary topology resulting from the continuous and unpredicted movements of the nodes.
- Lack of central points of control or certification authorities.
- Limitations on battery life, computing capabilities, and bandwidth.

In order to eliminate, or at minimum reduce, the exposure to security threats, many variant security solutions are oriented to work in MANET. These solutions are either (i) proactive solutions that consist of security-aware protocols and application design (e.g., secure routing

[1]Internet of Things aims to enable seamless communication of moving smart objects with nodes on the Internet.

protocol, communication encryption), or (ii) reactive solutions (e.g., Intrusion Detection Systems) which are seen as a second line of defense. In fact, the reactive solutions aim at handling the threats that have succeeded in penetrating the proactive solutions. The proactive solutions are out the scope of this dissertation. We instead focus on the reactive solutions, and more specifically on the Intrusion Detection System (IDS for short). Intrusion detection is the process of identifying any (malicious) unauthorized use, abuse, and misuse of the information system or network [17, 18]. It aims at discovering any violation of the defined security policy[2] before it leads to a security failure [20]. Such failures happen when the system or the network fails in providing secure service(s). The IDS is seen as an implementation of the mechanisms and the practices of the intrusion detection. According to the the Common Intrusion Detection Framework (CIDF) model and the IETF intrusion detection working group (IDWG)[3], any IDS has four key components:

- Event box (e-box for short): is responsible for gathering evidences about the security threats or intrusions.
- Database box (d-box for short): is responsible for saving the gathered evidences. It may also apply some preprocessing on these evidences so as to, e.g., normalize them in a common formate.
- Analysis box (a-box for short): is the core of the IDS that analyses and correlates the gathered evidences in order to detect the intrusions.
- Response box (r-box for short): is concerned with the possible reactions/responses that are taken upon the detection of an intrusion (e.g., isolating the attacker, alerting).

Many IDSs have been proposed for infrastructure-based networks. However, such IDSs cannot be directly used in MANETs because of the MANET 's specific features, e.g., the lack of fixed infrastructure, the limited resources, and the dynamic topology [21, 22]. Therefore, new approaches need to be developed for MANET. In the literature, the works that concern the IDS for MANETs can be categorized into two large disciplines: design of the architecture of IDS, and detection mechanisms. However, some works, but few of them, handle other issues such as evidences gathering, and countermeasure strategies. In general, the existing architectures of IDS in MANET fall under three classes: stand-alone, cooperative and distributed, and hierarchical [23]. Regarding the detection mechanisms used in the IDS for MANET, we can find the three well-known categories: (i) anomaly detection which searches for the deviations from the normal expected behavior of the node, and further considers them as attacks, (ii) signature-based (or misuse) detection which compares the observed and/or gathered events to scenarios of known attacks, and thus considers any matching as an attack, and (iii) specification detection which extracts constraints from the specifications of the, e.g., applications and protocols that are used in the network/the hosts, and then considers any violation of these constraints as an attack. However, looking deeply into the different detection mechanisms illustrates that a variety of techniques derived from different sciences have been used. More precisely, the detection mechanism may use

[2]Security policy is a formal statement of the rules/constraints by which users/devices who are given access to a network/system must abide [19].

[3]http://datatracker.ietf.org/wg/idwg/charter/

techniques that are based on statistics, information theory, data mining, graph theory, neural network, social engineering, or trust.

The remaining part of the chapter is covering the following topics. Section 2.2 provides an overview about MANET; its history, domains of use, and characteristics. Section 2.3 represents some of the reference IDSs in MANET. These latter are further classified according to their detection mechanisms. At the end of this section, the performance of the mentioned IDSs is compared in terms of detection accuracy and resources consumption.

2.2 Mobile Ad hoc Network

The *Mobile Ad hoc NETwork* (MANET) is originally designed for military purposes in the 70's and 80's (called at that time Mobile Packet Radio Networking) [24]. The goal was to employ a self-configuring, self-healing, and autonomous network in the battlefield - a highly dynamic and unpredictable environment. MANET is a dynamic and infrastructure-less network that is generally composed of limited resources and free to move arbitrarily nodes [25]. Furthermore, they are, usually, a priori unknown to each other and can join and leave freely the network [26]. These nodes are connected by respectively bandwidth-constrained wireless links - the union of which form a random topology. Nodes can communicate with others that are either inside or outside their radio ranges. For the latter scenario, intermediate nodes are used to forward the packets towards the destination.

Nowadays, applications for MANET technology are widely diverse: MANET provides communication for pollution-, emergency- and safety- applications [27, 28], vehicular communications [29], Unmanned Aerial Vehicles (UAVs) swarms communications [30], or any other scenario that requires self-organizing and fast-deployable communications with survivable, efficient dynamic networking. MANET may operate in a standalone fashion, or may be used to extend the communication for a larger (wireless or wired) infrastructure-based network. This independence of the equipments encourages the use of MANET in rescue missions and after-disaster situations where it is hard and/or costly to deploy an infrastructure-based network. MANET may further be a non-expensive, robust alternative or extension of wired/cell-based mobile networks in popular event sites, e.g., airports or sport stadiums, wherein network traffic demand is huge [31]. Moreover, it can be used to realize a low-cost, large-scale wireless coverage in urban areas (so-called Mesh networks) [32]. In fact, some nodes in MANET may work as gateways; they have, in addition to MANET interface(s), interface(s) connected typically to non-MANET network(s). Thus, the traffic generated inside MANET can reach the outside nodes that are belonging to non-MANET networks and vice versa through these gateways. A MANET is characterized by the following properties:

Dynamic Topology - The topology of MANET may change rapidly and randomly in an unpredictable manner for two reasons [25]. First, nodes in MANET are free to move arbitrarily. Second, wireless connectivity significantly varies over the time because of the nodes' transmission power variation, or interference. This dynamic topology poses a challenge of performance and scalability. Thus, the communication routes should be updated rapidly and accurately in MANET.

Infrastructure-less - MANET is an infrastructure-less network that does not rely on fixed network equipments, e.g., radio base stations and wired/wireless routers. Indeed, the routing functionality is incorporated into the nodes: each node (i) detects its adjacent nodes, (ii) discovers the route - a route is seen as a sequence of intermediate nodes - to any destination, and (ii) forwards the packets for other nodes.

Self-configuration - Fixed structures typically do not exist in MANET, thus nodes are autonomous and have to configure themselves on-the-fly [33]. Self-configuration consists in optimizing the auto-configurable parameters inside the node [34]. Note that automatic re-configuration is also required when predicted or unpredicted changes happen. Both, i.e., automatic configuration and re-configuration should be done with minimal or no human intervention. In general, self-configuration in MANET:

- Automatically discovers the node's neighbors to whom communication links are set up. This discovery is specific to the radio access technique used. Note that if a node has more than one radio interface, then each one of those performs its own neighbors discovery process. Furthermore, a node should have the ability to automatically announce/discover the service(s) in the network. In [35], authors propose to have an up-to-date service directory in the cluster-head of a hierarchically-organized MANET. Other nodes can then simply search for the required service(s) in this directory. More advanced service discovery and delivery protocols, e.g., Konark [36] and Allia [37], have been proposed in order to accommodate the special characteristics of MANET. In fact, these protocols operate in completely distributed, peer-to-peer fashion, and take into consideration device capabilities and limitations.

- Obtains an IP address. Two approaches are proposed: stateful and stateless. A stateful approach consists in tracking the free and the used addresses through one or several synchronized allocation tables, which are maintained by some nodes [38, 39]. Thus, these nodes play the role of DHCP server and assign free addresses to the un-configured nodes. Whereas in a stateless approach, an un-configured node, which wants to join the network, selects a random IP address and then broadcasts it to validate its usability [40]. In principle, the first approach leads to zero address collision but requires a synchronization between the allocation tables so as to ensure that any used address is in the allocation table. While the second one provides more flexibility and scalability but it may lead to address duplication [34].

To summarize, self-configuration issues still constitute an active research field for MANET [41].

Limited Resources - Most of the nodes that form a MANET (e.g., laptops, PDA, and Internet mobile phones) are characterized by limited battery life, computing capabilities, storage size, as well as communication capacities, bandwidth and transmission power. These limitations constitute a critical issue; applications/protocols oriented to work in MANET should consider the frequent occurrence of packet loss and congestion as long as the necessity to conserve the resources. Since the node operates as, both, an end terminal and a router, optimizing packet forwarding constitutes a cornerstone for resource conservation. This optimization is investigated

at different layers [42], e.g., using of directional antenna [43], avoiding unnecessary transmissions and re-transmissions [44], reducing to minimum the number of routing related messages, using power-efficient error control schemes, and reducing the size of packets. Another factor that impacts the design of applications/protocols is the heterogeneity, in terms of resources, of the nodes [45]. This implies that the most powerful nodes should be charged with the roles that consume more resources (e.g., acting as *cluster-head* in hierarchical MANET).

Routing - Each node operates as a router relying on a routing protocol and maintaining the related routing table. The Internet Engineering Task Force (IETF) MANET working group[4], standardizes the IP level routing protocol suitable for MANET. MANET working group proposes two standard types of routing protocol:

- **Reactive MANET Protocol (RMP)**: in this type of protocols (also called on-demand protocol), a route is found only when it is needed. More precisely, when a source needs to send packets to a destination node for which the source does not already have a valid route, this latter broadcasts a route request. Once the route request reaches the destination, or an intermediate node that has a valid route to the destination, this latter sends, in the reverse route, a route reply. As the route reply propagates back to the source, intermediate nodes set up forward pointers to the destination. Hence, the route from the source towards the destination, or any intermediate node, is established. On the one hand, reactive protocols, e.g., AODV [46, 47], AODVv2 [48], and DSR [49, 50], give a reduced average of traffic overhead as a burst of messages are generated only when a route is required. On the other hand, they suffer from an additional transmission delay because of the lack of immediate routes. To reduce this delay, some of these protocols propose that every node maintains a route cache so as to avoid doing a route discovery for already known routes.

- **Proactive MANET Protocol (PMP)**: a proactive protocol (also called table driven or periodic protocol) is characterized by maintaining a constantly updated routing table on each node. For this purpose, each node exchanges periodically its point of view of the network topology with others. Henceforth, available routes in the network can be calculated. Opposite to the reactive protocols, the proactive protocols, e.g., OLSR [51, 9], DSDV [52], and TBRPF [53], guarantee instantly available routes. But, they lead, in general, to more overhead traffic.

In order to cumulate the benefits of *reactive* and *proactive* approaches, **hybrid** protocols, e.g., TZRP [54] and CBRP [55], have been proposed for MANET. The basic idea is to organize the network into zones or clusters. Inside the zone/cluster a proactive approach is applied, i.e., updated routes are maintained between the nodes composing the zone/cluster. While the routes between nodes belonging to those zones/clusters are found thanks to a reactive approach. Nowadays, there is a large number of proposed reactive, proactive and hybrid MANET routing protocols. However, only four of those: AODV, DSR, OLSR and TBRPF are proposed for experimental Request For Comments (RFC for short) by IETF MANET working group. Furthermore, AODV and

[4] http://datatracker.ietf.org/wg/manet/charter/

OLSR are considered as the most mature MANET protocols whilst most of others suffer from the lack of either updated or freely available[5] implementations [41].

Security - A MANET is much more vulnerable to security threats/attacks than an infrastructure-based networks [10]. This results from three reasons. First, such a network relies on an open radio-based medium of communication, hence the adversaries within the radio range can easily launch the attacks [21, 42]. Second, the lack of centralized administration/security enforcement points e.g., switches and routers, from which preventive strategies (e.g., firewalls and encryption software) can be launched [25]. Thus, each node is vulnerable and should guarantee, by itself, its own security [56]. Third, the dynamically reconfigurable network topology and the resource constraints, both, complicate the development of security solutions in MANET [31].
Apart from the attacks targeting the network layer, the other attacks targeting application, link and physical layers, are shared with infrastructure-based networks. Recall that, in MANET, the communication between the source and destination depends on the cooperation of the intermediate nodes that relay packets, therefore routing attacks have attracted significant attention [57]. Many security schemes, e.g., authentication [58], encryption [59], access control [60], and digital signature [61], have been proposed so as to secure *ad hoc* routing protocols, leading to secure versions of AODV and OLSR protocols, based on digital signature technique [62, 63] and [64] respectively. However, these conventional mechanisms aim at preventing outside attacks but they do not address the problem of compromised nodes. Moreover, new attacks emerge and find a way to penetrate the aforementioned mechanisms [65]. Therefore, there is a need to detect and deal with attacks. To that end, an Intrusion Detection System (IDS) is an indispensable part of a security scheme [66].

2.3 Intrusion Detection

Intrusion Detection Systems (IDS) [67] have been proposed to constitute a second line of defense [68]. IDS aims to identify "any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource" [69]. They can operate at different layers, e.g., link, network, application layers, and even on the security policy [70]. In MANET, the IDSs that have been initially developed for infrastructure-based network have failed because of the special characteristics of MANET: the open medium of communication, the dynamic network topology, the limitations of resources, and the absence of centralized administration/security enforcement points [21, 22]. Thus, there is a need to design IDSs specifically for MANET.

Most of the IDSs proposed for MANET are **distributed and cooperative** [14]. In practice, each node depends on locally gathered evidences or audit data, i.e., the chronological records that describes system/network activities, so as to detect the attacks. In addition, it exchanges information and/or alarms with other nodes in order to detect wider attacks [68, 71]. Beside distributed and cooperative architecture, it is possible to find standalone or hierarchical IDSs [23].

In **standalone architecture** [72, 73], each node detects individually the attacks based on its own evidences and does not cooperate with others. Standalone IDSs generally offer a modest

[5] A TBRPF implementation was previously available from Stanford Research Institute (SRI), but has since then been retracted.

detection accuracy and address limited range of attacks [14]. On the other hand, such IDSs are suitable for the MANETs where some of the nodes cannot or do not want to participate in a cooperative detection. Here, a stand-alone IDS is installed uniquely on the powerful nodes.

Hierarchical architecture [74] has been proposed for multi-layered network infrastructures where nodes are organized into clusters. For each cluster, one node is elected as a *cluster-head* according to topological, security- and/or resources-related criteria. The *cluster-head* is responsible for regulating (i) the detection inside its cluster and (ii) the cooperation by exchanging detection-related evidences and/or alarms, with other clusters. Although this architecture scales, the maintenance of the clusters imposes additional challenges for the IDS.

Regardless of its architecture, an IDS performs usually three main functions: audit of the data/evidences that are collected, attack detection, and responding [14]. Evidences are continually collected and used to find the signs of attack. The collection of evidences can be done locally or globally (i.e., evidences are provided by other nodes) [75]. Depending on the nature of the attacks, evidences can include user and system call activities, communication and routing activities, and security-related alarms [68]. In MANET, most of the IDSs collects evidences by promiscuously monitoring the wireless communications to collect evidences [74]. This method of collection provides direct evidences (for nearby traffic) and it avoids the need to rely on evidences from other nodes, which might lie. But, it consumes a lot of energy, and may be unreliable under high collision rate [72].

Responding, represents the reaction that should be taken upon detecting an attack (e.g., taking countermeasures). It depends on the protocols and/or programs used, along with the type of the attack [68]. In general, most of IDS proposes traditional responses such as broadcasting alarms, notifying the end user, precluding the attacker from participating in packet forwarding process, refusing to forward attacker's packets, and reinitializing the communication channel[75]. Recently, more sophisticated response mechanism has been proposed so as to take into account the special characteristics of MANET. For instance, authors in [57] propose a risk-aware countermeasure that aims at balancing the damage, in terms of network performance, of ignoring an attack against launching incorrectly a countermeasure. In practice, an attacker is isolated as long as this isolation does not lead to a network partition.

The cornerstone of an IDS is the detection mechanism employed (so-called method of analysis). Classically, detection mechanisms are classified into anomaly-, misuse- (so-called also signature-), and specification-based detection [23]. **Anomaly detection** reports as an intrusion every deviation between the standard behavior acquired during a training phase and the current behavior of the node. This model may detect unknown attacks but it generates a high rate of false alarms, i.e., considering a well-behaving node as an attacker. The basic challenge of anomaly detection in MANET comes from the difficulty of building the normal behavior profiles in such dynamic network. **Misuse detection** reports as intrusion any match between a series of observed events and a predefined signature; an intrusion signature is thought as a partially ordered sequence of events that characterizes a malicious activity. This model offers a high accuracy but it requires that the signatures are always complete and up-to-date. **Specification detection** reports as an intrusion any violation of a set of constraints on the protocols and/or the programs used. These constraints are extracted from the specifications that describe the correct behavior/functionalities of the protocols/programs. This model could detect unknown

attacks while exhibiting a low rate of false alarms. However, extracting the operational constraints from the specifications is a time consuming task, which should be repeated for every new (or modified) protocol and/or program. Moreover, this model cannot detect the attacks that do not violate directly the specifications, e.g., DoS (Denial of Service) attack [76]. In the literature, IDSs for MANET use a large variety of detection mechanisms. However, the anomaly- and specification-based detection mechanisms occupy the biggest space, while few research on signature-based detection have been done [14].
We believe that the evaluation of an IDS should not be based hastily on its detection model for a twofold reason. First, many IDSs use a hybrid detection mechanism that belongs to more than one detection model [75, 76]. Second, the importance of an IDS comes from its capabilities of detecting the targeted attack(s), along with taking into account the special characteristics of MANET (e.g., dynamic topology and limited resources). Thus, we do not follow the previous high-level classification of the detection mechanism for presenting the most significant IDSs in MANET. We rather go deeply and describe the mathematical approach of the detection mechanism used, its advantages and disadvantages, and its suitability for MANET.

2.3.1 Classification-based Detection

In classification-based detection, statistical significances are used to distinguish well-behaving from misbehaving nodes by comparing the observed operational profile of the node (or the network) with the normal profile(s) which are built in training phase and in the absence of attacks [77]. Indeed, a group of features (e.g., number of advertised neighbors, delay of hello messages, power consumption) is selected so as to describe the profile. Then, statistical properties of these features are analyzed within a time window, and compared to their equivalents in the normal profile. This comparison is done usually by using computational intelligence techniques (e.g., neural networks, data mining, SVM [78]), and aims at discovering the deviations from normal profile. A discovered deviation confirms the occurrence of the attack.

The IDS proposed by Zhang and Lee [68, 79] is considered as a de facto standard for the IDSs in MANET. It aims at detecting the attempts to falsify the routes provided by AODV, DSR and DSDV routing protocols. During the training phase, the impact of the node's movement on the percentage of changes in the routing table is analyzed. These changes come from the movement (i.e., velocity, direction and position), and are provided by a Global Positioning System (GPS). Then, during an operation phase, an actual percentage of changes differing from the predicted one, is defined as an anomaly. In practice, each node classifies the percentage of changes in its routing table as either predicted or unpredicted. This classification is done either by the Support Vector Machine (SVM) Light [78], a high-dimension hyperplane-based classifier, or RIPPER [80], a rule-based classifier. An unpredicted change is then considered as an attack. If the locally gathered evidences are not sufficient to have a certain detection result, node exchanges the result of the local detection with its neighbors so as to deduce a global decision. Simulation-based evaluation shows that the SVM classifier outperforms significantly RIPPER. In addition, the highest detection accuracy is obtained when the routing protocol was DSR. Choosing a proper time window for the analysis is a critical factor for limiting the amounts of false alarms. Moreover, the malicious node(s) may dominate the global decision and send false accusations and/or false praises (i.e., *blackmail* attack), thus this decision may not be reliable [77].

The hierarchical IDS proposed in [81] detects *blackhole* and routing request flooding attacks targeting the AODV protocol. In practice, a MANET is organized into clusters so that the cluster's member with the highest residual energy and number of connections is elected as a cluster-head. In addition, the cluster-head is periodically re-elected so as to fairly distribute the working load. Each cluster-head searches for the abnormal behaviors of the members of its cluster. In practice, three features are selected to characterize the normal behavior profile during a training phase: the percentage of changes in the routing table, the propagation of the routing packets, and the propagation of the data packets. During the operation phase, each cluster-head monitors the cluster's members either randomly or deterministically. The random monitoring consists in selecting randomly a cluster member so as to transmit its own set of features to the cluster-head. While the deterministic monitoring consists in listening to the whole traffic generated in the cluster. A deviation, which exceeds a certain threshold, between the monitored and the expected value of a feature is confirmed as an attack. Such deviations are identified thanks to a one-class Support Vector Machine (1-SVM) classifier [82]; 1-SVM needs to be trained with only normal or abnormal scenarios but not both of them.Since the nodes' connectivity is considered during the cluster-head election, the elected cluster-heads monitor the activities of a large portion of network. Thus, this IDS offers a high detection accuracy. However, the cluster-heads may become failure points. Furthermore, a malicious node may foil the detection process by transmitting to its cluster-head false features.

In [83], a *blackhole* attack targeting the AODV protocol is detected by investigating some features like the number of route requests/replies, or the average difference of sequence numbers in the routing messages[6]. If the distance between an actual value of a feature and the average value (as recorded during the training) exceeds a given threshold, then an intrusion is reported. This work distinguishes itself by continuous training in which the training data is updated in every given time interval. In practice, if the data collected in the time interval δ_i is considered as normal, i.e., the average difference of sequence numbers does not exceed the given threshold, then the corresponding data will be used as training data in the next time interval δ_{i+1}. Otherwise, it is discarded and the former training data is maintained. Thus, the training data is adaptively defined according to the changing network environment. However, this training method may constitute a point of failure. More precisely, if an attack is not detected within a time interval, then the collected data for a malicious activity will be used as training data for the following intervals.

In [73], the authors propose a two-stages application-based detection. In the first stage, Maxima Detection System (MDS for short) is used to detect (almost) immediately a potential attack. This rapid detection of an attack helps in defining a threshold to calibrate the Cross-Correlation Detection System (CCDS for short) during the second detection stage. MDS analyzes the peaks of the Probability Density Function (PDF) [84], which is statistically generated from the observed interactions between the nodes at the application layer. By confronting the discovered peaks of the PDF to a normal profile that is created offline, MDS identifies rapidly the outlying peaks that characterize the suspicious interactions. If a suspicious interaction is identified, CCDS is activated and uses the detected suspicious interaction to calibrate a threshold. CCDS calculates then the average of the application-related interactions of each node and compares those with

[6]Largely increased sequence numbers are known as a sign of *blackhole* attack

the threshold. The nodes which have their average interactions exceeding the threshold are considered as misbehaving. This combination of two detection techniques increases the accuracy of detection. In addition, this two-stages detection method triggers an increase in the memory usage equals to only 7.4MB. However, the proposed IDS is prone to false positive, especially when nodes are mobile because the threshold of the CCDS is calibrated only once during the startup.

In [85], a more sophisticated Cross-Features Analysis (CFA) is applied to detect both *blackhole* and *grayhole* attacks on the AODV and DSR protocols. These features including the reachability between two nodes and the number of delivered packets, are analyzed within a time windows. This analysis attempts to quantify the relation existing between one feature f_i and the remaining features, i.e., $f_1, \cdots, f_{i-1}, f_{i+1}, \cdots, f_k$ (with $k+1$ defining the number of features analyzed). During the operation phase, the probability of matching between the predicted feature f_i and the observed feature f'_i, which is established based on $f'_1, \cdots, f'_{i-1}, f'_{i+1}, \cdots, f'_k$, is calculated by a decision tree classifier named C4.5 [86]. If this matching probability is less than a given threshold, then an intrusion is confirmed. The used threshold is the lower matching probability that was observed during the training phase. Simulation-based evaluations show that the C4.5 classifier outperforms, in terms of recognizing abnormal behaviors from normal behaviors, other classifiers such as RIPPER. However, the size of the sampling data, on which the correlation between the features is calculated, may impact largely the detection accuracy. Moreover, the detection accuracy of this IDS is not provided. Similarly, the IDS proposed in [87] applies the CFA and the C4.5 so as to detect flooding and *blackhole* attacks targeting OLSR and AODV protocols. Herein, traffic-related features (e.g., number of received/transferred packets) and OLSR- or AODV-related features (e.g., MPR updates for the former and route discovery for the latter) are used for the classification. This IDS distinguishes itself by proposing to aggregate the detection result at several hierarchical levels. More precisely, a graph-based clustering scheme is used to organize the MANET into 3 hierarchical levels: nodes, cluster-heads, and managers. Each node depends on a cross-features analysis (CFA) technique along with the C4.5 classifier so as to detect the deviation between an expected and an actual value of a feature. Then, it sends the anomaly index, i.e., the average of the classification result of the entire monitored features, to its cluster-head. The anomaly index is further averaged at 2 levels: at the cluster-heads, which average anomaly indexes from neighboring nodes, and at the managers, which average anomaly indexes from the cluster-heads. Simulation-based evaluation shows that the detection accuracy increases along with moving up in the hierarchy. In addition, this IDS performs better on AODV than on OLSR since the selected attacks increases AODV overhead, and thus abnormality becomes more noticeable. However, a malicious node may foil the hierarchical aggregation of detection results by sending false accusations/praises.

A cooperative IDS that uses three parallel detection engines working on MAC, routing, and application layers is proposed in [88]. The use of multi-layers detection is motivated by the fact that the attacks which target upper-layer protocols can be seen as legitimate events at lower-layers, and vice versa. The Mac-layer detection engine employs a Cross-Features Analysis (CFA) technique like the one used in [85]. In practice, Naïve Bayes [89], a probabilistic classifier, is used to calculate $P_i(f_i|f_1, f_2, \cdots, f_{i-1}, f_{i+1}, \cdots, f_k)$ (with $k+1$ defining the number of features analyzed), which defines the probability that a classification feature takes the value f_i when

the other features have the values $(f_1, f_2, \cdots, f_{i-1}, f_{i+1}, \cdots, f_k)$. During the operation phase, if the difference between the actual and the expected probability of a feature f_i exceeds a certain threshold, then an anomaly is confirmed. The routing layer detection engine depends on the Markov chains (state transitions) to discover the anomalies in the calculated routes. More precisely, let s_1 represents the state when the number of hop counts takes the value $V1$, while s_2 represents the state when the number of routes takes the value $V2$. The probability of the transition from the state s_1 to the state s_2 is calculated as $P(s_1, s_2) = N(s_1, s_2)/N(s_1)$. Note that $N(s_1, s_2)$ is the number of times where, both, s_1 and s_2 have taken place. While $N(s_1)$ is the number of times where s_1 has taken place, regardless of other states. During the training phase, the probability of the transition $(s1, s2)$ is estimated. During the operation phase, if the difference between the calculated and the expected probability (i.e., the one that is estimated during the training phase) for the transition $(s1, s2)$ exceeds a certain threshold, an anomaly is confirmed. In the application layer, an association rule mining technique [90] detects abnormal frequencies of features related to the source node, the destination node, and the received packets. Nevertheless, there is a lack of details about how the aforementioned techniques are used by the detector. In every node, the results of the three detection engines are combined. In addition, the results received from neighbors are also combined with the local detection result. Again, there is no sufficient details about the method of exchanging and combining the results. Simulation-based evaluation shows that combining the results of the detection engines at the different layers leads to an increase in the detection accuracy, up to 20%. However, deploying all these detection engines increases largely the processing overhead. In addition, exchanging the detection results between the neighbors constitutes a novel vulnerability that can be exploited to launch a *blackmail* attack.

A modified Markov chain-based IDS is proposed in [91] so as to detect the attacks that disrupt the routes established by the DSR protocol. A malicious activity is detected when there is an important change in the routing table. More precisely, two routing features are monitored so as to detect possible abnormal values. The first is the percentage of changes in the number of routes (PCR) that represents the added/deleted routes within a certain time window. The second is the percentage of changes in the number of hops (PCH) which indicates the change in the sum of hops of all the routes within a given time window. Based on the previous w consecutive values of the feature (i.e., PCR or PCH), also called the $from\_state$, the next value of this feature, called *to_state*, can be predicted. If there is a large difference between the actual and the predicted *to_state*, an attack is detected. In practice, the Vector Quantization (VQ)[92], a lossy data compression method based on the principle of block coding, is used to categorize the values of a feature into symbols. The feature value whose probability is under a certain threshold is categorized as a "rare" symbol. Thus, the noise and unnecessary states during the construction of the Markov chain are avoided. A Markov chain is constructed such that the $from\_state$ represents the previous w ordered symbols of a categorized feature $X_i, X_{i+1}, \cdots, X_{i+w-1}$. Whereas *to_state* represents the next value X_{i+w} (w is a parameter that characterizes the Markov chain). By sliding a window of size w through the routing traces, the probability of the transition from a state s_1 to a state s_2 is calculated as $P(s_1, s_2) = N(s_1, s_2)/N(s_1)$. Note that $N(s_1, s_2)$ is the number of times where the transition from s_1, which refers to $from\_state$, to s_2, which refers to *to_state*, has taken place. While $N(s_1)$ is the number

of times when s_1 represents the $from\_state$, i.e., s_1 is the initial state of a transition. If the difference in the probability of a transition (s_1, s_2) between training and operation phases exceeds a preset threshold, attack occurrence is confirmed and neighbors are alerted. Simulation-based evaluation shows that the classifier constructed using PCH performs better than the classifier constructed using PCR. It also proves that this IDS is able to detect more than 90% of routing disruption attacks when the nodes' mobility is relatively low. However, the detection accuracy and the rate of false alarms are negatively affected by the mobility. This results from two reasons. First, in a high mobility scenario, a node can notice only few changes in the routing table before changing its location. Second, the changes in the routing table are rapid and inconsistent when the nodes are highly mobile. Furthermore, the exchange of alerts between the neighbors constitutes a new vulnerability that can be exploited by a malicious node so as to launch *blackmail* attack.

This previous Markov chain-based IDS has been enhanced later on [93] by taking into account the mobility impact and by adjusting the detection correspondingly. Since the speed of the nodes does not reflect MANET dynamics accurately, Link Change Rate (LCR) is used as a unified metric that captures the common features of different mobility models. LCR represents the number of changed neighbors within a time interval. For each mobility model, the corresponding LCR is computed as the average of nodes LCRs. During the training phase, a normal profile is built for each LCR. This profile includes a detection threshold and a probability transition matrix for the Markov chain. Here, the Markov chain is built only for one routing feature: the percentage of changes in the number of hops (PCH). During operation phase, link change information is collected and averaged periodically so as to have the corresponding LCR. Then, the normal profile whose LCR is the closest to the calculated one is selected to be used during the next time interval. Any difference between the expected and the actual probability of a transition in the Markov chain, which exceeds the threshold defined in the selected normal profile, is confirmed to be an attack. Simulation-based evaluation proves that the enhanced IDS has a high detection rate similarly to the original IDS defined in [91]. In addition, there is a significant reduction of the negative impact of the mobility on the detection rate and the rate of false alarms. However, a malicious node may launch an attack without being detected in the presence of high mobility [23].

The IDS proposed in [94] uses a cost-sensitive classification that aims at minimizing the expected cost, in terms of network connectivity, rather than the probability of misclassification. This association between the classification and the cost is motivated by the fact that raising a false alarm has a significantly lower cost than allowing an undetected intrusion. Predefined fixed costs of, both, false positive, i.e., classifying a normal behavior as abnormal, and false negative, i.e., classifying an abnormal behavior as normal, are specified for 4 types of attacks: flooding network, interrupting the active routes, dropping route error notification, and *blackhole* attacks. The detection rate and the false positive rate with and without cost sensitive classification are compared for the following four classifiers: MultiLayer Perceptron (MLP)[95], artificial neural network classifier, Naïve Bayes [89], a probabilistic classifier, Linear classifier [96] that applies linear combination of feature values, and Gaussian Mixture Model (GMM) [97], a probabilistic classifier. Another contribution of this work includes the utilization of cross-validation [98], an unbiased hyper-parameter selection method that trains the classifier relying on a sequential partition rather than a random one. In practice, the training data is sequentially divided into n

parts so as to quantify the relation existing between the selected features. At the i-th iteration, the part i is used to validate the performance of the classifier while the remaining parts are used in the training. At the end, the obtained measurements are averaged. With this method, the classification features are the number of received/forwarded route requests/replies, the number of received/forwarded route error, the number of received/forwarded data packets, the number of neighbors, and the percentage of changes in route entries and hop counts in AODV-based MANET. Broadly speaking, experiments show that including cost dimension in the classification, surprisingly, minimizes not only the expected cost but also it decreases the misclassification for most classifiers. However, the proposed sequential cross-validation method offers a modest improvement on the accuracy of classification in some cases. Furthermore, this IDS considers that a false alarm is significantly less costly than allowing an undetected intrusion. But, this consideration is not proved in all the cases [57].

The work proposed in [99] distinguishes itself from aforementioned IDSs by: (i) considering the power consumption as a classification feature, and (ii) comparing the operational power consumption with a set of power consumption profiles induced by known attacks rather than attack-free profile. The basic idea in this standalone IDS is to monitor power consumption in every node's battery. This consumption is further compared with the predefined power consumption patterns induced by known attacks, using smart battery technology. In an experimental implementation, this IDS was able to detect up to 99% of the attacks, provided that no more than one type of attacks takes place at the same time. It also detected multiple attacks, but only when other activities were absent, i.e., the node is idle. Since this IDS monitors the local hardware operation, it overrides the problem of manipulating the audit data/evidences. As a result, it is more reliable than other IDSs. However, it detects only attacks which cause irregular power consumption and only when the nodes are idle, something that rarely occurs in MANET.

An energy-aware, neural network-based detection approach [100] searches for abnormal internal links delays so as to detect *wormhole* and *Byzantine* attacks targeting AODV protocol. In practice, one node is periodically elected, according to an energy-related criteria, so as to be a root in MANET. The root identifies a spatial-time logical network topology model and estimates the corresponding link delay distribution. The link delay distribution is estimated with Network Tomography (NT) [101], a limited-overhead technique for inferring information about the network internal link performance based on end-to-end measurements. If the inferred link delay distribution deviates from the expected distribution, which is defined in a training phase, then an attack is detected.

To detect such deviation, Self-organizing Map (SOM) [102], a neural network-based classifier, is used. Simulation-based evaluation shows that the proposed IDS has a high detection rate with a limited number of false positive. However, mobility affects negatively the detection accuracy. Moreover, the computation overhead increases along with the network size and traffic throughput. Since link performance data is exchanged between the nodes, a malicious node can exploit these communications and launch a *blackmail* attack.

Social network analysis methods are employed to detect dropping, *blackhole*, sleep deprivation[7], and TCP SYN flooding attacks in MANET [71]. In practice, every node collects the control

[7] The attacker tells other nodes that the victim node has the best routes towards all the destinations and hence, all the traffic is oriented towards the victim. Consequently, the resources of the victim are exhausted.

and data traffic from its ego network. An *ego* network is made of the node itself (*ego*) together with the nodes which are connected to it (so-called *alters*) and all the links among those *alters*. The detection engine then searches for abnormal values for social related metrics such as centrality measurements [103]. These measurements reflect the relative importance, in terms of packet sending/relaying or packets overhearing, of every *alter* in an *ego* network. Similar to aforementioned works, abnormality is detected thanks to some normal profiles which are built during an attack-free training phase. Nevertheless, the required training phase is relatively long. This social-based detection engine creates less computational complexity compared to conventional classification engines. However, in a high mobility MANET, a node has a limited time to create social relations with neighbors. Hence there is maybe not enough information to realize social analysis. Consequently, the detection accuracy decreases. Furthermore, since a malicious node may either transmit false audit data or avoid transmitting any of them, the detection process may be hindered or mis-leaded.

A summary of the aforementioned classification-based IDSs is presented in Table (2.1). Except [99], these IDSs identify a malicious action any deviation between the predicted and the actual value of selected feature(s). In general, their performance depends on:

- the selection of a proper set of features such that their values clearly differentiate when an attack takes place,
- the selection of a proper classifier that categorizes accurately the observed/calculated values of a feature into the defined classes,
- training the classifier over a wide range of scenarios involving e.g., a varying mobility/density, as well as different types of attacks.

Although, most of the classification features are related to routing (e.g., percentage of changes in route entries), other types of features (e.g., power consumption, link delay, and importance) have been used. The classifiers are diverse: they are based on statistical, probability, rule, neural network, data mining or even hardware techniques. Note that, C4.5 and SVM, outperforms other classifiers. Regarding the strengths of the analyzed IDSs, we can infer that: (i) the majority exchange and/or aggregate detection results so as to increase detection accuracy, (ii) some of theses IDSs attempt to employ multiple detection stages/engines so as to enhance detection accuracy and detect a wide range of attacks, and (iii) some of them try to minimize the processing and communication overheads through organizing the nodes into clusters or "ego" networks. On the other hand, we can infer that: (i) in most of the studied IDSs, mobility negatively impacts the detection accuracy, (ii) almost all of them are vulnerable to *blackmail* attack, (iii) the majority of them generate extra processing and communication overheads especially in the presence of mobility, and (iv) some of them have points of failure, e.g., root nodes or cluster-heads, that may spoil detection process.
It is worth to mention that a large portion of the IDSs in MANET are classification-based since they are proposed to detect a wide range of attacks. However, other approaches, e.g., trust-based IDSs, are proposed to override some drawbacks of the classification approach.

Table 2.1: Classification-based IDSs for MANET

[79]	SVM RIPPER	Routing	Route falsifications	- The cooperative detection enhances the local detection accuracy. - Vulnerable to *blackmail* attack.
[81]	1-SVM	Routing Traffic	*Blackhole* Network flooding	- Cluster-heads in a hierarchical architecture are charged of detection. - High detection accuracy since a large portion of network activities is monitored. - The cluster-head is a failure point and vulnerable to a *blackmaile* attack.
[83]	Trivial difference	Routing	*Blackhole*	- Training data is updated at regular time intervals. - Dynamic training method may constitutes a failure point.
[73]	MDS CCDS	Application	Injecting malformed instructions	- A two-stage detection method offers a high detection accuracy along with limited memory usage. - Vulnerable to false detection decisions due to the fixed threshold of calibration.
[85]	C4.5 CFA	Routing Traffic	*Blackhole* *Grayhole*	- C4.5 outperforms, in terms of abnormality recognizing, other classifiers. - The size of sampling window affects detection accuracy.
[87]	C4.5 CFA	Routing	*Blackhole*	- Hierarchical combining of anomaly indexes enhances detection accuracy. - Vulnerable to *blackmail* attack.
[88]	CFA Naïve Bayes Markov chains	Mac Routing Application		- Multi-layer detection enables detecting more kinds of attacks. - High consumption of resources. - Vulnerable to *blackmail* attack.
[91]	Markov chains	Routing	Route Disruption	- High detection accuracy only with moderate mobility. - Vulnerable to *blackmail* attack.
[93]	Markov chains	Routing	Route Disruption	- Link Change Rate is used as a unified metric of mobility. - Reducing the negative affects of mobility on the detection accuracy.
[94]	Naïve Bayes	Routing	*Blackhole* Forging packet Flooding network	- Including the cost dimension in the classification increases network connectivity and decrease the misclassification. - Sequential cross-validation method offers a modest improvement on the detection accuracy.
[99]	Smart battery	Power	Flooding network	- Comparing power consumption with the consumption profiles of known attacks. - Invulnerable to manipulated evidences. - A node should be idle so as to detect the attacks who cause irregular power consumption.
[100]	SOM	Link delay	Byzantine *Wormhole*	- The used neural network detects abnormal link delay with a high detection accuracy. - Mobility and network size affects negatively the detection accuracy. - Vulnerable to *blackmail* attack.
[71]	Social analysis	Importance	Dropping packets Sleep deprivation Network Flooding	- Incurring less computational complexity compared to conventional classification engines. - Mobility affects negatively detection accuracy. - Vulnerable to *blackmail* attack.

2.3.2 Trust-based Detection

Trust-based detection attempts to define to which extend a node that cooperates and/or provides evidences should be trusted. Based on the answer, the node takes a decision about accepting or refusing to cooperate with others. Meanwhile, answering this question requires to have a trust system that entails two main activities: the establishment of trust relationships and the dynamic update of these existing relationships. In general, each node monitors and examines the behavior of the other nodes so as to establish trust relationships. In addition, the reputation of a node (i.e., the opinion that other nodes have about it) may also be an important factor to determine whether this node is trustful or not.

Among the first trust-based IDSs in MANET Watchdog and Pathrater [72], that works on top of DSR protocol, are of prime interest. Watchdog aims at detecting misbehaving nodes that drop the packets while Pathrater combines the knowledge of misbehaving nodes with link reliability data to choose the most reliable route. In practice, when a node forwards a packet to a neighbor, it keeps a copy of the forwarded packet in a buffer and promiscuously listens to the transmission of this neighbor. If this latter forwards, in a certain time, a packet that matches to the one stored in the buffer (i.e., the neighbor correctly forwards the packet), then the node removes the packet from the buffer. Otherwise, Watchdog increases the failure counter assigned to this neighbor. If the failure counter exceeds a certain threshold, the neighbor node is confirmed as misbehaving. Pathrater is installed on each node in order to rate all the other nodes. More precisely, it increases (resp. decreases) the rating of a node each time this latter successfully forwards (resp. drops) a packet. When it is required, the metric of a route is calculated by averaging the ratings of the nodes in this route. Since Pathrater assigns a high negative rating for the misbehaving nodes detected by Watchdog, a low reliability metric is assigned to the routes that contain misbehaving node(s). Consequently, they are not selected for packets forwarding. If the Pathrater does not find any route free of misbehaving nodes to the destination, it will send a new route request. Simulation-based evaluation shows that using Watchdog and Pathrater on DSR enhances the throughput of data packets. However, they increase the DSR overhead up to 31% in some simulation scenarios. In addition, Watchdog is not able to detect packet dropping in the presence of collisions, limited transmission power, and selective (partial) dropping.

CONFIDENT [104] is a trust-based extension of the DSR protocol that aims at detecting dropping attack. It distinguishes itself from Watchdog and Pathrater by: (i) exchanging alarm messages, and (ii) punishing the misbehaving nodes. In practice, CONFIDENT consists of 4 components: a monitor, a reputation system, a trust manager, and a path manager. Monitor promiscuously listens to neighbors traffic and reports the malicious behaviors (e.g., dropping packets) to the reputation system. This latter maintains a rating table that registers the number of malicious behaviors detected for every other nodes. In practice, upon receiving a report from the monitor, the reputation system increases the rating of the reported nodes. If the rating of a node exceeds a certain value, the reputation system classifies this latter as misbehaving and passes this information to the path manager. Consequently, the path manager punishes the misbehaving node by removing the routes that contains this latter. Furthermore, it refuses to forward the packets generated by the misbehaving nodes. In addition, the trust manager sends an alarm message to report the misbehaving nodes that have been detected. Upon receiving this alarm message, the monitor checks whether the source of this alarm is distrustful, and hence it

discards the alarm. Otherwise, it will be stored in a table (so-called alarm table). When several alarms with sufficient level of reliability report a node, then the reputation system is notified so as to add the reported node as a misbehaving one. Fortifying Dsr with Confident reduces up to half the rate of dropped packets due to the misbehaving nodes. However, the punishment method proposed by Confident may be exploited to launch a DoS attack against a legitimate node. This can be done when colluding nodes cooperate so as to report on a legitimate node, and consequently this latter would be fired from the network.

Similarly, Core [105] proposes a trust-based extension working on Dsr and aims at detecting and isolating misbehaving nodes. However, Core distinguishes itself in three points. First, the estimation of a node's trust value is based on the direct observations but also on the gathering of the opinions of others. To that end, a node interrogates its neighbors about their lists of trusted nodes. By exchanging only the positive reports, Core avoids to be vulnerable to a DoS attack that results from false accusations against a legitimate node. Second, more importance is given to the past observations. Hence, sporadic misbehavior, resulting from a low battery or a topology change, in recent observations leads to a minimal influence on the trust calculation. Third, trust value is functional, i.e., a trust value is attributed for each function (e.g., route discovery, packet forwarding). Thus, the global trust value assigned to a node is a combination of its functional trust values. Finally, when a node receives a request to cooperate, e.g., forwarding a packet, the request is rejected if its source has a negative trust value. Even though Core solves the problem of false accusations, it is vulnerable to false praises that enable the colluding nodes to enhance their reputations. Moreover, giving more importance for past observations makes Core vulnerable to the intoxication that takes place when: (i) a legitimate node is compromised and starts misbehaving, or (ii) a misbehaving node tries to gain the trust of others by correctly cooperating for a moment before it starts misbehaving.

In Ocean [106], a node relies only on its direct observations during trust establishment, and thus the false accusations and/or praises are avoided. Ocean aims at detecting misleading and selfish nodes in Dsr protocol. A misleading node is defined as the one that participates in route discovery operation but does not forward the packets for others. The selfish node is the one that does not participate neither in route discovery nor in forwarding packets for others. In order to detect misleading nodes, Ocean follows a similar mechanisms to the one used in Confident. More precisely, when forwarding a packet to a neighbor, the node buffers the packet checksum. If the neighbors does not attempt to forward this packet within a certain time, the node decreases the rating of this neighbor. Otherwise, the rating of this latter is increased. In both cases, the checksum is removed from the buffer. Once the rating of a node falls below a certain threshold, it is added to the so-called faulty list. A route is rated bad if the next hop in this route belongs to the faulty list. A node that broadcasts/re-broadcasts a Route Request Message (Rreq) on Dsr appends its faulty list to this message so as to specify the nodes to be avoided in the future routes. Upon receiving a packet from a misleading node, the node can reject this packet. In order to detect the selfish nodes, Ocean proposes that each node maintains a counter for every other nodes. The counter assigned to a node is increased every time this latter forwards a packet and vice versa. When the counter of a node falls below a certain threshold, then it will be classified as selfish and thus, its packets will not be forwarded. Compared to the approaches where alarm (or positive rating) messages are exchanged between the nodes, Ocean operates well in term

of network throughput. However, it does not punish the misbehaving and/or selfish nodes. In addition, when a node refuses to forward packets originated by a misleading/selfish node, neighbors of this node may consider this action as misbehaving since nodes do not exchange alarm messages.

The IDS proposed in [107] aims at detecting *blackhole* attack targeting the AODV protocol. In practice, several nodes are elected to work as monitors. They sniff the AODV route request (RREQ) and route replay (RREP). It is supposed that the monitors are uncompromisable and cover all the nodes in the network. A monitor node maintains information about every route request/reply flow, i.e., source, destination, sequence number and the intermediate nodes that have broadcasted the related RREQ messages. If a node sends a RREP message without being neither the destination nor one of the intermediates that have participated in broadcasting the corresponding RREQ, then the suspicion value assigned to this node is increased. When the suspicion value of a node exceeds a certain threshold, it will be confirmed that this node is an attacker and a blocking message will be broadcasted so as to ask other nodes to cooperatively isolate the attacker. Upon receiving a blocking message, a node increases the suspicion value of the node(s) advertised in this message. Simulation-based evaluations prove that this simple detection mechanism is able to greatly reduce the number of dropped packets due to *blackhole* attack, especially when there is a sufficient amount of monitors. Moreover, it offers a high detection rate along with a limited number of false alarms. However, it is oriented towards a very specific implementation of *blackhole* attack and does not take into account other scenarios (e.g., increasing maliciously the sequence number of a RREP message before forwarding it). In addition, a malicious node may launch a *blackmail* attack by tampering or forging blocking messages.

The IDS proposed in [108] aims at detecting *blackhole* and *gray-hole*[8] attacks targeting the OLSR protocol. Indeed, a source validates the transmission path towards a destination by sending periodically a validation message, called PVM, to the destination. If this latter acknowledges this message, then all the nodes composing the path are considered as well-behaving. Otherwise, the number of failed validations is increased. If the number of failed validations exceeds a certain value, then the source of the corresponding path activates an attack search which consists in sending an Attacker Message Finder (AMF) that should be acknowledged by each node along the path. The acknowledgment of AMF includes the hop count and the next node towards the destination. Thus, the source will be able to determine at which intermediate along the path the packets were dropped. Consequently, the misbehaving intermediate is added to a blacklist. For the case where all the intermediates have acknowledged the AMF messages, i.e., the attacker drops only the data packets (PVM), the source sends successively a PVM message to every intermediate. If one of these intermediates failed to acknowledge the corresponding PVM message within a specific time, it will be considered as an attacker and added to the blacklist. The attacker is eliminated from the routing table as long as it exists in the blacklist. Furthermore, its packets will be rejected. A node exchanges with its neighbors information about a detected attacker by sending an Attack Information Message (AIM) with initial rate equals to 1. Upon receiving an AIM message, if the rating of this message is more than a certain threshold then, the received node adds, if not existing, the mentioned attacker into its blacklist and forwards

[8] This attack refers to packets selectively dropping.

this AIM messages to neighbors. Otherwise, the received node verifies whether the mentioned attacker is really a misbehaving node by sending a PVM message to the attacker. If there is no returned acknowledge within a specific time, the node adds the attacker into its blacklist and relays the AIM messages after increasing its rating. Otherwise, the mentioned attacker will be deleted from the blacklist and the AIM message is relayed with a rating equal to -1. The evaluation of this IDS shows that the overhead generated due to the detection-related messages constitutes, at maximum, 12% of OLSR traffic. It is also worth to mention that this IDS is vulnerable to *blackmail* attack where a malicious node can accuse a legitimate node through an AIM message with a high rating. Thus, the accused node is discarded as its packets are rejected by every node that has received the false AIM message.

The voting-based IDS proposed in [109] aims at detecting dropped or maliciously modified packets by setting traps for the attackers. In practice, the nodes are grouped into either *cliques* or *clusters*. In a clique, each pair of nodes are within the radio range of each other. While in the cluster, there is, at minimum, one node that has all other members of the cluster in its radio range. In each cluster or clique, a monitoring node is elected using various schemes. It is assumed that a monitor can never be malicious, and is rotated periodically in order to prevent unfair use of the resources and battery depletion. When a monitor receives a suspicious or modified message from a node in its cluster/clique, it activates the detection process. Indeed, the monitor broadcasts (resp. unicasts) a monitoring message to all the nodes in its cluster (resp. clique). It is worth to mention that the monitoring message should look like any regular message, so that no node will suspect it. Upon receiving the monitoring message, a node should forward this message to its cluster/clique members. If any of the cluster/clique member receives a modified monitoring message (or no message at all), it marks the corresponding node that transmitted the modified message (or did not transmit anything) as suspicious. After that, the monitor initiates a voting and asks its cluster/clique members to notify which nodes they believe being suspicious. If the votes against a node exceeds a certain threshold, then the monitor confirms this latter as malicious. The performance of this detection scheme is evaluated in a simulated MANET working with the AODV routing protocol. Since there is no use for intensive operation and the only traffic exchanged is monitoring and voting messages, low processing and communication overhead is generated. However, packet loss, due to congestions or nodes mobility, increases substantially the ratio of false alarms. Besides, in case of an attack against the monitor or monitor failure, the detection process will be disabled. Furthermore, the voting process is vulnerable to *blackmail* attack that results in accusing (resp. praising) a legitimate (resp. a colluding) node.

[75] proposes a friend-assisted, hybrid (i.e., anomaly and misuse), and two-tier (i.e., local and global) IDS wherein real world friendships are used to filter collected evidences. The first tier employs only locally collected audit data and evidences in a signature-based detection engine. If suspicious activity is detected without being able to accurately determine a specific attack, an anomaly detection engine (also located in the first tier) is activated. However, if both engines in the first tier are not able to confirm whether the detected suspicion is an attack or not, the second tier which is a collaborative friend detection mechanism will be triggered. In the second tier, a node requests its neighbors' opinions regarding the detected suspicion. An interrogated neighbor uses its proper local audit data and evidences to determine how it analyzes the suspicious activity, i.e., malicious, good, or neutral. Having collected the opinions, the node takes the decision and

notifies the participating nodes about the voting result. In order to protect the voting operation from colluding *blackmail* attackers which are malicious nodes that send false accusations and/or false praises, only the opinions of trustful nodes are considered. The trust relationships between the nodes are initially based on the friendship of bearer in the real world. Later, these direct trust relationships are exchanged between friends to create a new set of trust friendships. Simulation-based evaluation, wherein dense MANETs (e.g., university campus and city) are considered, proves that when a friendships-based voting is used in place of a general voting, detection is less susceptible to *blackmail* attack. However, network density, number of initial friendships, and the age of the network are 3 factors that can largely impact the performance of this IDS. For instance, the rate of false alarms and the detection accuracy are negatively affected by the lack of friendships among the nodes. Since the aim of this work is to present the role of friendships in improving the global detection, no details about the used anomaly and misuse detection are offered.

A summary of the aforementioned trust-based IDSs is presented in Table (2.2). They all depend on monitoring the nodes' behavior so as to build trust relationships in the network. In practice, a rating is assigned to every node, and is increased (resp. decreased) each time the corresponding node well behaves (resp. misbehaves). In addition to direct monitoring, some of them employ other indicators during trust estimation such as the real world friendships and the negative/positive ratings coming from the network. Regarding the strengths of the analyzed IDSs, we can infer that: (i) the majority cause a low processing overhead since they use simple algorithms during trust estimation, (ii) all of those IDSs do not require a training phase or predefinition of the attacks, and (iii) mobility has limited effects on the majority of those IDSs. On the other hand, regarding the weaknesses, we can infer that: (i) the majority addresses only dropping attack, (ii) almost all of those IDSs are vulnerable to *blackmail* attack during voting or alert exchange operations, (iii) some of those IDSs lead to non-trivial increase in traffic overhead, and (iv) network density and/or initial trust relationships affect the detection accuracy in some of those IDSs.

2.3.3 Automata-based Detection

Finite state machine (FSM) or automata is used to model the correct behavior of nodes supporting *ad hoc* routing protocols. A non-expected transition between two states refers to a violation of the routing protocol specification that may occur due to an attack.
In [1], a detection mechanism based on finite state machine is proposed for detecting link spoofing, man-in-the-middle, and deny of service attacks targeting OLSR protocol. Indeed, the normal processing of OLSR control messages is modeled in a FSM (Figure 2.1): When a node receives a hello message it updates its neighbor and MPR sets. Upon receiving a TC message, the node (i) updates the topology and routing table and (ii) forwards the received TC message if the node is selected as a MPR. In addition, the node (resp. the MPR) should periodically broadcast hello (resp. TC) message. This FSM is further used to extract the following constraints:

1. The neighboring relation must be reciprocal (e.g., if A's hello messages lists B as a neighbor, then B's hello messages must list A as a neighbor).

2. The MPRs of a node A should reach all the 2-hop neighbors of this latter.

Table 2.2: Trust-based IDSs for MANET

[72]	Watchdog Pathrater	Local	Dropping packets	- Enhancing data traffic throughput. - Increasing the traffic overhead. - Inefficient against the selective dropping or in the presence of collisions.
[104]	Watchdog Path manager Exchange of misbehaving list	Local Global	Misbehaving	- Reducing the rate of dropped packets due to misbehaving nodes. - Isolating misbehaving nodes. - Vulnerable to false accusations.
[105]	Monitoring Exchange of trustful list	Local Global	Misbehaving	- Functional estimation of trust so that past evidences are the most relevances. - Invulnerable to false accusations. - Isolating misbehaving nodes. - Vulnerable to false praises.
[106]	Watchdog Route Ranker	Local	Misleading Selfishness	- Invulnerable to *blackmail* attack since it uses only local observations. - Vulnerable to false positive since no alarm message is exchanged.
[107]	Watchdog Exchange of misbehaving list	Local Global	*Blackhole*	- Providing a high detection accuracy. - Reducing significantly packet loss rates. - Vulnerable to *blackmail* attack.
[108]	Verification messages Exchange of misbehaving list	Local Global	*Blackhole* *Grayhole*	- Imposing limited resources consumption. - Isolating misbehaving nodes. - Vulnerable to *blackmail* attack.
[109]	Monitoring Voting	Local Global	Dropping packets Tampering packets	- Hierarchical combining of anomaly indexes enhances detection accuracy. - Voting mechanism is vulnerable to *blackmail* attack.
[75]	Real world friendships Voting	Local Global	Route falsifications	- Two-tires, hybrid detection increases detection accuracy. - Invulnerable to *blackmail* attack since voting is restricted within friend nodes. - Limited initial friendships impacts negatively detection accuracy.

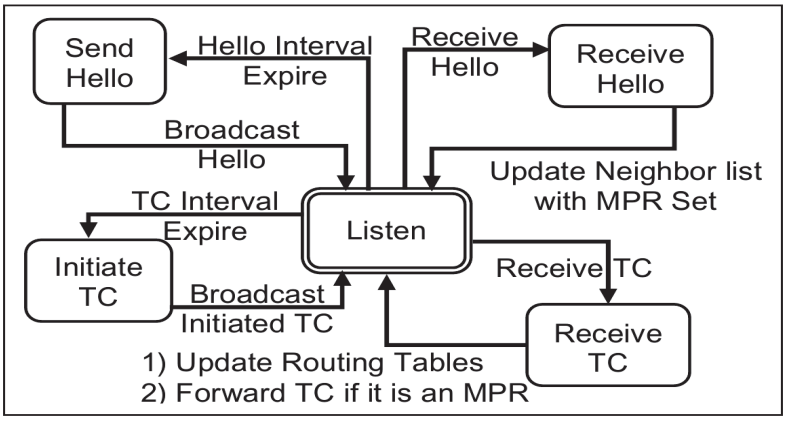


Figure 2.1: OLSR routing FSM [1]

3. If A finds itself advertised as a MPR selector in a TC message generated by B, then A should have been selected B as a MPR.

4. A MPR must transmit periodically TC messages and forward others' TC messages without modifications.

A centralized detector analyses the hello and the TC messages that are generated or forwarded by each node so as to detect the violations of these properties. A violation should last over a period of time[9] before considering it as an attack. This waiting is necessary to prevent false alarms resulting from temporary violations that arrive when, for example, network topology changes. Simulation-based evaluation shows that this IDS is able to detect all violations of the aforementioned constraints. Moreover, ignoring temporary violations reduces largely the rate of false positive in some scenarios of mobility and traffic. However, the centralized detection constitutes one failure point. Moreover, forwarding routing traffic to be processed in a central point may expose it to falsifications along with increasing the traffic overhead.

Similar approach is proposed in [2] so as to detect malicious modifications of AODV routing messages. Herein, distributed sensors are used to sniff and group the routing messages (i.e., route request (RREQ) and route replay (RREP) messages) per request-reply flow so as to predict the forwarding path. This prediction is done with a FSM (Figure 2.2) that models route discovery flow: If the source needs to find a route towards the destination, it broadcasts a RREQ message (i.e., a transition from Source state to RREQ Forwarding state takes place) with a new sequence number. When a RREP message is detected then there is a transition from RREQ Forwarding state to RREP Forwarding state. Upon the arrival of the RREP message to the source, the route is set up and the flow returns to the first state. If the sequence number and/or the hop counts are modified improperly, then the flow goes to suspicious state and an alarm is triggered. Besides, an attacker may drop or improperly forward the RREQ/RREP messages. It may also improperly modify the message header fields. Therefore, sensors keep track of the routing packets that are related to the same request-replay flow. As RREP message is a unicast message, a sensor can follow it easily by looking to its source and destination IP address. But this is not the

[9]This period is related to the intervals of hello and TC messages used during OLSR implementation.

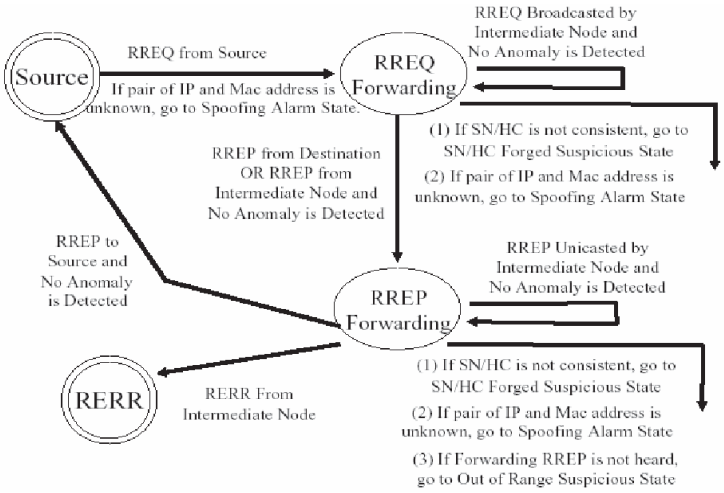


Figure 2.2: AODV route discovery FSM [2]

case for the RREQ message, which is a broadcast message, i.e., its destination address is always (255.255.255.255). Therefore, a new attribute, called Previous Node (PN), is added to the RREQ header so as to indicate the node that previously forwarded it. Hence a sensor is able to identify the node that drops or forwards a route request/reply *via* a non expected path. Unfortunately, performance evaluation is not provided. However, the previous FSM is used recently in [110] in order to detect *blackhole* and packet dropping attacks targeting AODV. Here, simulation-based evaluation shows that up to 93% of attacks are detected in some scenarios. Moreover, there is a significant enhancement in the packet delivery ratio, but, colluding nodes affects badly the detection accuracy. In addition, if a malicious node is not included in any request/reply flow, then there is no possibility to be detected.

[3] introduces an intrusion detection approach that is based on FSM and aims at detecting route request/reply spoofing, dropping packets, impersonating identity, packets fabrication on the DSR protocol. Indeed, a subset of nodes is randomly elected to act as monitors such that each monitor observes the traffic of its neighbors. These monitors, which are periodically re-elected, may exchange observation-related information so as to handle the case when a monitored node moves from one monitor zone to another. To detect the attacks, monitors employ FSMs that model, both, the proper and the improper forwarding of data packets or route requests/replies. A FSM, similar to the one defined for AODV in [2], is defined to model route discovery operations on DSR. Another FSM (Figure 2.3) is defined to model packet sending process and the corresponding attacks: malicious packet modification, impersonating, and fabrication attacks. More precisely, when a node sends a packet (i.e., the FSM moves from state 1 to state 2), the corresponding monitor checks first whether the packet is an originated one (i.e., state 6). In this case, the source address of this packet should be identical to the node which has sent the packet. If not, the monitor triggers an alarm and alerts an impersonating attack (state Alarm3). If the packet is not an originated but a forwarded one, i.e., the node address belongs to the list of the forwarding nodes, then the FSM goes to state 3. This means that the monitor does not previously see the packet, and therefore it needs to inquire neighbor monitors about this packet (state 4). If no other monitor confirms receiving the packet once, the corresponding monitor triggers an alarm

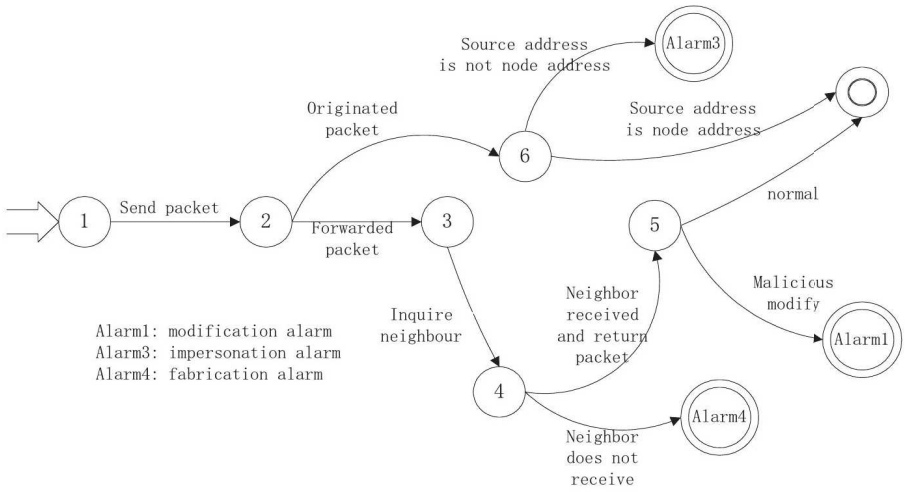


Figure 2.3: FSM for packet sending [3]

and alerts a fabrication attack (state Alarm4). Otherwise, other monitors) should return the packet to the corresponding monitor (state 5). This latter then checks whether some fields of the packet are maliciously modified. If it is the case, then the corresponding monitor triggers an alarm and reports a spoofing modification attack (state Alarm1). Simulation-based performance evaluation presents that this IDS offers high detection rate with a limited number of false alarms for most of the addressed attacks. Since the monitor is the only responsible for the detection in its zone, it may constitute a single failure point. Moreover, when a compromised monitor is inquired, it can foil the detection process by returning incorrect answers without being detected.

Since FSM may not be sufficient to manage all the complexity of *ad hoc* routing protocol, Extended Finite State Machine (EFSM) are used to specify the behavior of such protocols [76, 111]. An EFSM is similar to FSM except that its transitions and states are described by parameters and variables respectively. In addition, EFSM allows to put constraints on the variables of the transitions [111].

In [76], a hybrid detection that couples statistical classification-based method with an extended FSM-based method is proposed to detect attacks against the AODV protocol. In practice, AODV routing operations are modeled using an extended finite state machine (EFSM). Invalid state (e.g., a state with a negative hops count), incorrect transition, unexpected packet delivery, and unexpected state variable assignments are direct violations of the AODV specification, and thus they are considered as attacks. Since the attacks with a temporal and a statistical nature, e.g., flooding of data and/or routing traffic, can not be recognized as violations of the EFSM, a machine learning classification is used to distinguish normal and abnormal events. In this method, the classifier used is RIPPER [80] which is a rule-based classifier. The classification features employed are related to the frequencies of transitions in the EFSM. For instance, a flooding attack is detected when the frequencies of the transitions exceeds the expected frequencies that is estimated during a training phase. Simulation-based evaluation proves that this approach presents a high detection rate especially for the attacks related to direct violations of the AODV specification. However, it is not able to detect the attacks requiring a knowledge beyond a local node, e.g., falsification of the sequence number in a routing message.

The intrusion detection approach proposed in [111] employs also an EFSM so as to detect attacks against the OLSR protocol. In practice, an EFSM is defined to model the normal exchange of messages routing between two neighbors. Then, each node traces and maps the sent and received routing messages to compare those to the EFSM. Here, the comparison is done thanks to the Backward Checking algorithm [112]: a passive test checks in a backward fashion whether the final state of a flow of routing messages has been reached through one of the possible departure states according to the EFSM. In order to clarify the type of attack that causes the detected violation, a complementary signature-based IDS is suggested to be coupled with this approach. Analytical evaluations show that this approach is able to detect a violation of OLSR specification. However, it is unable to determine whether this violation is generated due to an error in the implementation or a security attack.

Instead of using FSM in modeling the correct scenarios of the routing operations that should be respected, some IDSs [4, 5] propose to use the FSM in order to model the attacks signatures. During the detection, these IDSs search whether there is a match between a sequence of gathered evidences and an attack signature, and hence the occurrence of an attack is confirmed.

AODVSTAT [4] is a stateful intrusion detection approach that aims at detecting dropping, identity spoofing, network flooding and message tampering attacks targeting the AODV protocol. In practice, few sensors sniff the traffic and match it against predefined FSMs. In such FSMs, the transitions between the states are annotated with actions that, if omitted from the execution of the corresponding attack scenario, would prevent this latter to be completely successful. For instance, figure (2.4) represents the FSM of the flooding attack that aims at depleting network resources: The number of packets received from a node is maintained within a specific time

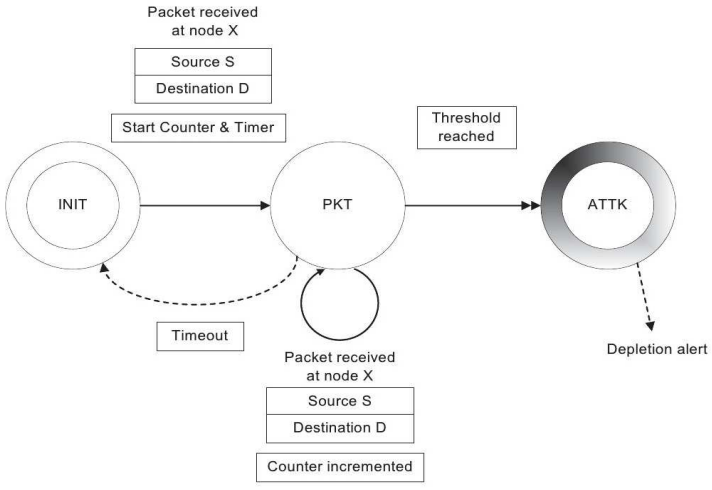


Figure 2.4: FSM for network flooding attack [4]

window. If this number exceeds a certain threshold, then the sender is considered as an attacker and a depletion alert is launched. Otherwise, the counter is reset. Similar FSMs are defined for dropping attacks, failing to replay a route request/reply, spoofing attacks (the falsification of IP or MAC address), tampering routing message, and increasing abnormally the sequence number of a route reply message. Note that this last attack requires that the sensors cooperate and exchange messages (so-called UPDATE messages) containing details of the adjacent nodes of each sensor.

Update messages contain a list of known IP/MAC pairs, the sequence numbers of adjacent nodes, and information about the detected attacks. Aodvstat is evaluated in, both testbed and simulation environments. It presents a high detection rate with a limited number of false alarms. It further triggers a limited increase in Cpu load and memory utilization. However, a malicious node may foil the detection through false accusations and/or forged Update messages. Moreover, Update messages increases the traffic overhead.

Fsm is also employed to model the signature of hello message fabrication in Olsr protocol in an agent-based Ids [5]. In practice, each node uses a Simple Network Management Protocol (Snmp) agent so as to collect audit data from the Management Information Base (Mib). After that, events are extracted from the collected audit data and are matched to the attack signature. The addressed attack aims to break the link between a victim node and its neighbors, and thus a DoS takes place. To that end, the attacker impersonates the identity of the victim and sends a faked hello message advertising one of the victim symmetric neighbor with lost link status. Upon receiving the fabricated message, the neighbor changes the status of its link with the victim to "heard" and stop routing packets through the victim. Figure (2.5) represents the signature of the aforementioned attack: The detecting node, which is in this case the victim's

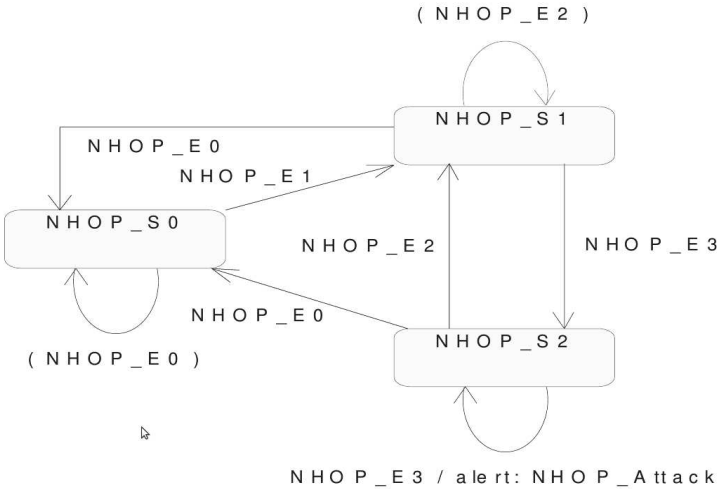


Figure 2.5: FSM for spoofed hello message attack [5]

neighbor, stays in the initial state ($NHOP\_S0$) as long as it has not received any advertisement from the victim with a "lost" status type. Upon receiving one of such advertisement (i.e., event $NHOP\_E1$ is extracted from the audit data), a transition to state $NHOP\_S1$ takes place. At state $NHOP\_S1$, if a "symmetric" or "Mpr" advertisement (event $NHOP\_E1$) is received, i.e., link type is changed twice within one hello message interval, a transition to state $NHOP\_S2$ takes place and an attack is confirmed. Otherwise, if only "lost" advertisements (i.e., event $NHOP\_E1$) are received, the behavior is considered normal. It is worth to mention that Stepping Stone, an attack at the application layer that aims to create a telnet connection chain until the victim, is similarly treated in this work. However, the detection of such attack can not be done locally, i.e., by one node, but it requires the cooperation between the nodes along the created chain. The use of Mib, as a source of events, enables gathering events about different types of attacks on several layers (e.g., network, system and application layers). Furthermore, it

provides a kind of abstraction/standardization of detection-related information, which facilitates the cooperation with other IDSs or security approaches. Performance evaluation was done in a real MANET composed of 7 non-mobile nodes and shows that the proposed IDS is able to detect the aforementioned attacks. Nevertheless, there is no detail about the detection accuracy or the generated overhead due to mobile agent exchange. However, these mobile agents could be compromised, and thus they constitute a novel vulnerability.

A summery of the aforementioned FSM-based IDSs is offered in Table (2.3). They depends on FSM to model either (i) some/all functions of the routing protocol, or (ii) the signatures of attacks to be detected. In the first case, an unexpected transition means that there is a violation of the protocol specification, and thus an attack is confirmed. While in the second case, reaching one of the final states means that all the required actions for the attack to be successful have been achieved. Hence, the attack is confirmed. In both cases, covering all details (resp. deviations) of the protocol (resp. the attack) by the defined FSM(s) is a critical condition of successful detection. Regarding the strengths of the analyzed IDSs, we can infer that: (i) the majority offer a high detection accuracy, (ii) some employs extended FSM since it has more capacity to cover the different details and conditions in *ad hoc* routing protocols, (iii) mobility has limited effects on the majority of those IDSs, (iv) some of those IDSs couples the FSM-based detection with other approach so as to address a wide range of attacks. On the other hand, regarding the weaknesses, we can infer that: (i) the majority address specific types of attack, (ii) most of those IDSs are vulnerable to *blackmail* attack. (ii) some of those IDSs lead to non-trivial increase in the traffic overhead, and (iii) some of those IDSs is unable, in some cases, to define the source or the type of the attack.

2.3.4 Rule-based Detection

In this approach, grammatical rules are defined to clarify how a routing function or a network service should be executed. Thus, a violation of one or more rules means that some nodes operates incorrectly due to a malicious intention or an error in the implementation. Another use of such rules is to define signs of inconsistency in the node messages. Detecting such inconsistency leads to the conclusion that an attack is occurring. In general, detection rules are related to packets forwarding, packets count, and route calculation.

In [6], a malicious node, which aims at falsifying routing messages, is detected using four rules that restrict the neighborhood relations and the MPR selection in the OLSR protocol. Indeed, upon receiving a hello or TC message, the node checks the satisfaction of the following rules:

1. Neighbor relation must be reciprocal (i.e., 2 neighbors must hear the hello message sent by each other).

2. A MPR node must be adjacent to its MPR selectors, i.e., the nodes that select this latter as a MPR.

3. A node that finds itself advertised as a MPR selector in a TC message must be adjacent to the originator of this message.

4. A TC message should be received without modifications by its originator.

Table 2.3: FSM-based IDSs for MANET

[1]	OLSR routing operation	Link spoofing Man-in-the-Middle DoS	- High rate of detection that is based on constraints on OLSR specifications. - Keeping an eye on temporary violations reduces the rate of false alarms. - Central detection constitutes one failure point and increases the traffic overhead. - Vulnerable to *blackmail* attack.
[2] [110]	AODV route discovery operation	Message Falsification dropping packets *Blackhole*	- Enhancing packet delivery ratio along with high detection rate. - Monitoring all the messages belonging to a route request/replay flow is a critical condition of the detection. - Vulnerable to *blackmail* attack.
[3]	DSR route discovery operation Received data packet processing	Message tampering Message fabrication Spoofing identity Dropping packets	- Offering high detection accuracy. - Monitors constitute failure points. - Vulnerable to *blackmail* attack.
[76]	EFSM for AODV routing operations	Message falsification Route fabrication Network flooding	- EFSM- and classification-based detection, addresses a wide range of attacks. - The absence of cooperation decreases detection accuracy in the cases where information beyond a local node is required.
[111]	EFSM for hello and TC messages exchange and processing in OLSR	Link spoofing	- Capacity to detect violations of OLSR specification. - Incapable of deciding whether an attack is the cause of an violation or not.
[4]	Identity spoofing attack Network flooding attack Dropping packet attack Routing message falsification attack	Identity spoofing Network flooding Dropping packets Message falsification	- Offering high detection accuracy along with limited increase in CPU load and memory usage. - Exchanging UPDATE message increases traffic overhead. - Vulnerable to *blackmail* attack.
[5]	DoS Stepping Stone attack	Message fabrication Telnet chain	- Offering a high level of standardization. - Capacity to detect the addressed attacks. - Detection-related mobile agent could be compromised and constitute new vulnerability.

If one or more rules are not respected then the routing table is modified to eliminate the routes containing the malicious node(s) that has caused this disrespect. The detection ability is evaluated with the link spoofing attack on hello messages over a network composed of 11 non-mobile nodes. In this method, the link spoofing implementation falsifies the hello message with links to both a non-adjacent and a fictive node. Thus, the attacker is guaranteed being selected as a MPR. Similar rules are defined in [113] to detect link spoofing attack on TC messages. In both works [6, 113], rules violation can be detected, but identifying the responsible is not always guaranteed. For instance, let a node X declares, in its TC message, B as a MPR selector, and B does not include X as an adjacent node in its hello message. When a node A receives, both X' s TC and B 's hello messages, A can not recognize who is the misbehaving node between X and B.

Instead of defining rules for a correct behavior that should be respected, others propose to define rules that, if are achieved, give indications about the occurrence of an attack. In [114], four rules, in a conditional form (If (cond.) Then), are used to detect replaying, forging and tampering routing message attacks targeting the AODV protocol. Each node collects and analyses RREQ and RREP messages that the node can overhear within its transmission range. To that end, each node maintains in a table (so-called extended history table (EHT)) information about the received or overheard RREQ/RREP messages (e.g., source IP, destination IP, sequence number, hop count, etc.). Upon receiving or overhearing a new routing message, the node searches for the inconsistencies between this message and the information maintained in its EHT. If an inconsistency is detected then an attack is reported. Otherwise, the information contained in this message will be added to the EHT. An inconsistency is confirmed when, at least, one of the following rules takes place:

1. The same RREQ or RREP message has been received earlier from the same node X. Thus, X is accused of a replay attack.

2. A node X sends a RREP even though this latter has not received the corresponding RREQ message. Thus, X is accused of a forging attack.

3. A node X modifies the source sequence number before re-broadcasting a RREQ message. A tampering attack is then reported.

4. A node X increases the destination sequence number before relaying a RREP message. X tampers the RREP message to be in the route between the source and the destination.[10].

According to simulation-based evaluations, the proposed IDS presents an acceptable detection rate against basic attacks. However, it was less efficient against more complex attacks, e.g., forging RREP messages with a high destination sequence number, especially when the number of nodes in the network is less than 60.

In [115], a link spoofing in OLSR routing messages is detected by mistrust reasonings. Indeed, intrinsic rules (similar to those used in [6, 113]) are derived from the OLSR specification. These rules describe the consistency that should be available between hello and TC messages. Inspired from these rules, the following mistrust reasonings are derived:

[10]Recall that the route with the highest destination sequence number is considered as the freshest route and hence, it is selected to relay the packets between the source and the destination.

1. If a node X declares in its TC message a links set that is not contained in the links set declared in the X's hello message, then X must be mistrusted.

2. If a node A finds itself as a MPR selector in a TC message of a non-neighbor or a non-MPR node X, then A mistrusts X.

3. If X_2 is selected as a MPR for X_1, then X_2 should forward the TC messages of X_1. Otherwise, both X_1 and X_2 are mistrusted.

4. If X_1 announces, in a hello message, X_2 as a neighbor while this latter does not include X_1 as a neighbor in its hello messages, then both X_1 and X_2 should be mistrusted.

5. If a node X, which is selected as a MPR, does not periodically broadcast a TC message declaring its MPR selectors, then X is mistrusted by its MPR selectors.

6. If a node X, which is selected as a MPR, does not forward the data and routing packets sent by its MPR selectors, then X is mistrusted by its MPR selectors.

7. If X_1 and X_2 have the same neighbors set, then they should have the same MPRs[11]. Otherwise, both X_1 and X_2 are mistrusted.

8. If X_1 and X_2 have the same neighbors set and a node A selects both of them as MPRs, then X_1 and X_2 should be mistrusted.

9. If the X_1's neighbors set is contained in the X_2's neighbor set, then X_1 must not be selected as a MPR. Otherwise, both X_1 and X_2 are mistrusted.

Each node continually searches for the inconsistencies in the received hello and TC messages by checking whether one (or more) of the aforementioned mistrust reasonings takes place. Simulation-based evaluation proves the ability of this detection approach to detect the existence of an inconsistency in others' routing message when an attacker sends falsified hello or TC messages. However, identifying the misbehaving node(s) is not always guaranteed.

The IDS defined in [74] proposes to have a hierarchical consolidation of the evidences gathered at different network layers. In practice, network is organized in clusters at several levels so that a cluster-head is elected according to topological-, security-, and resources-related criteria. Each node observes the activities of its neighbors in order to accumulate link-layer, network, and higher layers counts and statistics (e.g., number of received/forwarded packets, packets header and payload). The observations gathered are then successively aggregated by moving those up the hierarchy. Hence, summaries about each node behaviors are consolidated and correlated at the cluster-head of the top level in the hierarchy. The dropping packets attack is detected when the difference between the number of forwardable packets received by a node and the number of forwarded packets sent from this latter exceeds a specific threshold. MIM attack against the AODV routing protocol is illustrated as follows: an attacker increases the sequence number of a received route request RREQ message before forwarding it. Upon receiving the falsified RREQ message, the destination overrides the legitimate RREQ message that has a lower sequence number and thus, the attacker is inserted into the route between the source

[11] X_1 and X_2 may select different MPRs if the these MPRs have the same neighbors.

and the destination. Detecting this attack is done again by using the observations related to each node's received/forward RREQ messages, which are flowed upward in the hierarchy. The successive aggregation of routing information permits not only discovering the presence of a falsified RREQ message but also specifying the source of this falsification. In order to prevent excessive redundancy, the responsibility of the network and of the higher layers observation of each end-to-end flow is assigned to the first and last intermediate nodes in the flow. Moreover, several backups are maintained by the cluster-head at the highest level to avoid this latter to be a failure point. However, the detection accuracy and the overhead generated of this IDS is under question due to the absence of experiments. Also, a cluster-head, in lower levels, still constitutes a failure point that, if compromised, could mislead the detection process. Moreover, the hierarchical aggregation of observations is vulnerable to the *blackmail* attack.

In [116], Grammatical Evolution (GE) is explored to detect dropping, route request flooding, and route disruption attacks targeting the AODV protocol. Inspired from the natural evolution, GE is an evolutionary computation technique that aims at evolving programs written in a Backus-Naur Form (BNF) grammar. BNF grammar is a formal description of a language composed of several rules. In this work, GE is used make detection programs evolving in order to discover the ones that offer the highest detection accuracy. A detection program, i.e., a genome, is represented by a variable-length string so that each 8 bits refers to a rule from the BNF grammar (Table 2.3.4). The features used in the grammar are mobility-related (e.g., changes in the number of neighbors, number of recently added routes, etc.) and packets-related (e.g., number of route sending/receiving request/replay packets, number of non forwarded data packets, etc.). During the evolution, crossover and mutation are applied on the population, i.e., the current programs, to create new programs. These latters are then integrated into the next generation. In addition, the fitness for every novel program is calculated based on a training scenario. The fitness of a program reflects its detection accuracy, and is calculated as follows: *Fitness= detection rate - rate of false alarms*. After 2000 generations, the program with the highest fitness, i.e the highest detection accuracy, is chosen for each attack type. Simulation-based evaluation shows that the chosen evolved programs present a high detection rate even in high mobility scenarios. However, the chosen program for dropping attack generates a high rate of false alarms even in the scenarios without attack or mobility. In addition, the similarity between the training scenario, on which the evolution is based, and the operational scenario is a critical condition of a high detection accuracy.

A summary of the aforementioned rule-based IDSs is provided in Table (2.5). These IDSs employ grammatical or conditional detection-rules that are significantly oriented for attacks targeting *ad hoc* routing protocols. More precisely, these rules are built based on the specification of the routing protocol, usually extracted from the Request for Comments (RFCs). They aim at defining either constraints on the routing and packet forwarding functions to be respected, or signs of misbehaving nodes that violates the specification of the routing protocol. Regarding the strengths of the aforementioned IDSs, we can infer that: (i) all those IDSs have the capacity to detect a violation of the routing protocol specification, (ii) the majority are standalone IDS, i.e., detection process does not include cooperation, in terms of evidences or alarms exchanged between the nodes, and thus they are protected against the *blackmail* attack, (iii) some of those

Table 2.4: BNF grammar used for detection rules generation

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> ::= feature set

IDSs address a wide range of attacks which result in inconsistencies in the routing operations, (iv) some of those IDSs aggregate hierarchically gathered routing- and packet forwarding-related evidences, and thus the source of an inconsistency or a malicious action is identified accurately, and (v) some of those IDSs employ a genetic algorithm technique to enhance the rate of detection. On the other hand, we can infer the next weaknesses that: (i) some of those IDSs address only specific types of attack, (ii) the majority are not able to identify accurately the source of a malicious action or an inconsistency in the routing operations, (iii) almost all those IDSs evaluate the capacity of detection and ignore other important metrics such detection accuracy and the generated overhead, and (iv) some of those IDSs are vulnerable to *blackmail* attack.

2.4 Summary

In this chapter, we have provided a detailed overview about the Mobile *Ad hoc* NETwork (MANET). This network is composed of wireless mobile nodes that dynamically organize themselves in temporary and arbitrary topology. These nodes further communicate without the need of preexisting networking infrastructures. In MANET, a node communicates directly with all the other nodes within its wireless coverage range. When the destination is not in the coverage area of the source then the intermediate node(s) forward the packets from the source to the destination. Thus, each node in MANET operates as a router relying on a routing protocol and maintaining the related routing table.
Securing MANETs is particularly challenging because these networks rely on an open radio-based medium of communication. In addition, they are by nature cooperative, hence there is an absence of centralized management/security enforcement points e.g., switches and routers, from which preventive strategies can be launched. Thus, traditional ways of securing networks relying on e.g., firewalls and encryption technologies, should be enriched with reactive mechanisms, such as the Intrusion Detection Systems (IDSs) that constitute a second line of defense.
The IDSs that have been proposed for MANET vary significantly in terms of their architecture, detection mechanisms, and applied responding strategies. Most of them have a distributed and cooperative nature since it is the more suitable choice for running on top of MANET. However,

Table 2.5: Rule-based IDSs for MANET

[6, 113]	Neighborhood relations and MPR selection in OLSR	Link spoofing	- Capacity of detecting falsification in OLSR routing messages. - Identifying the source of a detect inconsistency is not always guaranteed.
[114]	Route discovery in AODV	Replay routing message Forging routing message Tampering routing message	- Providing accepted detection rate against elementary attacks. - Invulnerable to *blackmail* attack. - The absence of the cooperation between the nodes reduces the efficiency against the complex attacks.
[115]	Neighborhood relations and MPR selection in OLSR	Link spoofing	- Capacity of detecting inconsistency in OLSR routing messages. - Specifying the misbehaving node that causes an inconsistency is not guaranteed.
[74]	Attacks against AODV	Dropping packets Man-In-the-Middle	- Hierarchical aggregation of observations enables identifying the malicious node that has launched the detected attack. - Efficient against the complex attacks that require cooperative detection. - Vulnerable to *blackmail* attack.
[116]	Attacks against AODV	Dropping packets Route request flooding	- Grammatical Evolution of detection rules enhances significantly detection rate even in the presence of mobility. - Generating non-trivial number of false alarms. - Similarity between evolution scenario and operation scenario is critical for successful detection. - Triggering a non-trivial processing overhead.

several IDSs have either a hierarchical or a standalone architecture. We used the detection mechanisms to list and to classify the most well-known IDSs for MANET. Indeed, these mechanisms of detection fall under four categories:

- Classification-based detection: The basic idea in such detection mechanisms is to classify the values of a group of features, which are observed during a slot of time, into either normal or abnormal state. These features are selected because their values change significantly when an attack takes place. In practice, routing- and traffic-related features (e.g., percentage of changes in the number of advertised routes, number of route request/response messages, packet forwarding delay) are the most used features. However, other types of features are also used such as energy consumption and centrality measurements which is a social-related feature. The classifier(s) used is/(are) responsible for classifying a feature's value into one of the defined classes. There is a wide range of classifiers which are based on statistics, probabilities, rules, neural networks, data mining, or even hardware techniques. Regardless of its technique, a classifier needs to be trained so as to be able to classify a feature's value. The training takes usually place in an attack-free MANET wherein the classifier learn the legitimate values of a feature. Thus, any value that is not in these legitimate values means that an attack is taking place. However, one of the aforementioned IDSs does the inverse, and trains its classifier in the presence of known attacks. Classification-based IDSs are proposed to handle a wide range of attacks, even the unknown ones. However, the necessity of a training phase, that takes place usually in a scenario different from the operating scenarios, and the negative impact of mobility on the performance constitute the basic obstacles to employ them in MANET.

- Trust-based detection: Here, each node aims at identifying the distrustful nodes, and further avoid, as much as possible, cooperating with such nodes. Towards this purpose, each node observes its neighbors, i.e. the nodes within its radio range, in order to increase (resp. decrease) its trust in the neighbor node that cooperates correctly (resp. misbehaves). The observed cooperation is mostly related to packet forwarding and/or route calculation in the proactive routing protocol (e.g., DSR and AODV). In addition to the self-observations, some of the trust-based IDSs use recommendations where each node provides, periodically or upon the reception of a request, information about its distrustful nodes. In order to avoid the false accusations, some of these IDSs propose to solely exchange information about the trustful nodes. Others propose to accept only the recommendations provided by the trustful nodes. Since the algorithms used in building the trust relations between the nodes are usually simple, these IDSs impose a limited computing overhead. However, using the recommendations increases traffic overhead, and makes these IDSs vulnerable to false praises or accusations. Moreover, they are targeting limited types of attack, basically the dropping packets.

- Automata-based detection: This mechanism of detection depends on building a Finite State Machine (FSM) in order to model either the routing functions or the attack's scenarios. In the first case, if the routing activity of a node triggers unexpected or unknown transitions in one of the employed FSMs then an attack is confirmed. While in the second case, if the routing activities of a node lead to the final state in a FSM, which identifies the

required actions for an attack to be successful, then this attack is confirmed. The majority of the IDSs that are based on FSMs are oriented for the known routing protocols such as AODV and OLSR. They also provides a high detection accuracy. However, some of those are not able to identify the source or the type of the detected attack.

- Rule-based detection: Such detection depends on rules that have a conditional form (If cond. Then). Those rules are used to identify either the correct behavior or the malicious actions. In the first case, the rules represents the conditions that should be respected during, e.g., the route calculation or packet forwarding. Thus any violation of these conditions is considered as an attack. In the second case, the rules represents the actions that could be realized by the attacker. Thus, achieving one or more of these rules means that an attack has taken place. Similarly to the Automata-based detection, the rule-based IDSs are mostly used for the attacks targeting the routing protocols. However, some of those IDSs add some rules concerning packet forwarding. In general, these IDSs are not cooperative, and hence they are not able, in most of the cases, to identify precisely the source of an attack.

To sum up, these listed IDSs should be considered in the context for which they are proposed. However, we can generalize that the cooperative IDSs are most adapted to the dynamical nature of MANET, and they are able to detect a wide range of attacks. There is a wide diversity of the proposed intrusion detection approaches and methods for MANET. We cannot consider that one detection approach/method is better than the others in all the cases. They should be compared according to the operating scenario or environment. In addition, using more than one detection method in the same time increases the detection accuracy. However, ensuring the integration of several detection methods is not a trivial task.

All the listed IDSs focus on their capabilities of detecting the attacks and aim at providing a high detection accuracy. More precisely, they aim at increasing the detection rate along with minimizing the number of false alarms. But, few of those take into account the other factors that can deeply affect the intrusion detection in MANET. In fact, the accuracy of detection could be severely affected by the misbehaving nodes which aim at bypassing or even disrupting the detection. To achieve their purpose, these misbehaving nodes may provide incorrect evidences or improperly perform the detection-related operations. Therefore, a good IDS should be able to identify or, at minimum, avoid the damages coming from such nodes. A proper IDS for MANET should not only offer a high detection rate, but it should also takes into account the performance issues in such networks. Indeed, maintaining the available resources (e.g., battery life, bandwidth, processing capabilities) is a critical condition for any system or protocol oriented to work in MANET. Minimizing the consumed resources should not be ignored or forgotten during all the detection phases, i.e., evidences gathering, diagnosis and correlation, and countermeasures. Otherwise, the node will be forced to choose between keeping detecting the attacks, and thus losing rapidly its resources, or abandoning the detection in order to stay for a longer time in the network. Since MANET mostly operates over open, unpredictable, and maybe hostile environments, the evidences provided to the IDS could be incorrect or even falsified. Thus, there is a need to evaluate the reliability of the detection results obtained by the IDS.

Overall, there is a high need to have an IDS in place that offers a high detection accuracy and takes into account the dynamic and distributed nature of MANET. It must be robust, invulnerable, and resist against the misbehaving nodes. It must consider the performance issues and find a trade-off between detection accuracy and resources consumption. It must handle the doubt about its results and provides a metric of detection reliability. In this thesis, a lightweight and robust intrusion detection system for MANET is proposed to handle all the above mentioned circumstances.

CHAPTER 3

INTRUSION DETECTION SYSTEM DEDICATED TO AD HOC ROUTING PROTOCOL

Contents

3.1 Introduction

Securing *ad hoc* networks is particularly challenging because these networks often operate in adverse or even hostile environments [57, 117]. In addition, the open radio-based medium of communication facilitates the listening of transmissions [42]. The dynamic topology [21], the lack of centralized security enforcement points (e.g., switches and routers) [25] and the low degree of physical security of the mobile devices/nodes [79], make MANET more vulnerable to intrusions/attacks compared to traditional infrastructure-based networks [10]. Conventional preventive approaches, e.g., authentication [58], encryption [59], access control [60] and digital signature [61], significantly reduce the scope of the potential attacks but cannot eliminate it. In addition, new attacks emerge and find a way to penetrate the aforementioned mechanisms [65]. This naturally calls for proposing reactive mechanism, such as the Intrusion Detection Systems

(IDS for short) [67], as a second line of defense against the attacks that would succeed [66]. IDS attempts to identify both outsiders (i.e., non authorized nodes trying to break into the system and misuse it) and insiders (i.e., nodes with a legitimate access that abuse their privileges) [117]. In MANET, the IDSs have to face the following challenges:

- the limitation of the node's power supply requires an energy-efficient intrusion detection so as to maximize the survivability of the node and of the network,
- the stringent computing power, e.g., memory size and CPU processing, of the majority of the nodes in MANET makes the IDSs that impose intensive calculations impractical,
- the available bandwidth and radio frequencies are largely restricted and vary rapidly, thus heavily and excessive radio transmissions are not recommended,
- the intrusion detection must seamlessly adapt to the topological changes that result from the mobility,

The IDSs that have been developed for the infrastructure-based networks failed to meet these challenges. This calls for designing new IDSs that take into account these peculiarities.

We propose a **l**ightweight and robust **i**ntrusion **d**etection system for ad hoc **r**outing protocols (LIDR). LIDR distinguishes itself by auditing logs instead of sniffing/inspecting the traffic as it is the case with almost all IDSs designed to operate in MANET. Note that sniffing packets requires setting the wireless interface to promiscuous mode, which leads to a permanent strain of energy [118]. Moreover, it significantly consumes the computing power due to the packet-level analysis [73]. Our IDS classifies the intrusion evidences according to their gravity and level of suspicion. Thanks to such classification, the in-depth intrusion diagnosis is carefully planned. Indeed, this diagnosis starts only when there is a sufficient degree of suspicion. In order to minimize the traffic generated when gleaning intrusion evidences, our IDS is distributed and cooperative: it parses logs as close as possible from the device that generates it. Finally, the proposed IDS does not require any change on the implementation of the used routing protocol, services, or applications because it extracts the evidences from the logs. Routing logs (and any other beneficial logs) are parsed; this consists in discovering the suspicious actions of other node(s) so as to identify a pattern that characterizes an intrusion attempt. In other words, the discovered evidences are matched against a set of predefined intrusion signatures. Contrary to most other IDSs, our system applies a signature-based detection rather than anomaly-based detection which is based on identifying the deviations from the normal behavior of the node/the network. This choice is motivated by the fact that signature-based detection guarantees a high rate of detection along with a limited number of false alarms [119]. However, even though the signature-based detection cannot detect unknown attacks, it should be used, at minimum as a first stage of detection to detect accurately and lightly the wide range of the known attacks targeting MANETs [75].

Although physical, data link and network layers are all subject to vulnerabilities, our IDS is dedicated to the attacks on the network layer. Indeed, the vulnerabilities in the physical and data link layers are common to IEEE 802.11 wireless networks, while the vulnerabilities in network layer are specific to *ad hoc* networks. More specifically, zeroconf and routing protocols constitute the main target of attacks [56]. The reason is threefold. First, no security countermeasure is

specified or implemented as a part of the drafts or RFCs (Request For Comments) proposed through the Internet Engineering Task Force Manet[12] or Zeroconf[13] working groups. Second, any device may operate as a router, which facilitates the manipulation of multi-hops messages as well as the compromising of the routing functionality. Third, since routing is one of the most vital functions in MANET, disrupting the correct operations of the routing protocol causes the most devastating damage [57]. Although these attacks threaten both routing and zeroconf protocols, we hereafter concentrate on the routing protocol and illustrate our presentation by exemplifying attacks on a proactive, link state routing protocol called OLSR. We further evaluate the performance of our IDS by challenging it against an attack we purposely developed. The evaluation takes place in a simulated MANET coupled with virtual machines. More details are covered in the remaining part of this chapter. In Section 3.2, we provide an overview about OLSR and further details the attacks targeting this protocol. Then, we introduce our IDS in Section 3.3 and evaluate its performance in Section 3.4.

3.2 Attacks on the OLSR Protocol

Attacks on routing protocols fall into two main categories, passive *versus* active [120]. A passive attack typically intercepts the routing messages in order to get an access to useful information without disrupting the routing functions. An active attack involves an unauthorized action (e.g., the fabrication or tampering of the information). Thus, the normal functioning of the MANET is disrupted [121]. Active attacks are further sub-classified according to the action witch is undertaken on the routing messages into: drop, modify and forward, as well as active forge attacks [122]. But before going further, let first introduce the OLSR protocol [9] and then detail the attacks targeting this protocol (3.2.2).

3.2.1 Background on OLSR

The Optimized Link State Routing protocol (OLSR for short) is a proactive (or table driven) and link state routing protocol, tailored for MANET. It was originally drafted in 2000 as part of the effort carried by the IETF MANET working group [123], and then finalized as the Request For Comment (RFC 3626) [9] in 2003. Recently, a second version of OLSR (also called OLSRv2) has been drafted [124]. OLSRv2 distinguishes from its predecessor by considering the notion of link metric rather than only the hop count for measuring the shortest path. It is supposed that OLSRv2 would have more modular and flexible architecture that facilitates add-ons extensions for e.g., security, QoS, and multicast [125]. It will also possess a simplified packet format and reduced-size messages due to address compression [126]. However, both OLSR and OLSRv2, retain the same basic algorithms and mechanisms that are hereafter detailed. OLSR aims at (i) maintaining a constantly updated view of the network topology on each node and (ii) reducing the overhead resulting from disseminating the control messages. One fundamental is the notion of MultiPoint Relay (MPR): each node selects a subset of its 1-hop neighbors to be its MPRs. MPRs are charged of forwarding the control traffic that is intended for a periodic diffusion

[12] http://www.ietf.org/dyn/wg/charter/manet-charter.html
[13] http://www.zeroconf.org

over the entire network. The basic idea is to select the minimum number[14] of MPRs so as to reduce the number of nodes retransmitting control messages and hence, keep to a minimum the bandwidth overload. It is highly encouraged to use OLSR in large and dense networks, as the optimization done using MPRs works efficiently in such a context. Another optimization consists in spreading only partial links state, i.e., a node advertises only the nodes that have selected it as MPR (named MPR Selectors). However, additional link state could be flooded for redundancy purposes. OLSR works in a completely distributed manner, and does not require any central entity. It does not necessitate a reliable transmission of control messages; the periodical generation/sending of control messages sustains the reasonable loss of messages.
In OLSR, we can note the next basic functions[15]:

The neighborhood discovery aims at discovering the nodes that are within radio range. These nodes are named the *1-hop neighbors*. It also attempts to define whether the links with the 1-hop neighbors are bidirectional (i.e., symmetric) or unidirectional (i.e., asymmetric). For this purpose, each node periodically broadcasts a *hello* message including[16][17]: (i) the 1-hop neighbors from which a recent control message has been received but the symmetric links with them have not yet been confirmed, (ii) the 1-hop neighbors with which symmetric links, and (ii) the 1-hop neighbors that are selected to act as MPRs.

Table 3.1: Link Code of a *hello* message

UNSPEC_LINK	No information is provided about the link
ASYM_LINK	Link is asymmetric, i.e., neighbor interface can be haired
SYM_LINK	Link is symmetric
LOST_LINK	Link has been lost
SYM_NEIGH	Node has, at least, one symmetrical link with this neighbor
MPR_NEIGH	Node selects this neighbor as a MPR
NOT_NEIGH	Node is no longer or not yet considers the neighbor as a symmetric

Upon receiving a *hello* message, the node updates its routing table. More particularly, if the node finds itself advertised as a neighbor then it confirms the existence of a symmetric link with the originator of the message. Otherwise, it defines the link as asymmetric. If the node finds itself selected as MPR neighbor, then the originator of the message is defined as MPR selector. The received *hello* message permits also to learn the 2-hops neighbors and further to select the best MPRs.

[14] A redundant MPR may be selected to increase the reachability, but the overhead is also increased.
[15] Other functions of OLSR are provided in Appendix A
[16] The link layer information provided by e.g., the IEEE 802.11 protocol, may also be used.
[17] The *hello* message has a Time To Live (TTL) equals to 1, thus it is not relayed to more than one hop.

Multipoint Relay selection. As aforementioned, the optimization of OLSR comes from the fact that the control traffic is only relayed by a subset of nodes called MPRs. Indeed, when a node broadcasts a control message then only its MPRs retransmit this message. While the non-MPR nodes process the message, if they receive it, without retransmitting it. As a consequence, the overhead resulting from flooding the control traffic is reduced. Figure 3.1 illustrates the difference between the classical flooding and the optimized flooding that depends on the MPRs. In Figure 3.1-a, when the node in the center broadcasts a message, all its 1-hop neighbors retransmit this message. Thus, every 2-hops neighbor receives the same message several times. As a consequence, there is a great overhead resulting from the large number of redundant retransmissions. While in Figure 3.1-b, only the 1-hop neighbors that are selected as MPRs (represented by the black circles) retransmit the message emitted by the node at the center. The other 1-hop neighbors (represented by the white circles) do not need to retransmit this message because the MPRs have already covered all the 2-hop neighbors. Hence, the redundant retransmissions, and consequently the traffic overhead, are significantly reduced.

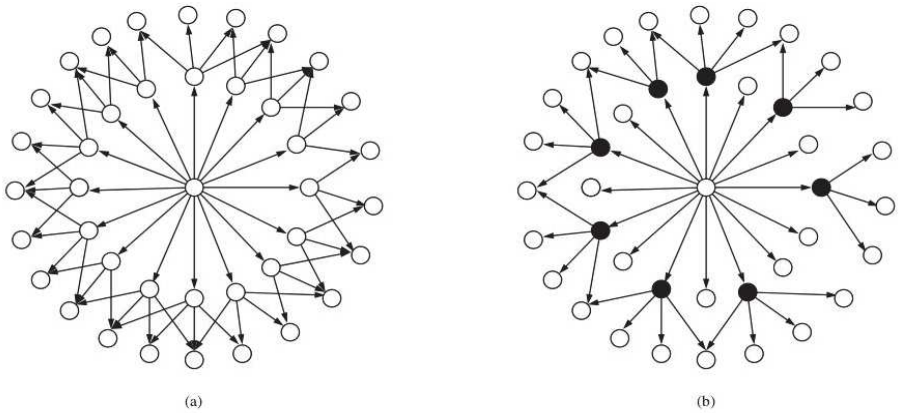


Figure 3.1: Classic vs. optimized flooding [6]

In order to have the maximal optimization, each node should select the smallest possible MPRs set [18]. On the other hand, the node must select sufficient MPRs that enable its messages to be diffused into the entire network. To answer the aforementioned requirements, the node must respect three basic rules during the MPR selection.

- **Rule 1:** The node must select its MPRs from its symmetric 1-hop neighbors. Thus, the MPRs can relay the messages emitted by the node. In Figure 3.2, A considers B, D, and E as symmetric 1-hop neighbors. Therefore, MPR_A should be a subset of $\{B, D, E\}$.

- **Rule 2:** The MPRs should cover, in terms of radio range, all the symmetric 2-hops neighbors. Thus, a message emitted by the node and relayed by its MPRs is received by all the symmetric 2-hops neighbors of this node. Covering the symmetric 2-hop neighbors by the MPRs is necessary to ensure diffusing the broadcast messages into the entire network. As a consequence, a 1-hop neighbor that is the only one to provide reachability to a 2-hops

[18] Redundant MPRs may be selected in order to increase the reachability of the node. Such increase is recommended, for example, when a lot of changes in the neighborhood are observed.

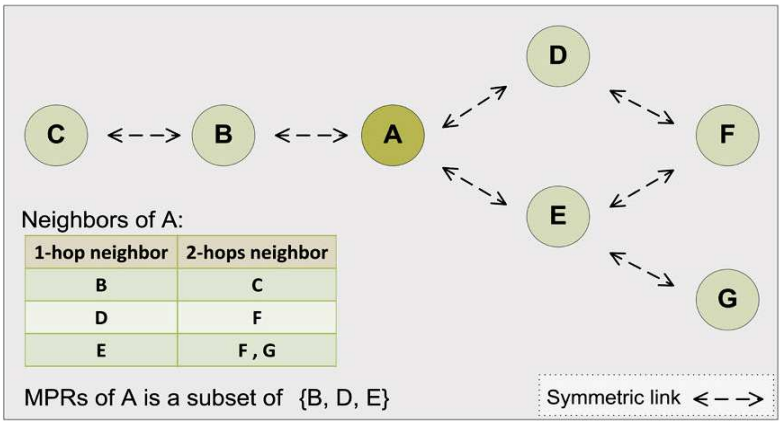


Figure 3.2: The MPRs are selected from the symmetric 1-hop neighbors

neighbor should be selected as a MPR [19]. Figure 3.3 illustrates the impact of the coverage of the 2-hops neighbor on the MPR selection that is done by the node A. Here, A aims at selecting the smallest subset of its 1-hop neighbors set (i.e., $\{B, D, E\}$) that covers its 2-hop neighbors C, F, and G. A selects B as a MPR because this latter is the only 1-hop neighbor that covers C. Similarly, A selects E as a MPR because E is the only 1-hop neighbor that covers G. Since D provides reachability only to F which is already covered by a MPR (i.e., E), A does not select D [20]. Thus, A ensures having the smallest MPRs set that covers its 2-hops neighbors.

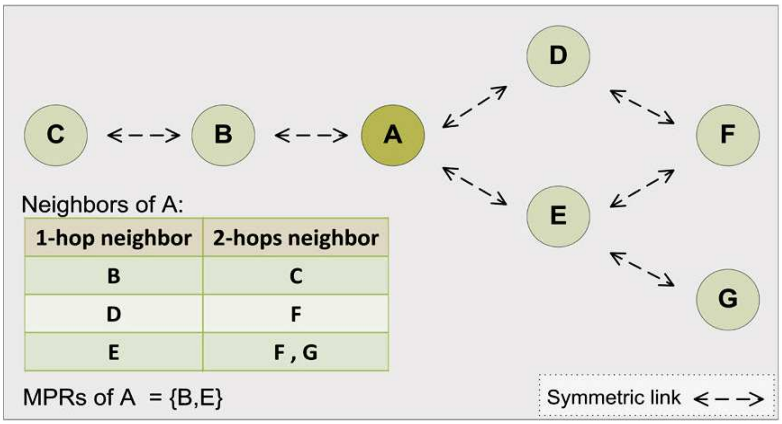


Figure 3.3: MPR selection: impact of coverage

- **Rule 3:** The willingness to forward traffic on behalf of others should be considered during the MPRs selection. In fact, being a MPR is costly in terms of resources (e.g. battery life) because the MPRs relay the broadcast traffic in the network. Thus, some nodes may not accept (or not prefer) to be selected as MPRs in order to maintain their resources. Therefore, the node must select the 1-hop neighbors that accept to be MPRs. The higher the willing-

[19] Note that this rule is valid provided that the 1-hop neighbor accepts to be a MPR.
[20] Note that D will be selected as a MPR if A desires to have redundant MPRs in order to increase its reachability.

ness of the 1-hop neighbor to be a MPR, the bigger the chance that this neighbor is selected as a MPR. Note that, every node advertises its willingness through the *hello* message. This willingness takes one of five values ranging from WILL_NEVER to WILL_ALWAYS (Table 3.2). In practice, the node must (resp. must not) select as a MPR the 1-hop neighbors that have a willingness equals to WILL_ALWAYS (resp. WILL_NEVER). In Figure 3.4, we illustrate the impact of willingness on the MPR selection that is performed by A. Similarly to Figure 3.3, A selects E as a MPR because this latter is the only 1-hop neighbor that covers G. Moreover, E does not refuse to be a MPR because it has a willingness equals to WILL_DEFAULT. In contrast to Figure 3.3, A selects D as a MPR even though D covers only the 2-hops neighbor F, which is already covered by a MPR (i.e., E). This happens because D has a willingness equal to WILL_ALWAYS. Moreover, A does not select B as a MPR even though this latter is the only 1-hop neighbor that covers C. This happens because B has a willingness equals to WILL_NEVER, i.e., it refuses to be selected as a MPR.

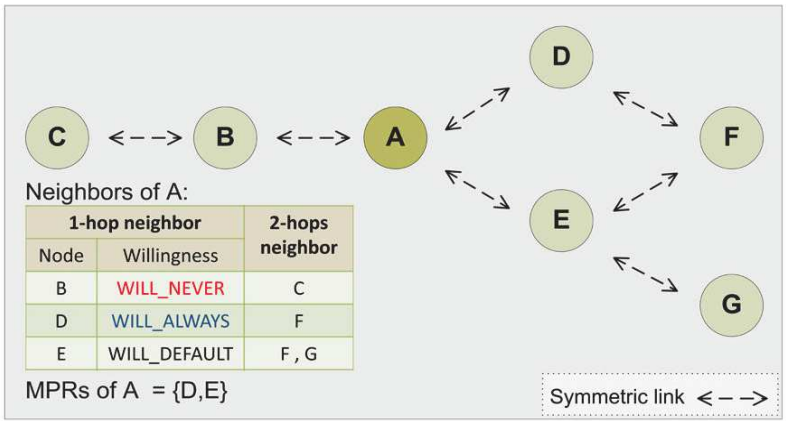


Figure 3.4: MPR selection: impact of willingness

Table 3.2: Levels of willingness

WILL_NEVER	0
WILL_LOW	1
WILL_DEFAULT	3
WILL_HIGH	6
WILL_ALWAYS	7

Declaring the MPR Information. In addition to relay the control traffic, the MPRs are in charge of routing the messages between the nodes. In fact, any route is composed of a sequence of MPRs along the path from the source to the destination. Therefore, every MPR periodically

broadcasts a Topology Control message (TC for short), which is intended to be diffused over the entire network. In this message, the MPR declares its MPR selector set, i.e, the 1-hop neighbors that have selected it to act as a MPR [21]. Based on the information included in the TC message, nodes update their topological information and further calculate the routes. TC messages are periodically sent but if the MPR selector set changes, a new TC message should be transmitted as soon as possible in order to override the stale topological information. In order to keep track of the most recent topological information, the TC message is provided with a sequence number called ANSN (Advertised Neighbor Sequence Number) [22]. If the node receives a TC message whose ANSN is less than the ANSN that is registered for the message's originator then the node ignores this message.

Gateways and multiple interfaces nodes. OLSR MANET operates either in isolated manner or connected to other networks (or even Internet) where other routing protocols, e.g., OSPF [23], are used. In the latter case, external routing information is injected into the OLSR routing domain through gateways. A gateway is a node that is equipped with, both, OLSR and non-OLSR interfaces. In Figure 3.5, X is a gateway that has, in addition to the OLSR interface, an Ethernet interface which is associated to a local area network (LAN for short). The gateway

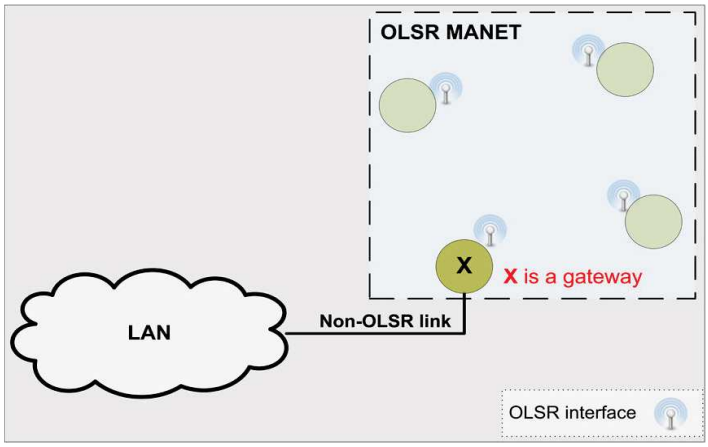


Figure 3.5: Gateway in OLSR

imports (and resp. exports) the routes provided by other routing protocols (resp. OLSR). In order to announce its reachability to an external host and/or network, the gateway periodically broadcasts a Host and Network Association (HNA) message. In this message, the gateway lists the network address and netmask of the associated external host and/or network. Thus, the recipient of the HNA message has a sufficient information to calculate the routes towards the external host and/or network. Note that the gateway differs from the node which runs OLSR on multiple interfaces in order to, for example, increase its connectivity. In Figure 3.6, X has two OLSR interfaces, $Inter\_X_1$ and $Inter\_X_2$, witch are in the radio range of Y and Z respectively. In

[21] A MPR may declare additional topological information, e.g., its MPRs and/or its 1-hop neighbors, for redundancy purposes.

[22] ANSN is wraparound number, i.e., when it reaches an extreme boundary, it is reset to zero.

[23] www.ietf.org/rfc/rfc2328.txt

order to avoid considering each interface as an independent node, X needs to map its interfaces to one identity. For this purpose, X first chooses one of its interfaces to be the main address [24] [25], which is used as the originator address in the routing messages emitted by X. After that, X

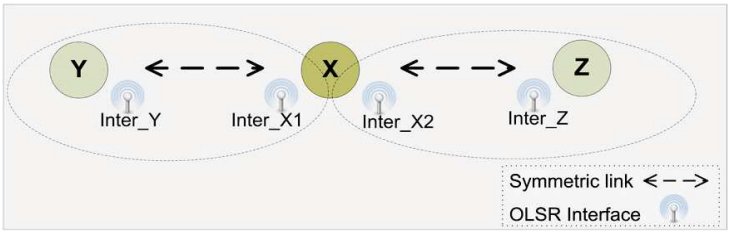


Figure 3.6: Multiple interfaces node in Olsr

maps the other interfaces to its main address through broadcasting, on a regular basis, a Multiple Interface Declaration (Mid) message. In this message, X announces its main address as well as its other Olsr interfaces. Thus, other nodes recognize that $Inter\_X_1$ and $Inter\_X_2$ belong to the same node X. It is worth to mention that Mid and Hna messages are disseminated by the Mprs since the information included in these messages must be delivered to all nodes. However, Mid and Hna messages are excluded from the specification of Olsrv2. In this version, the Tc message is responsible for declaring the node's multiple interfaces and/or non-Olsr interface(s).

Overall, Olsr is a proactive *ad hoc* routing protocol that provides constantly updated routes. Therefore, this protocol goes in favor of the applications that do not accept long delay in transmitting data traffic. Nodes exchange Olsr control traffic thanks to two types of message: a *hello* message that enables neighbor discovery and further Mpr selection, and a Tc message that serves in advertising the link state information between the nodes so that the available routes can be found. The standard specification document of Olsr does not define security measures. However, it lists possible vulnerabilities including: the breach of confidentiality, the breach of integrity, non-relaying, replaying, and security threats coming from insecure external routing domain. In the following section, the attacks targeting Olsr are represented.

3.2.2 Attacks Classification and Modeling

Attacks against Olsr are hereafter detailed and classified according to the model introduced in [11]. This model provides a level of expressiveness necessary to specify the relationship between the actions and their related consequences. We further enrich this model with temporal annotations (Table 3.3). As a consequence, complex attacks, their constituting actions and consequences are successfully depicted and categorized as drop attack, active forge attack, and modify and forward attack.

[24] Note that if the node has only one Olsr interface then this interface is used as the main address.
[25] There is no rule of choosing the main address but the node should keep the same main address all the time.

Table 3.3: Notations

$Y \xrightarrow{M_t} X$	At t, Y sends a message M that is received by X
$Y \xrightarrow{M_t}$	At t, Y sends a message M
$Y \not\xrightarrow{M_t}$	Y does not send a message M at t
$\xrightarrow{M_t} X$	X receives a message M at t
$\not\xrightarrow{M_t} X$	X does not receive a message M at t
$\triangle t, \triangledown t$	Period of time
$\ni$:	Such that
$Card(X)$	connectivity of X
$\mathcal{I}$	Set of malicious nodes
NS_X	Set of 1-hop neighbors of X
sq	Sequence number
hc	Hop count
w_X	Willingness of X to forward packets on behalf of others
MPR_X	MPRs of X
Sel_{MPR_X}	MPR Selectors set of X
$Hello$	Hello message
TC	Topology Control message
MID	Multiple Interface Declaration
HNA	Host and Network Association
CM	Control Message
RM	Received Control Message
FM	Control Message intended to be forwarded

3.2.2.1 Drop Attack

In practice, a drop attack consists in dropping a control message instead of relaying it. This dropping has an impact only if the dropped control message is intended to be forwarded: as illustration, suppressing a *hello* message that is broadcasted over one hop, is natural. Thus, threatened messages are restricted to the messages that are created and/or broadcasted by a MPR so as to be re-diffused by other MPRs, i.e., Topology Control (TC), Multiple Interface Declaration (MID), and Host and Network Association (HNA) messages. More particularly, let us consider a host H that sends a control message which is intended to be forwarded. This message, which is originated at t ($H \xrightarrow{FM_t} I$), is received by a malicious node I that drops it. In practice, I drops a control message if it does not forward this message during a period $|t' - t|$

lower than the maximum allowed[26] period $\triangle t$:

$$\begin{gathered} H \xrightarrow{FM_t} I, I \not\xrightarrow{FM_{t'}}, |t' - t| > \triangle t \\ \Downarrow \\ I \in \mathcal{I} \end{gathered} \tag{3.1}$$

Such dropping is naturally applied on any packet that is empty, expired, duplicated, or out of order. In addition, restrictive forwarding is provided: (only) the 1-hop neighbor(s) of H that have been selected by H to act as MPR(s) forward. Apart from the aforementioned conditions that lead to a convenient dropping, the remaining dropping may be attributed to an intruder (or a compromised node), a selfish or a malfunctioning node. The attempt to drop any packet is termed *blackhole* attack. In practice, it should be concluded that I creates a *blackhole* if I forwards no message, i.e., $FM_I(\triangle t) = 0$, even though it has received some forwardable messages, i.e., $RM_I(\triangle t) \neq 0$, during a period of time $\triangle t$.

$$\begin{gathered} RM_I(\triangle t) \neq 0, FM_I(\triangle t) = 0 \\ \Downarrow \\ I \in \mathcal{I}, \\ \text{I is a blackhole} \end{gathered} \tag{3.2}$$

A *gray hole* corresponds to a selective dropping that is performed arbitrarily or according to some criteria e.g., a given source or destination in the message. It is detected by taking into account additional fine-grained or discriminative criteria, including, for instance, the choice of a specific final or 1-hop-away destination or source, the percentage of packets received or forwarded, the message type. In a nutshell, when the difference between received and forwarded control messages during a period of time exceeds a certain threshold α, I should be concluded as a *gray hole*. The value of α is a policy decision, which is predefined by the end user. However, α should exceed the rate of natural dropping that results from collision, limited transmission power, lost packets, etc [72]. In [127], experiments show that α should not exceed 10% (resp. 15%) for the 20 (resp. 60) node network in order to detect more than 90% of *gray holes*.

$$\begin{gathered} |RM_I(\triangle t) - FM_I(\triangle t)| < \alpha \\ \Downarrow \\ I \in \mathcal{I}, \\ \text{I is a gray hole} \end{gathered} \tag{3.3}$$

Rather than dropping the routing control messages, an opposite misbehavior consists in introducing falsified routing information.

3.2.2.2 Active Forge Attack

An active forge attack takes place when a node introduces/provides novel deceptive routing control messages. Among others, the *broadcast storm* stems from forging control messages so

[26] Any control message is characterized by an emission interval as well as a holding time in the routing tables. These two related values participate in setting a validity time that should be herein considered in conjunction with the message type.

as to exhaust resources (e.g., energy, network bandwidth) and saturate the communication medium. For this purpose, an intruder I forges a large number of control messages CM within a short period of time $\bigtriangledown t$ (see Expression 3.4). This attack may be local (e.g., targeting the 1-hop neighborhood through *hello* message) or global (e.g., applied on the messages that are relayed). It may also be conducted in a distributed manner with several colluding nodes that are simultaneously emitting (a large amount of) control messages.

$$\begin{gathered} I \xrightarrow{CM_t}, I \xrightarrow{CM'_t}, |t' - t| < \bigtriangledown t \\ \Downarrow \\ I \in \mathcal{I} \end{gathered} \tag{3.4}$$

A special type of *broadcast storm* is the *routing table overflow* attack, which threatens especially the proactive routing protocols (e.g., OLSR) as they periodically update the routing information [128]. One or several colluding intruders prevent the well-behaving nodes from discovering new routes by dumping the network with route advertising non-existing nodes. In general, a *broadcast storm* constitutes a kind of denial of service that is characterized by a high visibility. Therefore, it is typically combined with masquerading - although, less intrusive attacks may also be preferred. Masquerading lies in maliciously switching the identity of the originator of a control message CM ($I \xrightarrow{CM(I)_t}, I \xrightarrow{CM(S)_t}$). Note that this case is distinguished from a node that holds several interfaces and that advertises them in a dedicated MID message. Apart from masquerading a denial of service, identity spoofing may be intended to create conflicting route(s). The identity spoofing is a fundamental step for creating loop(s) wherein some legitimate nodes spin the same control message(s) to infinity. As pointed out in [129], the spoofing attack may also be coupled with a modification of the willingness field so as to impact the MPR selection (Expression 3.5). In practice, if I emits a *hello* message including an originator address already assigned to another node S ($I \xrightarrow{hello(S)} D$), then I modifies the local topology as seen by its 1- and 2-hops neighbors. In addition, if I modifies the willingness field w'_S of S (with $w'_S \neq w_S$), then the selection of S as MPR is impacted; recall that MPRs are selected among the nodes with highest willingness and in case of multiple choices, the node which provides a reachability to the maximum number of nodes is primarily selected. For instance, a node whose willingness attribute set to $WILL\_NEVER$ (resp. $WILL\_ALWAYS$), is never (resp. always) selected as MPR, i.e., $w'_S = WILL\_NEVER \Rightarrow S \notin MPR_D$ (versus $w'_S = WILL\_ALWAYS \Rightarrow S \in MPR_D$).

$$\begin{gathered} S \xrightarrow{hello(S,w_S)_t} D, I \xrightarrow{hello(S,w'_S)t'} D, \\ |t' - t| < \triangle t, w_S \neq w'_S \\ \Downarrow \\ I \in \mathcal{I}, \\ S \text{ is spoofed}, \\ w'_S = WILL\_NEVER \Rightarrow S \notin MPR_D, \\ w'_S = WILL\_ALWAYS \Rightarrow S \in MPR_D. \end{gathered} \tag{3.5}$$

Active forge attacks are not restricted to identity spoofing (possibly coupled with a tampering of the willingness field). They also cover the tampering of control messages including incorrect

adjacent links (*hello* messages), topology information (TC messages), and network interfaces (MID and HNA messages). In the following, we detail each of those. In the first case (expression 3.6), I forges a *hello* message, which declares a list of 1-hop and symmetric neighbors NS'_I differing from the real set NS_I.

$$\begin{gathered} I \xrightarrow{hello(NS'_I)_t}, NS'_I \neq NS_I \\ \Downarrow \\ I \in \mathcal{I} \end{gathered} \tag{3.6}$$

Whenever forging the set of neighbors NS'_I, the attacker has three options:

1. declaring a non-existing node as a symmetric 1-hop neighbor, implies that I (or another misbehaving node) is further selected as a MPR (Expression 3.7). Indeed, if I advertises a non-existing node N ($N \notin \mathcal{N}$ with $\mathcal{N}$ defining the set of nodes composing the OLSR network[27]), I ensures that no other (well-behaving) MPR claims being a 1-hop symmetric neighbor of N. Recall that the set of MPRs is selected so that all the 2-hops and symmetric neighbors are covered thus, I is surely selected as a MPR.

$$\begin{gathered} S \xrightarrow{hello(NS_S)_t} I, I \xrightarrow{hello(NS'_I)_{t'}} S, |t' - t| < \triangle t, \\ \exists N \in NS'_I \ni: N \notin \mathcal{N} \cap NS_I \\ \Downarrow \\ I \in \mathcal{I}, \\ \exists I' \in \mathcal{I} \cap NS_S \ni: I' \in MPR_S, \\ Card(NS'_I \backslash (NS'_I \cap \mathcal{N})) > 0. \end{gathered} \tag{3.7}$$

   This assertion is verified as long as no other misbehaving neighbor of S is claiming the same. Overall, inserting at least one non-existing neighbor ($\exists N \in NS'_I \ni: N \notin \mathcal{N} \cap NS_I$) guaranties that a misbehaving node I' (with $I' \in \mathcal{I}$) is selected to act as MPR of S ($\exists I' \in \mathcal{I} \cap NS_S \ni: I' \in MPR_S$). In addition to the above, the connectivity of I is also artificially increased ($Card(NS'_I \backslash (NS'_I \cap \mathcal{N})) > 0$).

2. declaring that an existing node is a symmetric 1-hop neighbor while the declared node is far away (i.e., is not a neighbor) or is an asymmetric neighbor ($\exists X \in NS'_I \cap \mathcal{N} \ni: X \notin NS_I$). This claiming increases artificially the connectivity of I, i.e., $Card((NS'_I \backslash NS_I) \cap \mathcal{N}) > 0$. If no (well-behaving) MPR covers S ($\nexists A \in \mathcal{N} \backslash \mathcal{I} \ni: A \in NS_S \wedge X \in NS_A$), then at least

[27] According to the OLSR RFC [9], messages can be flooded into the entire network (with a maximum network diameter defined by the message Time To Live field, TTL for short), or flooding can be limited to nodes within a diameter (defined in terms of number of hops) from the originator of the message. For the sake of clarity, let be $\mathcal{N}$ represent the network in both case.

one misbehaving node (e.g., I') is selected as a MPR of S ($\exists I' \in \mathcal{I} \ni: I' \in MPR_S$).

$$\begin{gathered} S \xrightarrow{hello(NS_S)_t} I, I \xrightarrow{hello(NS'_I)_{t'}} S, |t'-t| < \triangle t, \\ \exists X \in NS'_I \cap \mathcal{N} \ni: X \notin NS_I \\ \Downarrow \\ I \in \mathcal{I}, \\ Card((NS'_I \backslash NS_I) \cap \mathcal{N}) > 0, \\ \nexists A \in \mathcal{N} \backslash \mathcal{I} \ni: A \in NS_S \wedge X \in NS_A \\ \Downarrow \\ \exists I' \in \mathcal{I} \ni: I' \in MPR_S. \end{gathered} \tag{3.8}$$

Such insertion typically characterizes an attempt to create a *blackhole*: I introduces a novel path toward M that whenever selected provisions the *blackhole*.

3. omitting an existing 1-hop neighbor and symmetric node P ($\exists P \in NS_I \ni: P \notin NS'_I$), decreases artificially the connectivity of both P and I ($NS_I \nsubseteq NS'_I$). Note that if P has no other connectivity than the one obtained through the misbehaving node I, P becomes isolated.

$$\begin{gathered} S \xrightarrow{hello(NS_S)_t} I, I \xrightarrow{hello(NS'_I)_{t'}} S, |t'-t| < \triangle t, \\ \exists P \in NS_I \ni: P \notin NS'_I \\ \Downarrow \\ I \in \mathcal{I}, \\ \exists I' \in \mathcal{I} \cap NS_S, NS_I \nsubseteq NS'_I. \end{gathered} \tag{3.9}$$

Overall, falsifying the neighboring adjacency by inserting (existing or non existing) neighbor(s) and/or omitting real neighbors potentially perverts the local topology seen by S (and more generally by one another) and impacts the MPR(s) selection function carried by S. Nevertheless, in order to be selected as a MPR of S (or to prevent its selection), there is no need for I to falsify the neighboring adjacency. Recall that, in a hello message, a field, termed *willingness*, designates the node's willingness to carry traffic on behalf of others. Indeed, I prevents (resp. ensures) its selection as MPR, by simply setting its willingness field to the value $WILL\_NEVER$ (resp. $WILL\_ALWAYS$). On the whole, the MPR selection is impacted by either falsifying the topological information included in hello message or by making use of the willingness attribute. Another (perhaps more straightforward) alternative refers to a slander node I declaring (resp. not declaring) itself as a MPR although it has not (resp. has) been selected as a MPR [130]. For this purpose (Expression 3.10), I forges a *Topology Control* (TC) message including an incorrect set of 1-hop symmetric neighbors that have selected I as a MPR. Depending on the required level of redundancy, a MPR advertises:

1. the MPR selector(s) Sel_{MPR_I} of I, or,
2. the MPR selector(s) along with the MPR of I: $Sel_{MPR_I} \cup MPR_I$, or,
3. the 1-hop neighbors NS_I of I (hence including the MPR selectors and the MPR of I).

Let $\mathcal{A}_I$ representing the set advertised in the TC message: $\mathcal{A}_I = Sel_{MPR_I} \vee (Sel_{MPR_I} \cup MPR_I) \vee NS_I$. This attack consists in I advertising an incorrect set $\mathcal{A}'_I$ differing from the real one $\mathcal{A}_I$:

$$\begin{array}{c} I \xrightarrow{TC(\mathcal{A}'_I)_t} S, \mathcal{A}'_I \neq \mathcal{A}_I \\ \Downarrow \\ I \in \mathcal{I} \end{array} \tag{3.10}$$

In particular, the possible falsifications lie in either I inserting a non-existing node, or inserting an existing node but non MPR selector, or omitting a node belonging to $\mathcal{A}_I$. Upon the reception of a falsified TC message, S uses the advertised set in this message to update its point of view of the network topology. Consequently, the routing table of S would be corrupted. This corruption contaminates the OLSR network and also any interconnected routing domain. Indeed, a node, well-behaving or not, acting as a gateway exports the wrong OLSR routes. Symmetrically, a malicious node may also import incorrect routes to the OLSR domain. This latter attack termed *sinkhole*, involves a malicious node I defining itself as a gateway that provides an access to associated host(s) and/or network(s). This gateway generates periodically a HNA message including those host(s) and/or network(s) (i.e., the related address(es) and netmask(s)). This attack constitutes a generalization of the previously-defined forging of corrupted TC messages: a node advertises either non-existing or existing but unreachable nodes, or omitting advertising reachable nodes. The similarity with the TC active forging leads us not detailing it.
Overall, the aforementioned forge attacks (e.g., link or route spoofing attacks, sinkhole) necessitate to tamper specific message fields while keeping this message syntactically correct. More generally, bogus control messages can be forged, hence creating an implementation-dependent effect. Generally speaking, similar tampering may be performed by a malicious node that has acted as a MPR prior forwarding a falsified control message.

3.2.2.3 Modify and Forward Attacks

The modify and forward attacks are characterized by an intermediate that captures control messages and replays or modifies those messages before forwarding them. This also includes the replaying of a control message that includes: delaying the emission of this message by recording and forwarding it later (potentially in another area), or repeating this message. As a consequence, routing tables are updated based on obsolete information. Note that each message contains a field indicating the period of time during which this message is considered valid and hence it is used to update the routing table. Both attacks can be systematic (i.e., targeting any multi-hops control traffic) or selective. They may also be performed in a distributed manner (Expression 3.11) with two intruders: one recording the control message from one region so as to replay it in another region (i.e., the one of the colluding intruder). Without loose of generality, let I_1 (resp. I_2) be the node that records (resp. replays) the control message. In practice, I_1 tunnels the control traffic by e.g., encrypting the routing message and/or relying on an alternative network interface (that may be advertised or not). Then, I_2 replays it. This attack leads to the creation of a *wormhole* whose length depends on the distance separating the two intruders. Note that I_2 should replay the message during a period ($\triangle t$) represents the validation time of the message. Otherwise, this message is directly ignored upon its reception.

In addition, I_2 may wait for a period ($\bigtriangledown t$) before replaying the message in order to ensure that the topological information included in this message becomes obsolete. However, such wait has no big importance when I_1 and I_2 are in different regions.

$$\begin{gathered} S \xrightarrow{CM(S)_t} I_1, I_1 \xrightarrow[enc]{CM(S)} I_2, I_2 \xrightarrow{CM(S)_{t'}} Y, \\ S \notin NS_Y, \bigtriangledown t < |t' - t| < \bigtriangleup t \\ \Downarrow \\ S \in NS_Y \\ \Downarrow \\ I_1, I_2 \in \mathcal{I} \end{gathered} \tag{3.11}$$

In order to stay invisible, both I_1 and I_2 may keep the identification field unchanged: the source is S rather than I_1 or I_2. This possibility corresponds to a masquerading. Sequence numbers constitute a standard mechanism that provides protection against replay attacks given that the sequence delivery is not required by Olsr. In counterpart, their usage can be hijacked so that the destination drops the message rather than using it. In practice, a malicious node I increases[28] (resp. decreases) the value of the sequence number sq ($S \xrightarrow{CM(S)_{t,sq}} I, I \xrightarrow{CM(S)_{t',sq'}}, sq' \neq sq$) so that the destination assumes that S is providing the freshest (resp. an obsolete) route and therefore ignores the subsequent (resp. the actual) control message(s).

$$\begin{gathered} S \xrightarrow{CM(S)_{t,sq}} I, I \xrightarrow{CM(S)_{t',sq'}}, sq' \neq sq, |t' - t| < \bigtriangleup t \\ \Downarrow \\ I \in \mathcal{I} \end{gathered} \tag{3.12}$$

A malicious node I may also corrupt the routing table by maliciously modifying a received control message before forwarding it. This modification consists in tampering either the contents of the message or the identification of its source. The former case is similar to the aforementioned forge attack, wherein the malicious node tampers the Mpr selector set in Tc message, the Olsr routes in Hna message or the interface(s) identification in Mid message. While in the latter case, a malicious node I forwards the packet containing the control message without changing the source address [131]. Consequently, two non-consecutive intermediates along the path of this message consider themselves as 1-hop neighbors although they are not in the communication range of each other. A malicious node may also disrespect the flooding mechanism in Olsr (Expression 3.13). This happens when the malicious node retransmits a control message that is received from a non Mpr selector node [129].

$$\begin{gathered} I \notin MPR_S, S \xrightarrow{CM_t} I, I \xrightarrow{CM_{t'}}, |t' - t| < \bigtriangleup t \\ \Downarrow \\ I \in \mathcal{I} \end{gathered} \tag{3.13}$$

[28] A Wraparound mechanism is implemented; when the sequence number reaches an extreme boundary, it is reset to zero.

This attack and any malicious forwarding of a control message, even though this message is not tampered, are classified as modify and forward attacks.
On the whole, attacks targeting OLSR protocol are classified into three types:

- **drop** attack that consists in totally or selectively dropping the control messages (e.g., dropping TC message in order to foil route calculation).
- **active forge** attack that attempts to poison OLSR's functionalities by introducing novel deceptive control messages (e.g., a hello message including an incorrect neighbor set that leads to foil MPR selection).
- **modify and forward** attack lies in an intruder that captures and modifies a control message before forwarding it (e.g., increasing the sequence number of a TC message, hence the subsequent TC messages are ignored) .

An intruder launches the attack either in a standalone manner or in collusion with other intruder(s), constituting what so-called *Byzantine* attack (e.g., *wormhole*), together usually coupled with masquerading. Detecting these attacks is far from being a trivial task because a minor deviation on an attack makes it undetectable. In addition, an attack can be composed of several sub-attacks. In order to tackle these issues, we described the attack as general as possible, hence circumventing possible deviations. We further propose a distributed, cooperative, log-, and signature-based intrusion detection system that detects composed attacks as much as their parties.

3.3 Intrusion Detector

In LIDR, the attack detection is organized under three operational phases (Figure3.7):

- **Evidence gathering**. Events are either gathered locally by parsing logs or obtained by requesting other nodes. More precisely, the routing logs (or any other beneficial log) are continuously parsed in order to extract information about the routing activities of the nodes. Meanwhile, information on the routing activities can be obtained by interrogating other nodes. Overall, both local and remote information is time-stamped and stored in a database.
- **Diagnosis** which consists in analyzing the extracted information so as to discover suspicious evidences. These evidences are then matched against predefined intrusion signatures in order to decide whether an attack occurs or not. Local evidence suggest the presence of an intrusion but they may not be sufficient to prove the intrusion, then additional evidences are obtained by interrogating other nodes.
- **alarm spread** If the diagnosis confirms the existence of an attack then an alarm is broadcasted over the network. This alarm contains information about the detected attack and the identity of the attacker(s).

Before going further, let us first introduce the architecture of LIDR and then detail its operations.

Figure 3.7: Operational phases in LIDR

3.3.1 Conceptual Architecture

In order to cope with the dynamic nature of MANET, LIDR is both distributed and cooperative. The proposed architecture (Figure 3.8) requires that every node participates in the detection[29]. More precisely, each node contains an instance of the IDS, which independently detects the signs of suspicion and cooperates with other instances so as to conduct the diagnosis in a broader range. Our system is written in *Perl*[30]. It is conceptually structured into four components:

- **Coordinator** which constitutes the heart of our IDS that orchestrates the other components. It also performs the cornerstone functionalities: it dynamically parses the logs so as to extract the suspicious signs and matches them against predefined intrusion signatures. Furthermore, it triggers the communication manager (resp. the alarm notifier) in order to launch the advanced diagnosis (resp. alert the network about a detected intruder).

- **Communication manager** that gathers information from the network about the suspicious node(s) as required by the Coordinator. Meanwhile, it answers to the cooperation requests coming from other nodes. This component runs into a separate thread so that other detection operations are not blocked.

- **Knowledge database** includes the information that is extracted from the logs or provided by other nodes through the communication manager. This database is built on *Mysql*[31]. Stored information encompasses neighborhood, MPR selection and topology related information in OLSR, etc.

- **Alarm notifier**: is responsible for alarming the network when an intrusion is detected. Note that more countermeasures, e.g. ignoring intruder's packets or excluding the intruder from the routing table, can be also included.

3.3.2 Evidence Gathering

Our system distinguishes itself from other IDSs by extracting signs of attack attempts from the logs instead of sniffing the traffic. Hence, it minimizes the consumption of energy and computing power (e.g., memory and CPU processing). In fact, sniffing traffic necessitates that the wireless network interface stays in the promiscuous mode at all time, thus it can overhear all the packets

[29] It is possible that a node with very limited resources delegates some of the detection operations to a neighbor device that has high resources. However, the trustworthiness of the delegate device and the efficiency, in terms of maintaining the resource, of such delegation are still under question.

[30] http://www.perl.org

[31] http://www.mysql.com

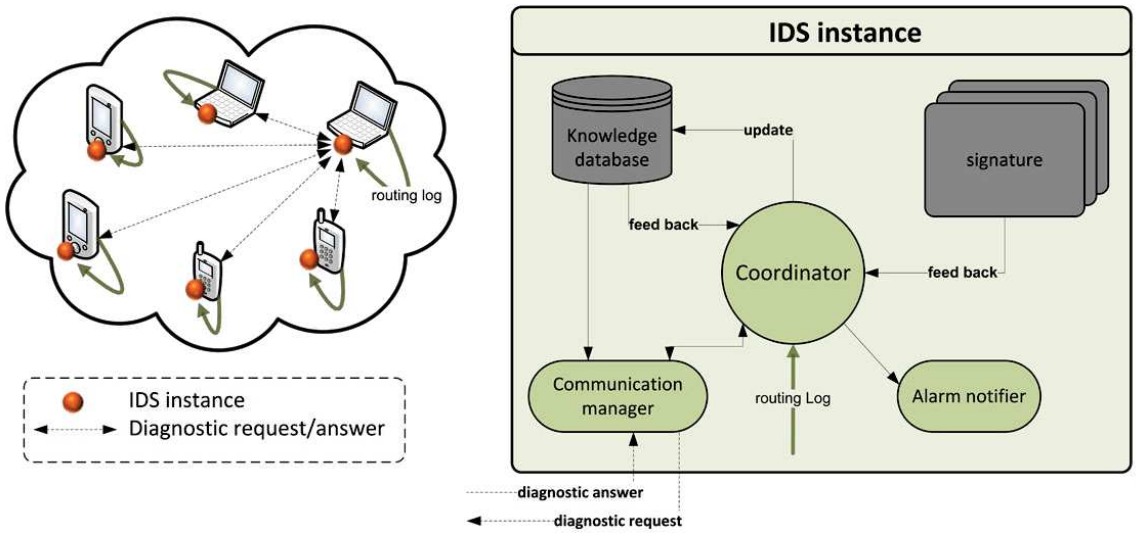


Figure 3.8: LIDR architecture

within its transmission range. But the promiscuous mode consumes more energy and hence reducing rapidly the lifetime of the nodes. Indeed, a non-destination node in the radio range of either the sender or the receiver overhears some or all of their traffic. For the IEEE 802.11 MAC protocol, a non-destination node in discarding mode (i.e. non-promiscuous) enters into a reduced energy consumption mode and discard others' traffic [132]. Such mode requires less energy than the idle mode, which is the default mode in *ad hoc* network, while a non-destination node operating in the promiscuous mode listens to all the traffic, whether or not it is the intended destination. Figure 3.9 represents a part of the experimental measurements realized in [118] about the energy consumption of an IEEE 802.11 2Mbps wireless card. It shows the significant difference in energy consumption between promiscuous and non-promiscuous mode[32]. Besides increasing the amount of consumed energy, sniffing traffic imposes a huge computing overhead [73]. More precisely, the packet-level analysis, which is applied on the sniffed packets, strains significantly the available resources, i.e., memory and CPU processing. Moreover, since all the traffic in the radio range is sniffed, many of the analyzed packets would be redundant and add nothing to the detection. In order to avoid this permanent strain of resources, our system does not sniff the traffic but it rather depends on the logs as a source of the evidences. In practice, it collects and parses in real time the local routing logs. Note that additional logs, e.g., system-, security-related logs, could be integrated and correlated. With the OLSR protocol, the portions of logs that characterize neighborhood relationships, MPR selection, and topological information are highlighted (Figure 3.10). More precisely, the 1-hop neighbors, the 2-hops neighbors, the topological entities, and the MPRs of other nodes are maintained in the Knowledge database. Figure 3.11 provides a simplified view of the used database. Note that the maintained information is extracted from the received *hello* and TC messages. More precisely, each node depends on the received *hello message* to specify the 1-hop neighbors and their types, the 2-hops neighbors, and the MPRs of other (neighbor) nodes, while the TC messages enables

[32]Note that the consumption in the non-promiscuous mode is, sometimes, negative because it requires less energy than the reference idle mode.

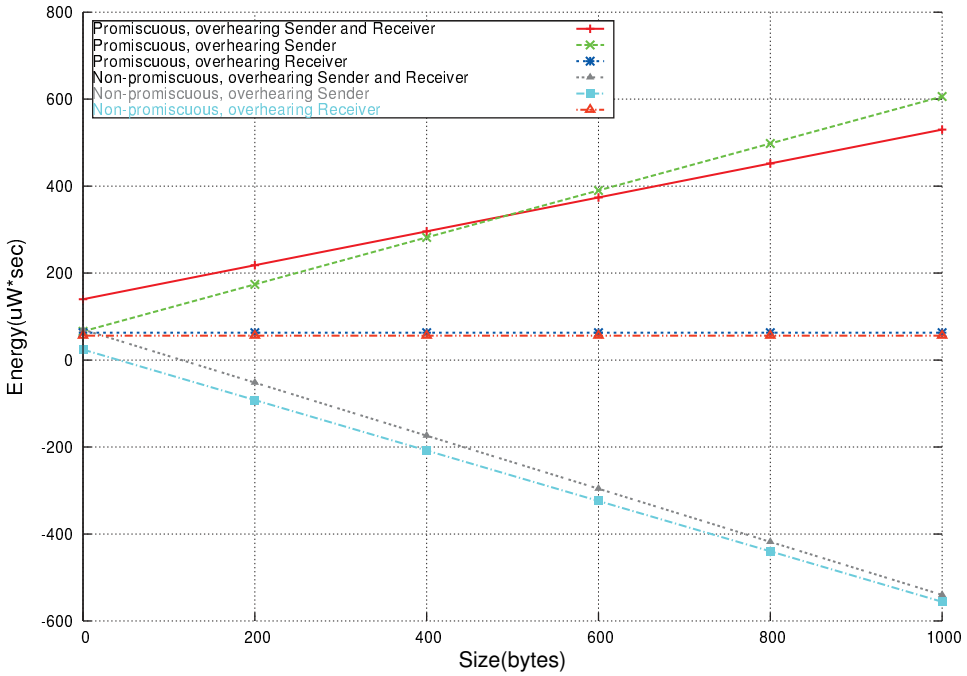


Figure 3.9: Energy consumption in promiscuous and non-promiscuous modes

specifying the MPR selectors and the topological point of view of others. This information is temporally correlated as to detect signs of attack attempts.

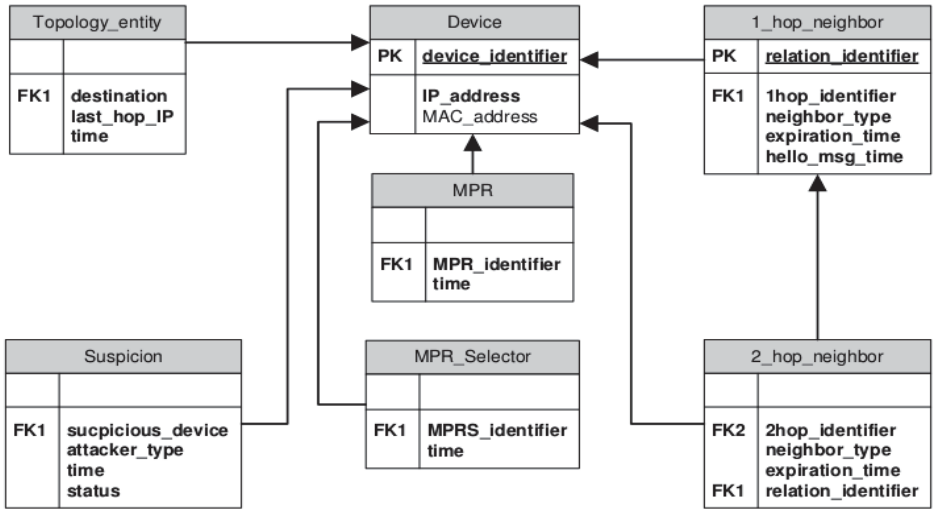


Figure 3.11: Schema of the knowledge database

3.3.3 Attack Diagnosis

In our IDS, the proposed diagnosis is logically composed of two stages: preliminary and in-depth (advanced) diagnosis. Such composition serves in reducing the detection overhead. The pre-


8.21392s 18 [NODE 18] [10.0.0.19] Olsr Msg received with type 1 TTL=1 origAddr=10.0.0.24
OLSR node 10.0.0.19 received HELLO message of size 40 seqNum=4
Receiving time and the originator IP of a hello message
** BEGIN dump Link Set for OLSR Node 10.0.0.19
LinkTuple(localIfaceAddr=10.0.0.19, neighborIfaceAddr=10.0.0.14, symTime=12152965283ns, asymTime=12152965283ns, expTime=18152965283ns)
LinkTuple(localIfaceAddr=10.0.0.19, neighborIfaceAddr=10.0.0.20, symTime=14107930189ns, asymTime=14107930189ns, expTime=20107930189ns)
LinkTuple(localIfaceAddr=10.0.0.19, neighborIfaceAddr=10.0.0.18, symTime=14052224970ns, asymTime=14052224970ns, expTime=20052224970ns)
LinkTuple(localIfaceAddr=10.0.0.19, neighborIfaceAddr=10.0.0.24, symTime=14213922056ns, asymTime=14213922056ns, expTime=20213922056ns)
** BEGIN dump Neighbor Set for OLSR Node 10.0.0.19
1hop Neighbors
NeighborTuple(neighborMainAddr=10.0.0.14, status=SYM, willingness=3)
NeighborTuple(neighborMainAddr=10.0.0.20, status=SYM, willingness=3)
NeighborTuple(neighborMainAddr=10.0.0.18, status=SYM, willingness=3)
NeighborTuple(neighborMainAddr=10.0.0.24, status=SYM, willingness=3)
Looking at Link Message from HELLO message: neighborType=1 (SYM_NEIGH)
Looking at Link Message from HELLO message: neighborType=2 (MPR_NEIGH)
Looking at Link Message from HELLO message: neighborType=2 (MPR_NEIGH)
2hops Neighbors
TwoHopNeighborTuple(neighborMainAddr=10.0.0.20, twoHopNeighborAddr=10.0.0.25, expirationTime=14107930189ns)
TwoHopNeighborTuple(neighborMainAddr=10.0.0.20, twoHopNeighborAddr=10.0.0.19, expirationTime=14107930189ns)
TwoHopNeighborTuple(neighborMainAddr=10.0.0.20, twoHopNeighborAddr=10.0.0.15, expirationTime=14107930189ns)
TwoHopNeighborTuple(neighborMainAddr=10.0.0.18, twoHopNeighborAddr=10.0.0.13, expirationTime=14052224970ns)
TwoHopNeighborTuple(neighborMainAddr=10.0.0.18, twoHopNeighborAddr=10.0.0.17, expirationTime=14052224970ns)
TwoHopNeighborTuple(neighborMainAddr=10.0.0.18, twoHopNeighborAddr=10.0.0.23, expirationTime=14052224970ns)
TwoHopNeighborTuple(neighborMainAddr=10.0.0.18, twoHopNeighborAddr=10.0.0.19, expirationTime=14052224970ns)
TwoHopNeighborTuple(neighborMainAddr=10.0.0.24, twoHopNeighborAddr=10.0.0.25, expirationTime=14213922056ns)
TwoHopNeighborTuple(neighborMainAddr=10.0.0.24, twoHopNeighborAddr=10.0.0.23, expirationTime=14213922056ns)
TwoHopNeighborTuple(neighborMainAddr=10.0.0.24, twoHopNeighborAddr=10.0.0.19, expirationTime=14213922056ns)
TwoHopNeighborTuple(neighborMainAddr=10.0.0.14, twoHopNeighborAddr=10.0.0.13, expirationTime=12152965283ns)
TwoHopNeighborTuple(neighborMainAddr=10.0.0.14, twoHopNeighborAddr=10.0.0.19, expirationTime=12152965283ns)
TwoHopNeighborTuple(neighborMainAddr=10.0.0.14, twoHopNeighborAddr=10.0.0.15, expirationTime=12152965283ns)
TwoHopNeighborTuple(neighborMainAddr=10.0.0.14, twoHopNeighborAddr=10.0.0.9, expirationTime=12152965283ns)
2-HOP Neighbour Set: [10.0.0.20->10.0.0.25 | 10.0.0.20->10.0.0.15 | 10.0.0.18->10.0.0.13 | 10.0.0.18->10.0.0.17 | 10.0.0.18->10.0.0.23
| 10.0.0.14->10.0.0.15 | 10.0.0.14->10.0.0.9]
MPR
MPR Selector
Step 4 iteration: 2-HOP Neighbour Set=[10.0.0.20->10.0.0.25, 10.0.0.24->10.0.0.25]
Computed MPR set for node 10.0.0.19: [10.0.0.14 | 10.0.0.18 | 10.0.0.20]
Computed MPR selector set for node 10.0.0.19: [10.0.0.14, 10.0.0.18, 10.0.0.24, 10.0.0.20]

These groups are populated with the evidences that are extracted from the logs. If an evidence belonging to the $initial-evidence-group$ is discovered, then, an in-depth diagnosis is launched so as to confirm (i.e., an evidence belonging to $confirmed-evidence-group$ is detected) or infirm (i.e., an evidence belonging to $cancel-evidence-group$ takes place) the intrusion; both lead to the termination of the diagnosis. Relying on these groups, the evolution of the attack and the related diagnosis is easily followed. In addition, their compactness facilitates the lightweight discovering of the long-terms attacks. It is worth mention that an in-depth diagnosis is launched in an independent thread, thus other detection operations are not blocked. In addition, during the in-depth diagnosis, the applied process depends on the used intrusion signature(s), i.e., the nature of the attack under diagnosis. For instance, when the intrusion signature aims at detecting the link or route spoofing attacks, then an in-depth diagnosis will include an interrogation of other node(s). While when the intrusion signature aims at detecting the drop or the broadcast storm attacks, then in-depth diagnosis performs some statistics on the routing and/or data traffic forwarded by the suspicious node. It is clear that the intrusion signatures constitute the cornerstone in diagnosis and therefore, they should be carefully established. In our IDS, we maintain the intrusion signatures in the form of conditional rules (*IF cond. THEN state.*). They are established based on the attack models previously introduced in (3.2.2). These models permit specifying the obligatory and optional preconditions as well as the consequences of an attack. By distinguishing the obligatory preconditions, i.e., those that are necessary for the effectiveness of the attack, from the optional ones, an intrusion signature would deal with unknown variations of attack that exploit similar mechanism and lead to the same consequence(s).

Overall, the mechanism for detecting attack is resumed as follows (Figure 3.12): first, OLSR logs are parsed in order to extract information about the neighbor relations, MPRs set, MPR selectors set, and topological point of view of other nodes. This information is analyzed in order to discover suspicious evidences. If an evidence with a sufficient level of suspicion is discovered then the in-depth diagnosis is launched. This diagnosis consists in matching the discovered evidences against the predefined intrusion signatures. It includes also interrogating other nodes, if necessary. The in-depth diagnosis terminates by either receiving an evidence that confirms the attack or an evidence that denies it. In the former case, an alarm message is diffused in the network as to announce the attacker identity and information about the attack. While in the latter case, the suspicion is eliminated. To have more clear idea, we further exemplify the signature establishment and the diagnostic procedure related to one active attack: the link spoofing.

3.3.4 Link Spoofing Attack

A link spoofing attack lies in falsifying *hello* message(s) so as to modify the local topology perceived by adjacent nodes. This attack influences the MPR selection. In particular, this attack has a global impact: the MPR position provides the intruder the possibility to eavesdrop, tamper, mis-relay or drop the traffic. As discussed in 3.2.2 (and further detailed in Expression 3.6), an intruder performs a link spoofing attack through one of the three following cases:

1. It advertises a non-existing and symmetric node. Thus, the intruder guaranties[33] being

[33] Unless another attacker advertises the same non-existing node.

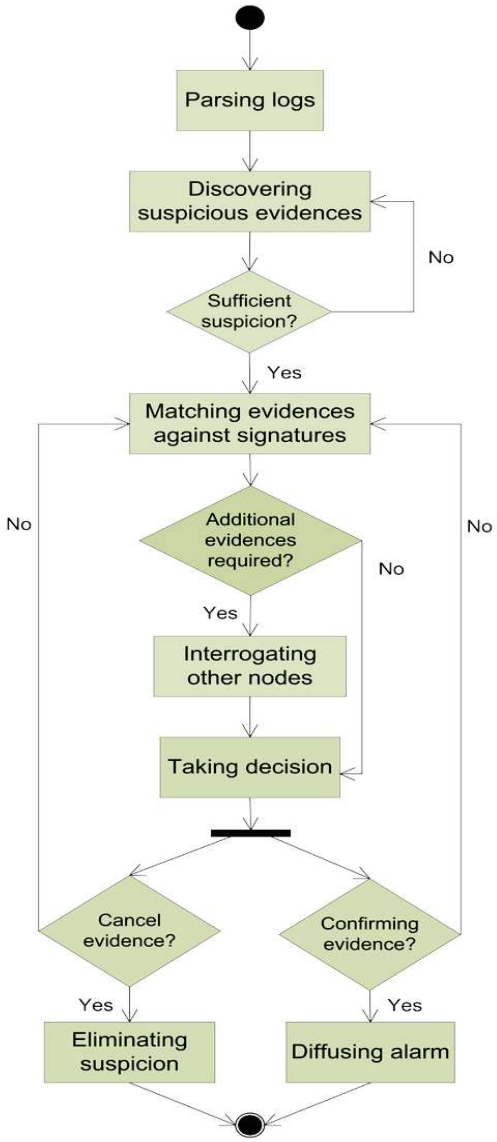


Figure 3.12: Intrusion detection in LIDR

selected as a MPR because this non-existing node is uniquely covered by the intruder,

2. It advertises existing but non-neighboring node(s). The intruder is selected as a MPR if the advertised node(s) is (are) not already covered by another (well-behaving or malicious) MPR.

3. It keeps under wraps neighboring and symmetric node(s). In this case, the connectivity of the intruder and consequently its chance of being selected as a MPR are both decreased. Used in a standalone manner, this attack aims at decreasing the connectivity of one or several nodes; a complete isolation necessitates that no other (well-behaving) MPR covers that node(s).

We develop an attack (Expression 3.14) wherein an intruder falsifies a *hello* message that contains both a non-existing node (Case 1) and existing but non-neighboring nodes (Case 2). The reason that motivates this choice is twofold. First, by advertising a non existing node N, the intruder I ensures being selected as a MPR by the victim S ($\exists N \in NS'_I \ni: N \notin \mathcal{N} \cap NS_I$). Note that with the last versions of the OLSR's RFC, a node (that is potentially an intruder) may dictate its selection as a MPR by advertising (in a *hello* message) its high willingness to relay messages. Nevertheless, previous versions (and their related implementations) ignore this case. Regardless of the version of the protocol, our intruder (as aforementioned previously) guaranties being selected as a MPR by advertising a non existing and symmetric node. Second, by announcing common neighbors with another node L (Case 2), I increases the probability that L is not selected as a MPR (I replacing L) and henceforth increases its proper ascendancy. Note that the attack we developed does not consider the third case that involves reducing the intruder's connectivity. Nevertheless, attack signature deals with it.

$$\begin{gathered} S \xrightarrow{hello(NS_S)_t} I, I \xrightarrow{hello(NS'_I)_{t'}} S, |t' - t| < \triangle t, \\ \exists N \in NS'_I \ni: N \notin \mathcal{N} \cap NS_I, \\ \exists L \in MPR_S \ni: [NS_L \backslash NS_I] \subseteq [NS'_I \backslash NS_I] \\ \Downarrow \\ I \in \mathcal{I}, L \notin MPR_S, \\ \exists I' \in \mathcal{I} \ni: I' \in MPR_S, \\ Card(NS'_I \backslash (NS'_I \cap \mathcal{N})) > 0 \\ Card((NS'_I \backslash NS_I) \cap \mathcal{N}) > 0. \end{gathered} \tag{3.14}$$

Herein, a key challenge stems from the need to generate a hand-coded intrusion signature that models such attack and henceforth permits to detect it.

Signature Establishment: As mentioned before, a link spoofing attack aims at inflecting the MPR selection; such selection is triggered upon a change in the symmetric 1- and 2-hops neighborhood. Rather than launching an in-depth diagnosis upon every change in the 1- or 2-hops symmetric neighborhood, we keep to a minimum the number of these diagnoses by initiating it only at the occurrence of an event related to a link spoofing attack. More precisely, we ignore the changes in the 1-hop neighborhood (e.g., apparition of 1-hop neighbor) because they are observed by the node itself. Thus, they are not subject to the remote falsification which is the cornerstone of a link spoofing attack. In contrary, changes in the 2-hops neighborhood are considered as long as they impact the MPR selection. In practice, the evidence that reveals a link spoofing attack (Table 3.4) are broken down into:

- a MPR replacement (Evidence 1 or $E1$ for short) that results from a change in the covering of the 1-hop neighbors; one (or several) 1-hop neighbor(s) (possibly the replacing MPR) increase(s) it (their) coverage to the detriment of the replaced MPR[34].

[34] A replace MPR is a 1-hop neighbor that is excluded from the MPRs set even though it provides the same connectivity in both the previous and current MPR selection round.

- No MPR replacement takes place but an already selected MPR is detected as misbehaving node. For instance, a misbehaving MPR may drop, falsify or mis-relay the control messages ($E2$). Note that an evidence of a special interest herein refers to a node promoting itself as a MPR without having been selected as a MPR. A misbehaving MPR includes also the case wherein a MPR continually advertises the same 2-hops neighbors set despite it is no more valid. Contrary to other cases, this one is not event-driven and should be handled based on random or periodical checks.
- a MPR is the only one that covers one or several nodes ($E3$).
- a MPR covers partially its adjacent neighbor(s) ($E4$).
- a MPR provides connectivity to a non-neighboring node ($E5$).

The occurrence of either $E1$ or $E2$ constitutes the starting point of an in-depth diagnosis. $E1$ and $E2$ belong to the *initial-evidence-group*. The act of being the only MPR that provides the connectivity to node(s) ($E3$) is suspicious but is not sufficient to launch an in-depth diagnosis because this situation is typical in a sparse network. Furthermore, two nodes within the covering range of each other often fail in communicating due to the unpredictable nature of wireless transmission resulting from, e.g., obstacles, noises. Thus, diagnosing $E3$ is especially difficult under no specific assumption. Overall, the occurrence of $E1$ or $E2$ and optionally $E3$ leads to an in-depth diagnosis (Expression 3.15). In practice, the 1-hop neighbor(s)[35] of the suspicious MPR are interrogated. Note that, part of the interrogated nodes may express a different opinion that results from the malicious nodes or an obsolete routing information. This calls for taking into account their respective reputation as we will see in the next chapter.

$$\begin{array}{cc} E1 \vee E2, & optional(E3) \\ \Downarrow & \Downarrow \\ (E4 \vee E5\ ) & (!E4\ \wedge\ !E5) \\ \Downarrow & \Downarrow \\ \text{The suspicious MPR} & \text{The suspicious MPR} \\ \text{is an intruder.} & \text{is well-behaving.} \end{array} \tag{3.15}$$

If the obtained answers confirm (resp. deny) that the suspicious MPR covers partially its neighbors ($E4$) and/or advertises a distant node as 1-hop neighbor ($E5$), then the MPR is declared as an intruder (resp. well-behaving) and the diagnosis is terminated. Obviously, the cooperation between the nodes constitutes a cornerstone in our in-depth diagnosis.

Diagnosis. The in-depth diagnosis of a link spoofing attack consists in verifying the existence of a symmetric and neighboring relationship between a suspicious MPR and its advertised 1-hop neighbors (Algorithm 1). More precisely, this diagnosis follows the following steps. First, the MPR that have been replaced by another MPR (or a 1-hop neighbor) are identified (lines 1-2) because such replacement represents the initial evidence of a link spoofing attack ($E1$ in

[35] These neighbors are announced in the neighbor advertised by the suspicious MPR.

Table 3.4: Evidences characterizing a link spoofing attack

E1	a MPR is replaced by a 1-hop neighbor or by an already selected MPR
E2	a MPR behaves maliciously i.e., it tampers, mis-relays or drops the control messages.
E3	a MPR is the only node that provides connectivity to one (or more) 2-hops neighbor(s).
E4	a MPR advertises a partial set of 1-hop neighbors.
E5	a MPR advertises non-neighboring node(s) as 1-hop neighbors(s).
E6	a MPR is defined as an intruder/well-behaving node.

Algorithm 1 :In-depth diagnosis

```
1: OldMprs = GetReplaced-Mpr();
2: SuspiciousMprs= GetReplacing-Mpr();
3: for (suspicious ∈ SuspiciousMprs) do
4:    InterrogatedNodes = GetCommon2HopsNeighors (suspicious,OldMprs);
5:    for (2HopsNeighbor ∈ InterrogatedNodes) do
6:       if (LinkExistence(2HopsNeighbor, suspicious) == false) then
7:          Generate-Alarm(suspicious);
8:       end if
9:    end for
10:   Cancel-Suspicious(suspicious);
11: end for
```

Table 3.4). For this purpose, the function *GetReplaced-Mpr* is called (Line 1). It establishes the MPR that have been replaced by comparing the current MPRs and the 1-hop neighbors that were MPRs in the last selection of MPRs. Then, the MPRs sharing 1-hop neighbor(s) with a replaced MPR, are identified (function *GetReplacing-MPR*, line 2). Those identified MPRs are tagged as suspicious and are subject to further analysis (lines 3-11); the objective is to verify whether some spoofed links have been advertised so as to replace the MPR. It follows that the 1-hop neighbors that are common to a replaced MPR and a suspicious one are determined (*GetCommon2HopsNeighors* function, line 4). Note that these 1-hop neighbors also correspond to the 2-hops neighbors of the node launching the diagnosis. They are interrogated (*LinkExistence* function) so as to verify the existence of links. The interrogation of a 2-hops neighbor, denoted Ai, consists in sending a request to Ai asking if this latter considers the suspicious MPR as a 1-hop neighbor at a specific time t. Whenever possible, this request is sent

without going through both the suspicious MPR I or any colluding intruder I'. This eluding is necessary in order to prevent I and I' from dropping the request and/or forging a deceptive answer. Therefore, the 1-hop neighbor(s) (primarily the MPR(s)) that covers the interrogated 2-hops neighbors is provided[36] the request. If no answer is obtained (i.e., the related time-out elapses), then the request is sequentially transferred through the rest of the covering 1-hop neighbors (keeping in mind that, as aforementioned, MPRs are primarily selected). However, when no neighbor is left, then a (multi-hops) alternative path is searched for in the routing table to reach Ai. Note that the verification related to a suspicious MPR is performed within an independent thread hence, the diagnosis of one node (and the result waiting) is not blocking to others. If Ai denies being a 1-hop neighbor of the suspicious MPR, then a countermeasure is triggered. More precisely, an alarm that establishes the detected attacker(s) is broadcasted (*Generate-Alarm* function, line 7). Otherwise, if no deny takes place, the suspicious MPR is no longer suspected (*Cancel-Suspicious* function, line 10). In both case, the diagnosis terminates. Note that if no answer is provided[37], then the suspicious MPR is tagged as not verified and the degree of suspicion in this latter goes up.

3.4 Performance Evaluation

In order to evaluate the performance of our IDS, a mobile *ad hoc* network has been simulated using the network simulator NS3[38] [133]. Each node in the simulated network is further coupled with a LinuX Container (LXC) virtual machine[39] [134]. The reason that motivates this coupling is twofold. First, NS3 offers the possibility to simulate a large-scale mobile *ad hoc* network. Second, LXC, an operating-system-level virtualization tool, permits to run multiple isolated machines (also called *containers*) on a single modified hosting kernel[40]. Each container has its own resources (e.g., process tree, network interface, IP address). Thus, the resource consumption (e.g., memory usage) can be isolated and measured. In practice, an instance of the IDS is installed on each container that appears as a standalone/separated machine. Figure 3.13 exemplifies the coupling between nodes in NS3 and LXC containers. From the IDS perspective (as well as from any application installed on the container), the emission and reception of packets is done through the network interface (*eth*0), while the container contains a simulated ethernet card (*veth*), which is connected through an *ethernet bridge* towards a kernel tap device (*tap*)[41]. The *ethernet bridge* implements the forwarding of link-level frames. The interface *tapRouter* [135] is implemented in NS3 and used to exchange packets between the *tap* device and the *WiFi* device, which enables NS3 nodes to communicate over an IEEE 802.11- and OLSR-enabled MANET. Overall, the outgoing packets from *eth*0 in the container are delivered to the *WiFi* device at the corresponding node in the simulated network and vice versa. Simulation is carried out using

[36] The suspecting node S is aware of the connectivity between the nodes that form its two hops neighbors. Thus, S may select the one (if it exists) that covers Ai.

[37] Inconsistent answers lead also to a claim in the degree of suspicion as we will see in the next chapter.

[38] http://www.nsnam.org

[39] http://lxc.sourceforge.net

[40] Up to 1024 containers over a single hosting kernel.

[41] Note that for each container-node coupling, new *tap* and *ethernet bridge* should be defined in the hosting kernel.

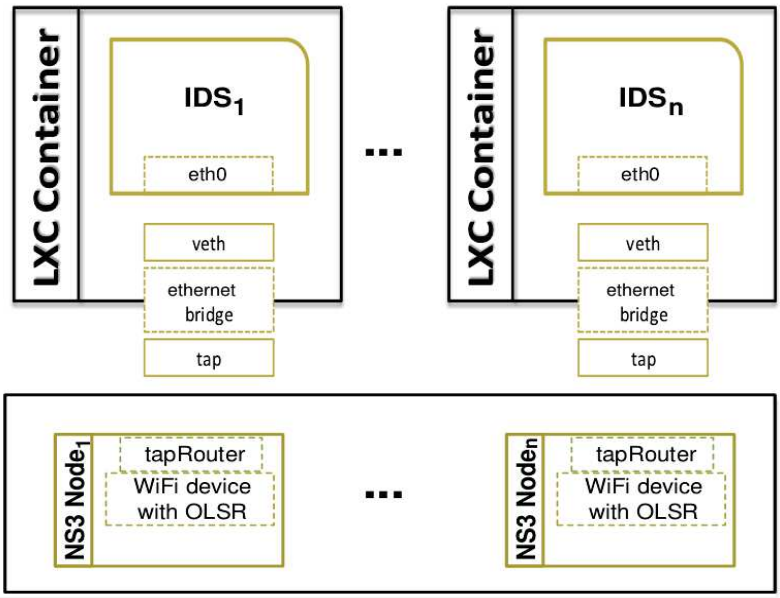


Figure 3.13: Experiment platform

the aforementioned platform (Figure 3.14), wherein the hosting device is equipped with 32GB of memory and 2 Intel(R) Xeon(R) (6) Core @2.40GHz CPUs. The containers hold a Fedora 12 operating system.

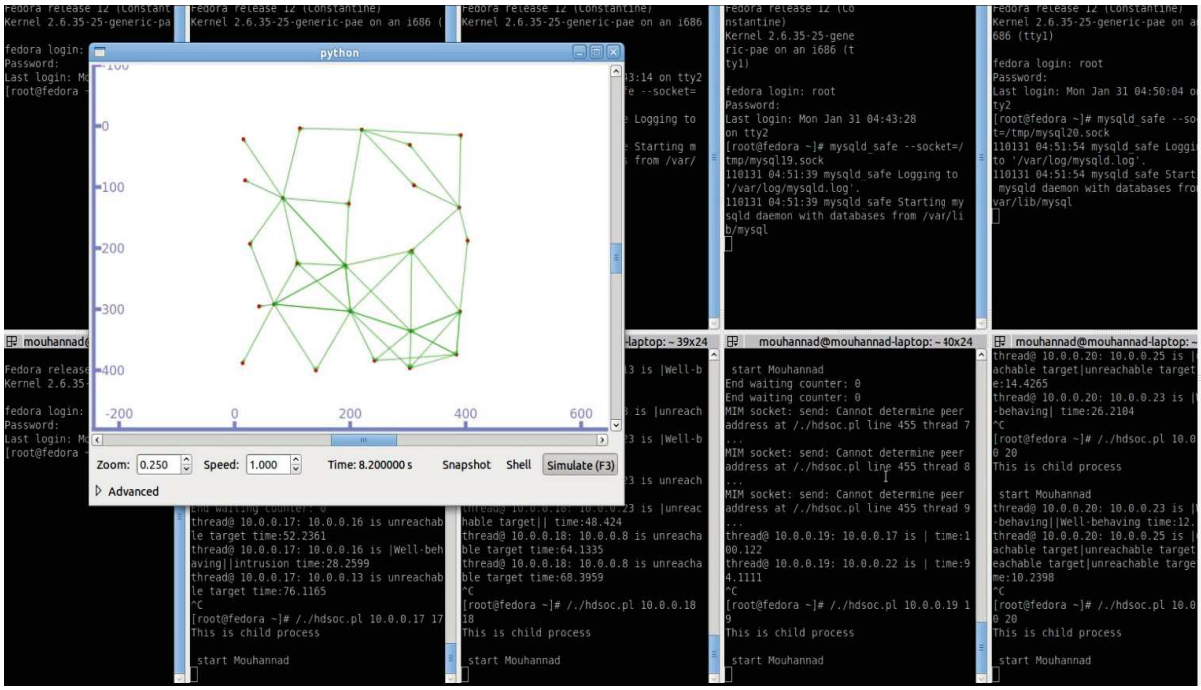


Figure 3.14: A snapshot of the evaluation environment

We consider a MANET composed of $N = 30$ nodes split into 25 well-behaving nodes and 5

Table 3.5: Simulation parameters

Simulator	Ns3 + Lxc virtual machines
Container operating system	Fedora 12
Hosting device memory	32GB
Hosting device CPU	2 Intel(R) Xeon(R) (6) Core @2.40GHz
Number of mobile nodes	30
Number of intruders	5
Topology	$310 \times 310\ m^2$
Wireless physical layer protocol	IEEE 802.11b
Transmission range	90m
Maximum bandwidth	1Mbps
Traffic	*V4PingHelper*: 56 bytes Icmp packet/node/s
Mobility model	*RandomWalk2d*
Maximum speed	8 m/s (or 28.8 km/h)
Simulation time	140 s
Hello message interval	2 s
Tc message interval	5 s

intruders[42]. Intruders launch repeatedly the implemented link spoofing attack[43] (as described in Expression 3.14). Nodes communicate via Ieee 802.11b and are characterized by a transmission range T_x of $90m$. Note that the previous communication parameters are inspired by a study on *ad hoc* network [136]. Nodes randomly move, following the so-called *RandomWalk2d* [137] mobility pattern provided by Ns3: each node moves according to a randomly chosen direction at a given speed that is the same for all the nodes. When a node hits the network boundaries (unless specified otherwise, the network area is defined by a square area of $S = 310 \times 310\ m^2$), it rebounds following a reflexive angle. Data traffic is further modeled using the *V4PingHelper* application in Ns3: each node exchanges 56 bytes Icmp[44] echo requests to one another and waits for $1s$ before sending it again. We use Olsr as the underlying routing protocol, preserving the configuration parameters promoted in the RFC 3626 (e.g., *hello* message interval is $2s$). For each experiment, the simulation is launched 5 times, each lasting for $140s$. Then, we present the maximal, the minimal, and the average values.

We aim at evaluating the performance of our system in terms of:

- Intrusion detection rate that reflects the capacity of detecting successful intrusions[45],

[42] The objective of having 5 intruders, and not only one, is to test whether our system is able to handle several intruders at the same time.

[43] In our experiments, the number of successful intrusions varies between 15 and 33 because the intruder's capacity for launching the attacks depends on its position, and thus this capacity varies according to the applied mobility and density.

[44] www.ietf.org/rfc/rfc792.txt

[45] A successful intrusion causes the replacement of a legitimate Mpr of the victim by an intruder.

- False positive rate that measures how long a legitimate node is wrongly designated as an intruder,
- Detection overhead which represents the additional network traffic that is generated because of intrusion detection. Note that further benchmarks are also provided in terms of memory usage.

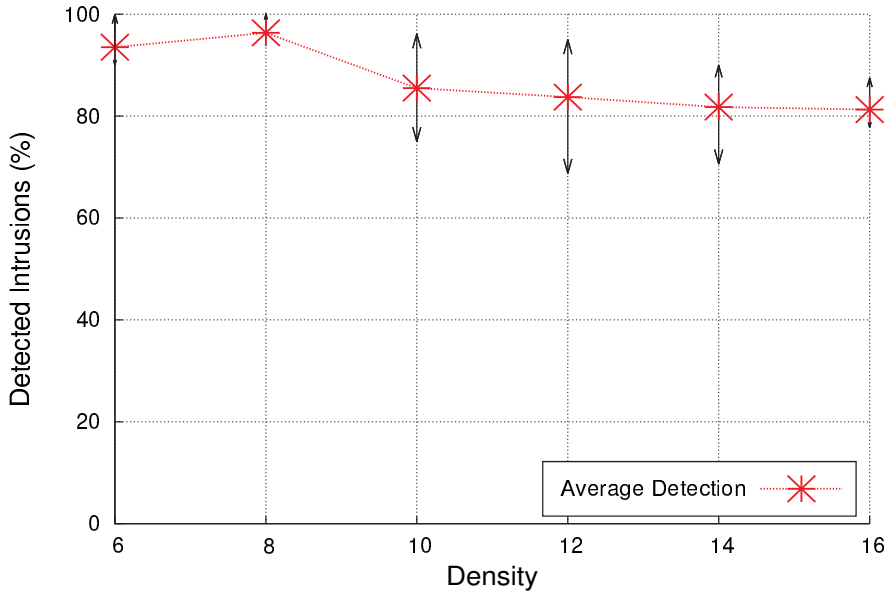
Based on those performance indicators, we evaluate the performance of our IDS.

Network Density. In order to evaluate the scaling properties of our system, we vary the density of the network. This density corresponds to the average number of neighbors, which is defined in [138] as follows:

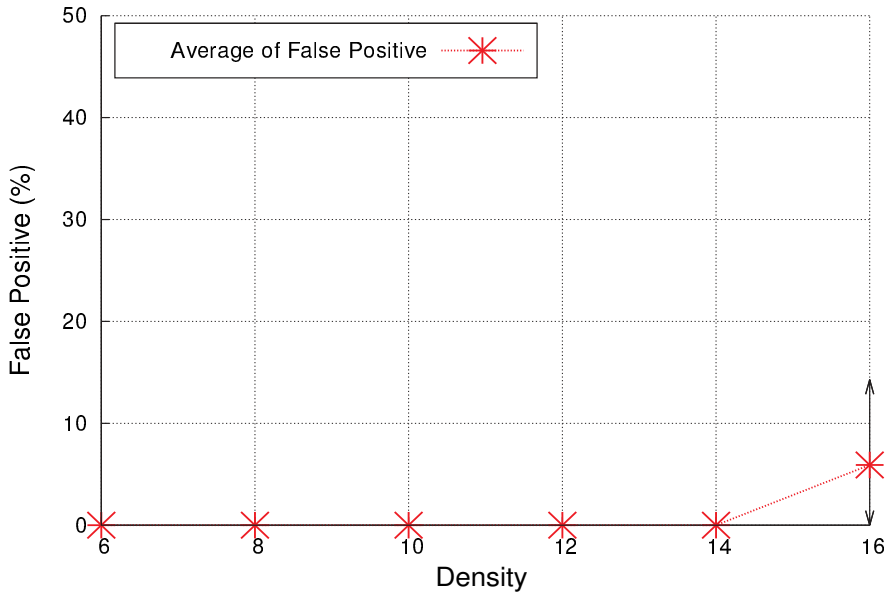
$$Density = \frac{N.\pi.T_x^2}{S} \tag{3.16}$$

Where N is the number of nodes, T_x is the transmission range of the node, and S is the area where the nodes are moving. By varying the area, we obtain different network densities. Indeed, we selected an interval of density that ranges from 6 up to 16 neighbors, which includes the intruder, the victim and the remaining neighbors. Note that having 16 neighbors represents a high density that is difficult to be reached in reality. Figure 3.15(a) shows that the detection rate keeps higher than 80% regardless of the network density. This demonstrates the relatively limited impact of the network density on the performance of our IDS. More precisely, the intrusion detection rate slightly rises from 93.5% up to its maximum 96.3% when the density varies from 6 up to 8 neighbors. Then, a slow diminution is observed: dense networks (with a density exceeding 8) are characterized by a high collision rate, which increases the amount of dropped packets including the diagnosis-related ones. Hence, the detection rate declines. Similarly, Figure 3.15(b) highlights that the average percentage of false positive is neglectful (under 5.9%). The analysis of several cases of false positive leads us to find that a false positive occurs when two nodes own different live-times for the bidirectional link that connects them. Consequently, one of the two nodes still confirms the existence of this link while the other one denies it. To override this problem, the period during which those links are considered valid should be amplified as proposed in [1]. Figure 3.16(a) presents the average increase of the memory usage caused by intrusion detection. This usage fluctuates between 17.6MB and 22.2MB. It is also worthy to mention that contrary to OLSR, our system does not assign additional functionalities to MPR. In fact, each node may launch the detection, thus our IDS provides an homogeneous load among the nodes. As illustrated in Figure 3.16(b), the traffic caused by detection-related communications is very low compared to OLSR traffic (under 0.5% for a network density inferior to 8). It then slowly rises to reach 1.3% when the network density is 16. In order to understand the slight fall when the density reaches 14, we compare the average number of suspicious cases with regards to a varying density. This average was 70.8 (resp. 65.8) with density equals to 12 (resp. 14). Consequently, the number of diagnosis-related packets is smaller with density equals to 14 than with density equals to 12. Even though more neighboring links should be verified in the dense network, there is a limited rising in the consumption of the resources, either memory usage or traffic overhead. This results from the fact that in the dense network each node has more neighborhood relations and hence, more information about the network topology. Thus,

interrogating few nodes would be sufficient to gather the required information about the links of the suspicious MPR. Still, a higher network density leads to more collision and consequently more dropped diagnosis-related packets.



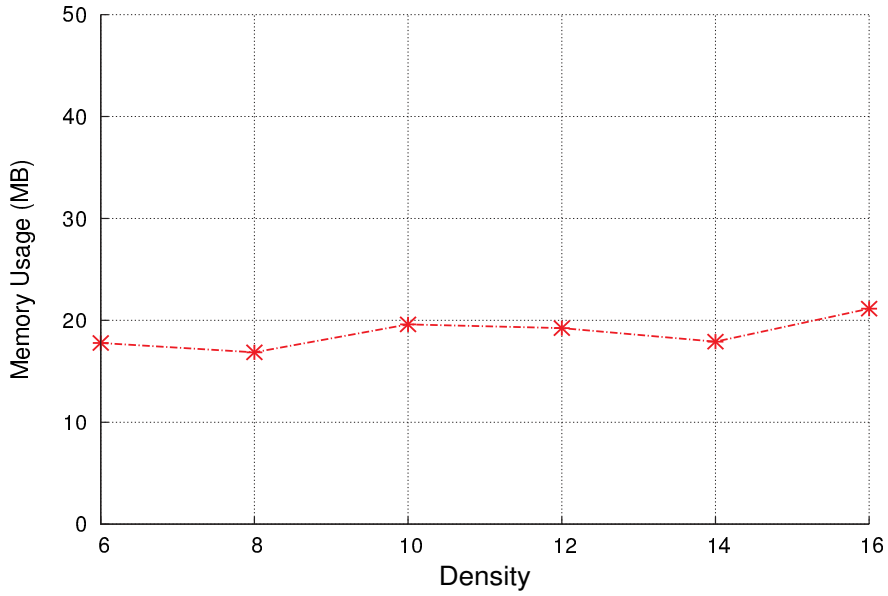
(a) Detection rate



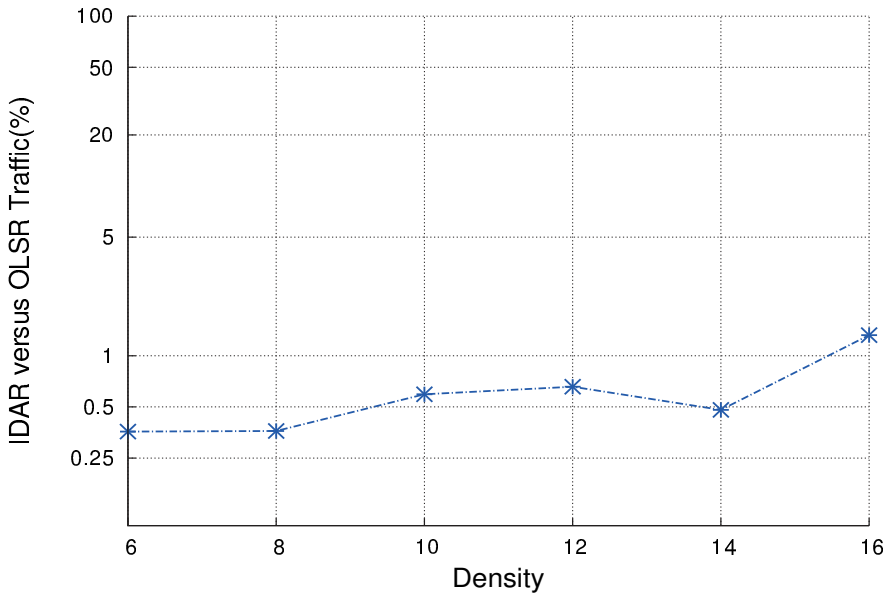
(b) Detection rate

Figure 3.15: Intrusion detection rate (a), and False positive rate (b) depending on the network density

Mobility In order to isolate the influence of the mobility, the network density is frozen to 8. Then, we launch the simulation with a gradual moving speed ranging from 0 m/s to 8 m/s (i.e.,



(a) Memory usage

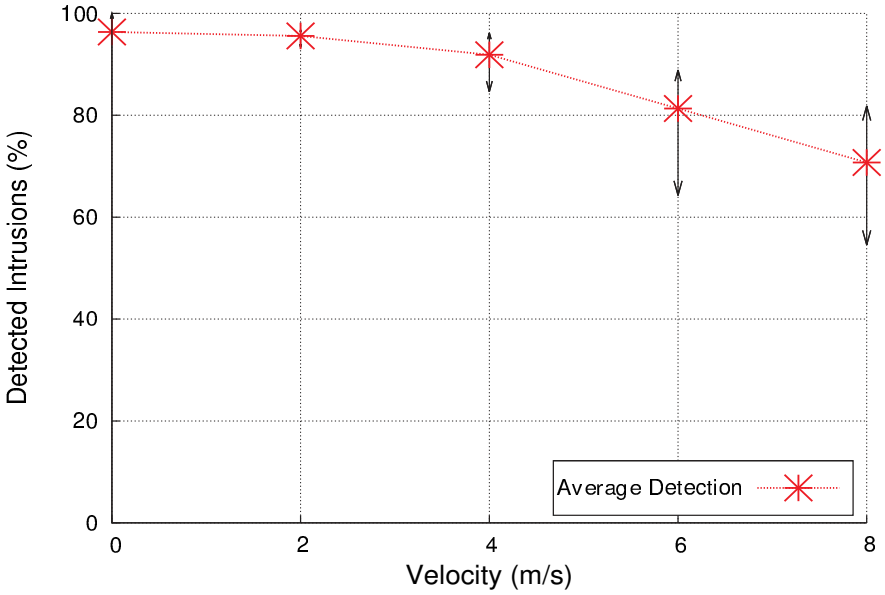


(b) Traffic

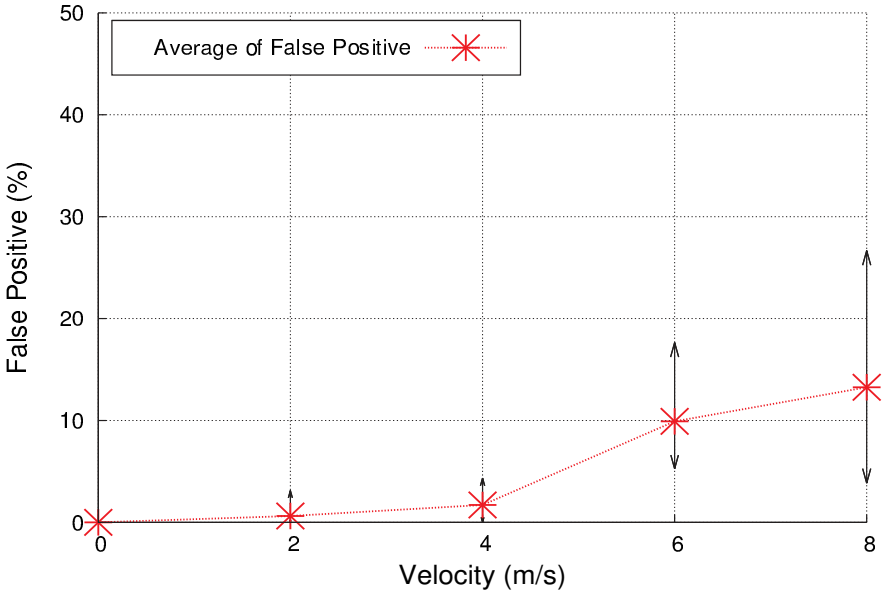
Figure 3.16: Memory usage (a), and Traffic (b) depending on the network density

around 28.7 kilometers/hour equivalent to 17.89 miles/hour). Note that increasing the moving speed above this value makes us dealing with a Vehicular *ad hoc* NETwork (VANET for short) [139], where the routing protocols of MANET are not feasible or do not provide the optimum performance [140]. The detection rate falls when the speed increases (Figure 3.17(a)). This results from the ever changing topology that results in increasing the number of broken links and the related drops of diagnosis-related packets. However, the proposed IDS still provides a high detection rate, e.g, the average of detection rate is 70.7% with a moving speed equals to

$8m/s$. Note that even though the detection rate decrease along with the moving speed (i.e., less diagnoses conclude by confirming/canceling the suspicion), almost all the launched attacks are listed as suspicious actions. Moreover, every attacker has been detected at minimum one time in the majority of simulation rounds. Figure 3.17(b) shows that the higher the moving speed, the greater number of false positives. This results because more out of date feedback



(a) Detection rate



(b) False positive rate

Figure 3.17: Intrusion detection rate (a), and False positive rate (b) depending on the network mobility

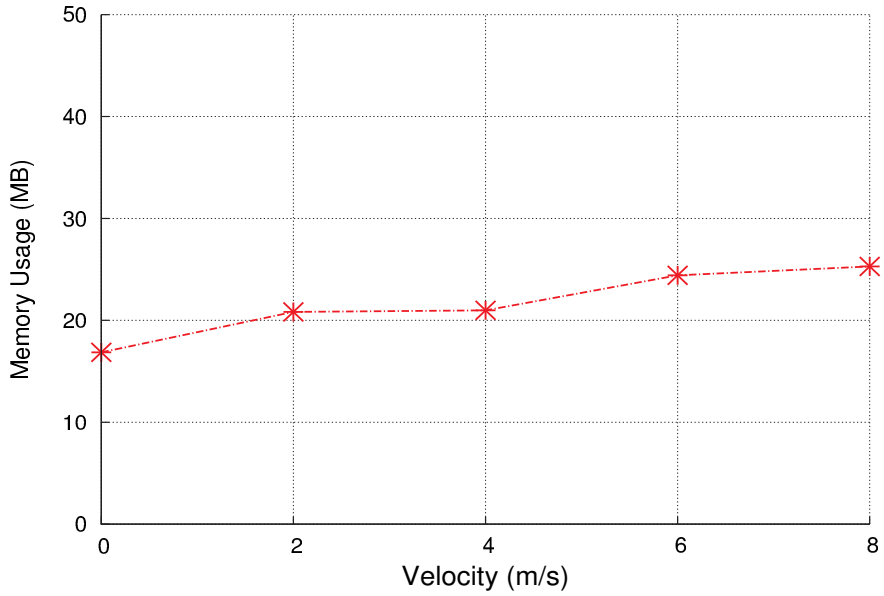
(i.e., diagnostic answers) are used in the diagnosis. Anyway, this percentage remains less than

12% even with the high mobility. The memory usage (Figure 3.18(a)) rises staidly from 17.6MB to 26.5MB, which is reasonable even for resource-limited devices. The impact of the mobility on the bandwidth is shown in Figure 3.18(b). The IDS traffic comparing to the OLSR traffic significantly grows from $0,36\%$ up to 3.1% for speed ranging from $0m/s$ up to $2m/s$. Then, the bandwidth usage gradually increases and reaches almost 6% when the moving speed is equal to $8m/s$. This increase comes from the fact that more diagnosis-related requests/answers are sent several times due to packets lost. However, this overhead is still negligible compared to the one generated by the OLSR protocol.

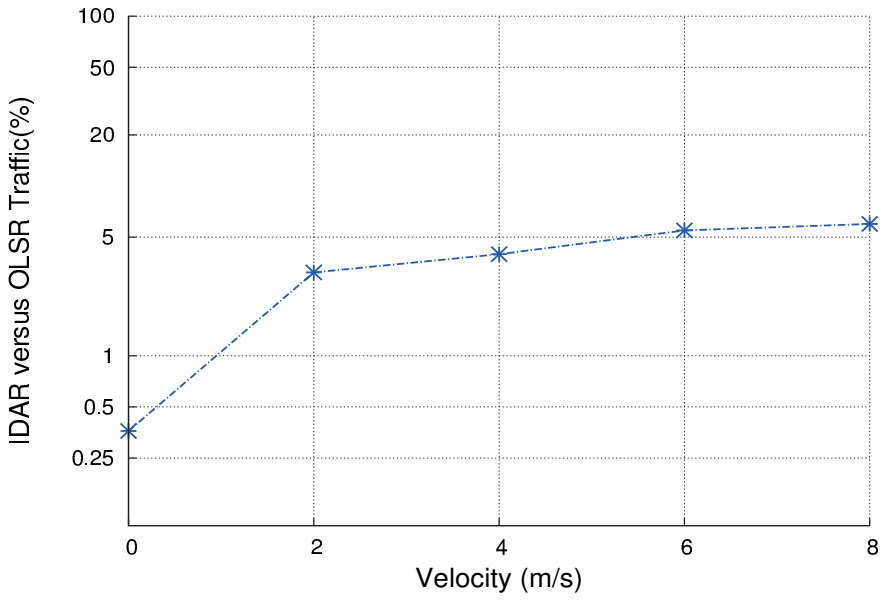
Discussion. It is clear that the density of the network holds less influence than the mobility on our IDS. More mobility in MANET causes more dropped diagnostic packets and hence less diagnosis concludes with a final result, i.e., confirming or refuting the suspicious. However, comparing the launched attacks to the discovered suspicious events shows that the majority of attacks are tagged as suspicious events and considered for further diagnosis. Hence, coupling our system with a mechanism of credibility/trust will lead, in worst case, to decline the trustworthiness of the intruders. Note that the mobility has a negative influence on all the IDSs especially the cooperative ones where there is an exchange of evidences/alerts. Table **??** provides a short performance comparison between our system and other IDSs that are presented in the previous chapter. Non of the IDSs, including ours, discussed the impact of *blackmail* attacks whereas the majority of those IDSs are cooperative; i.e, nodes exchange the evidences and alerts so as to enhance the detection accuracy. But this cooperation could be exploited by the malicious nodes as to falsify the evidences and alerts. More precisely, they may provide false accusations (resp. praises) so as to conclude that a legitimate (resp. a malicious) node is an intruder (resp. a benign node). In our IDS, detecting the falsified link(s) of a suspicious MPR depends largely on the proper cooperation of MPR. A malicious node and/or possibly colluding node may foil the detection by providing incorrect information about the suspicious MPR. Furthermore, when the interrogation messages are uniquely passed/provided through a malicious node, this latter can prevent the conclusion of the diagnosis by simply dropping the interrogation requests. Such dropping interrogation-related messages can take place not only because of the malicious nodes but also due to the absence of routes or collision, which is very common in MANET. Therefore, there is a need for:

- evaluating the correctness level of the returned feedback/answers so as to decide whether they should be considered or not during the diagnosis,
- providing the capability of concluding the diagnosis even though the gathered evidences are not complete.

It is worth to mention that gathering and processing all the evidences, especially in the dense networks, leads to consume a lot of the available bandwidth, as well as increasing nodes' activities. High consumption of the bandwidth might cause bottlenecks in communication, whilst the increase in the node's activities does not help in maintaining its resources. Therefore, limiting the diagnosis on the minimum number of evidences, which can provide the required level of detection accuracy, helps in providing a lightweight detection.



(a) Increase in memory usage



(b) Traffic overhead

Figure 3.18: Memory usage (a), and Traffic (b) depending on the network mobility

3.5 Summary

The attacks targeting *ad hoc* routing protocols and more specifically the OLSR protocol pertain to three classes: dropping, modify and forward, and active forge attacks. Drop attacks aim to drop completely or selectively the control messages, thus preventing routes building/maintaining in the network. The modify and forward attacks consist in maliciously modifying critical fields/-contents in the control messages (e.g. sequence number, source address) before forwarding them.

They also include delaying/repeating the emission of the control message(s), so that the routing tables of other nodes are updated based on obsolete information. The active forge attack, the malicious node takes the initiative and forges novel deceptive control messages in order to either exhaust the resources in the network, or poison the neighborhood and topological point of view of others. More precisely, by introducing incorrect neighbor set and/or willingness in the HELLO messages, the malicious node prevents its neighbors from correctly specifying their 1- and 2-hops neighbors, and further selecting their MPRs. While advertising non existing (resp. hiding existing) MPR selectors in the TC messages leads to adding incorrect (resp. eliminating valid) routes in the network. In order to facilitate the definition of intrusion/attack signatures, we extend a description model - an attack is expressed as the preconditions and the resulting consequences - and enrich it with temporal annotations. An intrusion signature takes the highest abstracted form so as to circumvent all possible deviations of the intrusion. Once hand coded, these signatures are utilized by our IDS; a log-based, distributed and cooperative intrusion detection system dedicated to operate in mobile *ad hoc* networks. The proposed system distinguishes itself by analyzing the logs generated by the routing protocol, instead of sniffing the traffic, in order to extract intrusion evidences. These latter are then compared against the predefined intrusion signatures during intrusion diagnostic. Such diagnosis searches for more evidences for confirming/refuting attack's occurrence. It may include the interrogation of other nodes or doing some statistics about the traffic of the suspicious node(s). In order to minimize the overhead resulting from the diagnosis, this latter should be carefully planned, i.e., it is activated only when there is a sufficient degree of suspicion about the attack, and is deactivated when a diagnostic result is concluded. For this purpose, discovered evidences are categorized into four groups according to their degree of suspicion/gravity:

- *Initial-evidence-group* contains the evidences that lead to activate a diagnosis over the network,
- *Suspicious-evidence-group* contains the evidences that reinforce the suspicion but they cannot, lonely, activate the diagnosis,
- *Confirmed-evidence-group* contains the evidences that terminate the diagnosis by confirming the occurrence of an attack,
- *Cancel-evidence-group* contains the evidences that eliminate the suspicion and terminate the diagnosis by declaring the suspicious node(s) as legitimate(s).

We further develop a link spoofing attack on the OLSR protocol, build the related detection signatures/rules, and evaluate the performance of our IDS relying on the NS3 simulator coupled with LXC virtual machines. This evaluation environment enables measuring the detection accuracy, as well as the amount of consumed resources. Comparing the performance of our system with other IDSs shows that the former has one of the highest detection rate. It also generates an acceptable number of false positive. Moreover, our system is one of the rare IDSs that takes into account the consumption of resources as a critical factor during the evaluation. In our IDS, the traffic overhead and the increase in the memory usage that result due to the detection operations are low and could be tailored by the resources-constrained devices.
It is worth to mention that our IDS can be easily adapted to detect attacks against other *ad*

hoc routing protocols (e.g., AODV) (or even the attacks against other layers such as application and data link layers in MANET). For this purpose, there is a need to: (i) modify the parser so as to suit the logs of AODV, and (ii) add the signatures corresponding to the attacks against AODV protocol [46]. The remaining operations, i.e., interrogating other nodes, matching evidences against the intrusion signatures and the countermeasure, keep unchanged.

Even though LIDR presents a high detection accuracy and a remarkable maintenance of resources, it still needs to override some challenges. Otherwise, its performance could be degraded. In fact, existing a large number of evidences to tackle during the diagnosis leads to a high resource consumption. Moreover, malicious node(s) may foil the detection by fabricating incorrect evidences. In the next chapter, we tackle these challenges by coupling our system with a trust mechanism that serves in evaluating the gathered evidences before using them in the detection. In addition, we propose the confidence interval as a measure that helps in finding a trade off between detection accuracy and resource consumption.

[46] We have already started defining and modeling the attacks against AODV protocol.

Table 3.6: Comparing LIDR performance with other IDSs

LIDR	OLSR	30	16.6%	0-8 m/s	70.7% – 96.3%	0% – 13.2%	17.6 - 26.5 MB	47.8 - 621.5KB (0.36% – 5.97% of OLSR traffic)
[108]	OLSR	30	3.3%, 6.6%, 10%	-	-	-	-	6% – 12% of OLSR traffic
[1]	OLSR	10	10%	0 m/s	Detection is possible	-	-	-
[5]	OLSR	7	14.3%	0 m/s	Detection is possible	-	-	-
[115]	OLSR	5	20%	0 m/s	Detection is possible	-	-	-
[6]	OLSR	11	9%	0 m/s	Detection is possible	-	-	-
[71]	AODV	30	3.3%	0-5 m/s	30.67% – 90%	0% – 20%	-	-
[81]	AODV	50	2%	0-20 m/s	90%	5%	-	-
[83]	AODV	30	3.3%	1-20 m/s	70% – 82%	12% – 18%	-	-
[100]	AODV	20, 50	10%	1-10 /ms	96.4% – 98.7%	0.79% – 0.93%	-	900kB
[107]	AODV	52	3.8%	0-20 m/s (Fixed IDSs)	100%	0% – 0.6%	-	-
[109]	AODV	5, 21	20%, 23.8%	0-10 m/s	50% – 100%	0% – 30%	-	-
[110]	AODV	30	-	0-20 m/s	70% – 100%	0.5% – 15%	-	-
[76]	AODV	50	Random	0-20 m/s	$79 \pm 10\% - 92 \pm 3\%$	$5 \pm 1\% - 32 \pm 8\%$	-	-
[4]	AODV	10,20,50	-	0-10 m/s	0% – 100%	0%– $\gg$ 100%	0.2% – 0.6%	-
[114]	AODV	20-200	-	0-20 m/s	30% – 95%	0% – 16%	-	-
[116]	AODV	50	-	1-20 m/s	99.8% – 100%	0.83% – 8.46%	-	-
[79]	AODV DSDV DSR	- - -	- - -	- - -	$88.48 \pm 4.14\% - 97.1 \pm 0.32\%$ $85.23 \pm 3.28\% - 90.61 \pm 2.99\%$ $85.2 \pm 2.38\% - 99.1 \pm 0.37\%$	$1.45 \pm 0.72\% - 20.2 \pm 6.27\%$ $5.37 \pm 3.10\% - 26.3 \pm 5.49\%$ $0.03 \pm 0.04\% - 15.3 \pm 4.08\%$	- - -	- - -
[72]	DSR	50	40%	0-20 m/s	-	-	-	11% – 31.3% of data traffic
[91]	DSR	30	3.3%	3-5 m/s	60% – 100%	2.5% – 50%	-	-
[88]	DSR	30	-	-	75%	1.2%	-	-
[3]	DSR	50	2%	0-20 m/s	83.7% – 97.4%	1.3% – 7.2%	-	-

CHAPTER 4

ROBUST AND LIGHTWEIGHT DETECTION

Contents

4.1 Introduction

Our IDS needs to face two threats. First, the *blackmail* attack that consists in a malicious node which sends incorrect evidence(s) so as to foil the detection. For instance, during the verification of the links that are advertised by a suspicious MPR, a colluding node may confirm the existence of falsified link(s) so that this MPR is not discovered. A malicious node may also accuse a legitimate MPR by denying the correctness of the advertised link(s). In both cases, the diagnosis may end with an incorrect result. Second, the evidences may be contradictory because legitimate nodes own different visions of the topology. Such difference is due to the mobility or to the obsolete routing information. Thus, there is a need for specifying to what extend a diagnosis is reliable. Moreover, gathering many evidences implies (i) a high consumption of resources and (ii) the dropping of some messages. This requires to find a trade-off between the detection reliability and the amount of the evidences that has to be gathered. To override the aforementioned challenges, we propose to:

- Couple our IDS with a trust system that helps in reducing the impact of the *blackmail* attacks.

- Employ the confidence interval as a measure of the detection reliability and also as a mean for determining whether additional evidences should be gathered to provide a reliable diagnosis.

More precisely, we propose an entropy-based trust model that specifies whether a node is trustful according to its historical participation to the intrusion detection. In practice, every time a node provides a correct (resp. an incorrect) evidence during a diagnosis, its trustworthiness is increased (resp. decreased). Here, self-observed evidences (also referred as first hand, or direct-observations) are uniquely used in estimating the trust. Recommendations coming from other nodes (also called second hand, indirect-observations) are not considered for three reasons. First, they are subject to falsifications. This renders the trust model vulnerable to false accusations and/or false praises. Second, exchanging such recommendations imposes more traffic overhead. Third, they are subject to the point of view of the recommender. During the diagnosis, every gathered evidence is first pondered according to the trustworthiness of its source. Then, it is merged with other pondered answers coming from other nodes. If the combination of the gathered evidences confirms the attack, then the trustworthiness of every node that provided an answer confirming (resp. denying) the existence of the attack is increased (resp. decreased) and vice versa. In the proposed trust model, the increase (resp. decrease) of a node's trustworthiness is associated with the role that this node plays in protecting (resp. harming) the network. More precisely, the higher the risk of a detected attack the greater the increase (resp. decrease) of the trustworthiness of the nodes that participate in detecting (resp. colluding) this attack. Thus, assessing the risk of an attack is based on two factors: the level of evolution of this attack and the vulnerabilities (or the attacks) that may be derived from it. The evolution level represents to which extend the attacker is far from achieving its attack: the larger the evolution of the attack, the greater the risk. The derived vulnerabilities represent the possibility to launch other, usually more serious, attacks based on this attack: the more derived vulnerabilities, the higher the risk of the attack. The aforementioned factors of risk are estimated based on an attack tree.

Moreover, we propose to use the confidence interval of a diagnosis so as to find a trade-off between the resources consumption and the detection reliability. Thus, our IDS can decide when to stop (or at minimum reduce) the gathering of evidences. This also helps in deciding whether a countermeasure should be launched or not. In fact, such decision is critical since punishing a legitimate node (resp. permitting an intruder to continue misbehaving), decreases the performance of MANET and may partition it[47] [57]. The confidence interval is a range of values that contains, with a specific level of certainty, the true value of the diagnostic result. During the diagnosis, if all the values in the corresponding confidence interval confirm (resp. affirm) the existence of an attack, then the true diagnostic result does the same (i.e. it confirms or affirms the existence of an attack). This means that the obtained diagnostic result, which belongs to the confidence interval, is sufficiently reliable and thus, no more evidences are needed. Otherwise, i.e., the confidence interval contains contradictory values, the required reliability is

[47] For instance, excluding a legitimate node due to an unreliable detection result leads to divide the network into two sub-networks when the excluded node is the unique connector between these two sub-networks.

not yet achieved. Thus, further evidences should be gathered before concluding the diagnosis.

More details are covered in the remaining part of this chapter. Section 4.2 provides an overview about the concept of trust and the trust models in MANET followed by a description of the proposed entropy-based trust model. In Section 4.3 provides a statistical background about the confidence interval followed by a description of how this interval is employed in our IDS.

4.2 Trust-based Intrusion Detection

The concept of trust has been originally used in social sciences in order to represent the level of faith about the behavior of a particular entity (e.g. a person or a thing) [141]. In computer science, trust has been defined as follows [142]: *"... trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent will perform a particular action, both before [we] can monitor such action (or independently of his capacity of ever to be able to monitor it) and in a context in which it affects [our] own action".* The utility of trust for taking a decision is explained in the definition provided in [143]: *trust is a subjective probability that helps the trusting entity in taking binary decisions by balancing known risks and the trustworthiness of the trusted entity.* Note that trust can be seen not only as a probability but also as a belief. For instance, trust is defined in [144] as the belief that an entity has, based on its own direct experiences, in another entity's capabilities, honesty, and reliability. Entity, here, can be a user, a computer, a smart phone, or a service provider. The previous definition ties the trust with the direct experiences of the trusting entity. However, information about the trusted entity's behavior can be provided by other entities in the form of recommendations. These recommendations play an important role in building the reputation of the trusted entity and thus, expecting its future actions [145]. In order to better understand the concept of trust, the following characteristics should be considered:

- Trust has a dynamic nature, i.e., the value of trust changes over the time,
- Trust is subjective, i.e., different entities may assess different levels of belief with respect to a trusted entity due to different experiences with this latter,
- Trust is asymmetric, i.e., the belief that the trusting entity has in the trusted entity is not necessary mutual,
- Trust is not always transitive, i.e., if A trusts B and B trusts C, then it is not necessary that A trusts C,
- Trust is function- or context-dependent, i.e., the level of belief between two entities is related to their considered interactions. For instance, an entity can be trustworthy for forwarding packets but not for detecting attacks.

Overall, trust can be defined as the level of belief/faith the trusting entity places on the trusted entity's capabilities and willingness to do/offer something in a given context and in a specific time slot. The trusting and the trusted entities are associated with a unidirectional relation called trust relationship. This relationship is characterized by three basic attributes: time slot, context, and trust value (Figure 4.1). The trust value expresses the strength of the relationship,

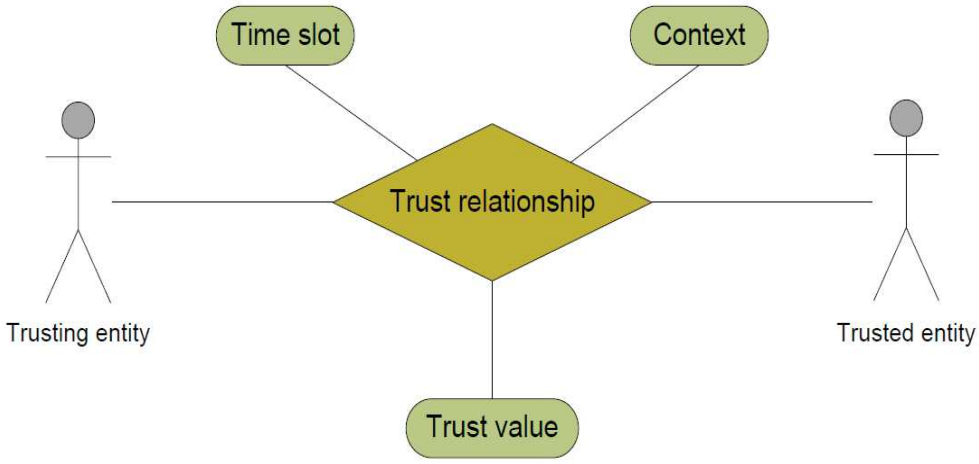


Figure 4.1: Direct trust relationship

i.e., the level of trustworthiness that the trusting entity assigns to the trusted entity in a given context during a specific time slot. Thus, several trust relationships may hold: each one is a unique combination of the aforementioned attributes (i.e., trust value, time slot and context). Initiating a trust relationship refers to deriving the trust value. This deriving generally depends [48] on the direct interactions and/or the recommendations. With direct interaction, the trust relationship is realized by a direct contact between the trusting and trusted entities. More precisely, the trusting entity depends on its own observations and experiences with the trusted entity to derive the degree of trust that should be assigned to this latter. Initiating the trust relationship by recommendation (also known as obtaining reputation) implies that the trust value is derived based on references/recommendations collected from third party mediators. In order to reduce the effects of unfair recommendations, the trusting entity commonly uses the trust value assigned to the mediator, as a trustworthy recommender, to weight the recommendations coming from this latter. The aggregation of the weighted recommendations results in the so-called reputation of the trusted entity (Figure 4.2). Building the trust relationships, determining and maintaining the trust values, and taking trust-based decisions constitute the overall trust model [146].

In Manet, the largest portion of the trust models is proposed to protect the routing protocol [147] against both selfish and misbehaving nodes [72, 104, 148, 149]. In practice, the source of a packet searches for the routes which do not contain untrustworthy nodes [150]. Other trust models have been further proposed for authentication [151, 152], access control [153], key management [154], and intrusion detection [155] purposes in Manet. We present our trust model (§4.2.1) that aims at building trust relationships and which is integrated into our Ids so as to have a robust intrusion detection (§4.2.2). The assessment of the attack risk is introduced in (§4.2.3). We then evaluate the robustness of the proposed trust-based detection against the falsified evidences (§4.2.4).

[48] Recently, several works are based on social engineering and real world friendships in order to build trust relationships between entities.

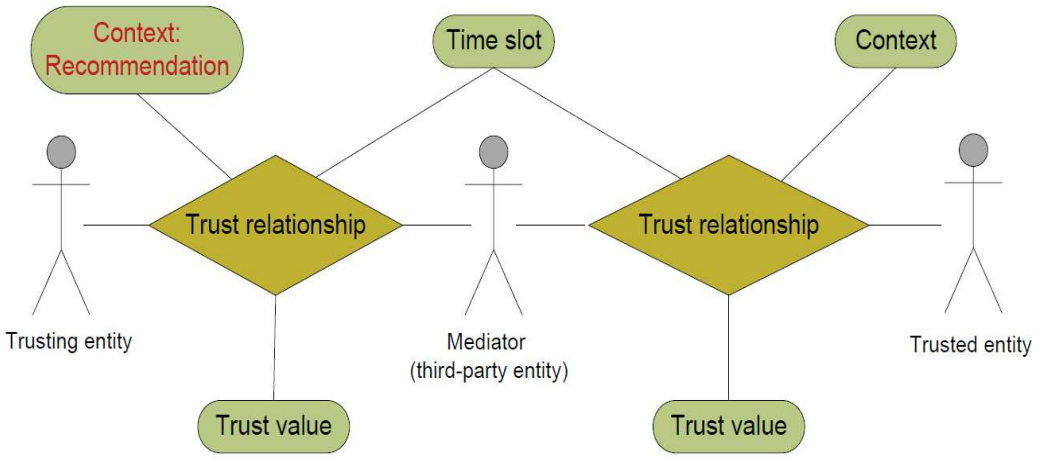


Figure 4.2: Recommendation-based trust relationship

4.2.1 Entropy-based Trust Model

We propose a distributed trust model that relies on the past evidences in order to establish the trust relationships between the nodes. A trust relationship $T_{A,B}$ between two nodes A and B represents to which extend A believes that B would act as expected (e.g., B forwards, without malicious modification, a control message towards the destination). This belief is represented as a trust value assigned to B. Several properties are taken into account during the calculation of a trust value:

- **Property 1**: the beneficial activities that are performed by a node increases the trust value assigned to this latter. While the node's harmful activities decrease its trust value. Examples of the beneficial activity include rightly relaying the traffic and providing a correct evidence during the intrusion diagnostic. In contrast, a harmful activity is related to, e.g., launching an attack or providing incorrect answers/evidences during the intrusion diagnostic.

- **Property 2**: the degree of gravity (versus reputability) of a harmful (versus beneficial) activity influences the risk for other nodes. Hence this degree should be reflected in the calculation of the trust value. For instance, relaying correctly the packets is less reputable than supplying correct answers to a diagnosis-related request.

- **Property 3**: fresh activities should be privileged in opposition to stale activities. Such privilege is needed so as to reduce the impact of Intoxication [156], a malicious node gains the trust of others by telling the truth over a sustained duration and only then starts lying.

- **Property 4**: first hand evidences (i.e., activities, either harmful or beneficial, that are observed by the node itself) are privileged compared to the second hand observations which are subject to controversial.

The above properties are enforced as follows. The node A calculates the trust $T_{\Delta t}(A, B)$ of the node B based on the n evidences $e_1^{A,B}, ..., e_i^{A,B}, ..., e_n^{A,B}$ about B which are observed/collected

by A during a time slot Δt:

$$T_{\Delta t}^{A,B} = \sum_{j=0}^{n} \alpha_j e_j^{A,B} + \beta T_{\Delta(t-1)}^{A,B} \tag{4.1}$$

The trust value ranges from 0 (i.e., complete distrust) up to 1 (i.e., complete trust), i.e. $T_{\Delta t}^{A,B} \in [0,1]$[49]. Beneficial and harmful evidences $e_j^{A,B}$ take respectively positive and negative values (property 1). A weighting factor α_j pondered $e_j^{A,B}$ so as to reflect the degree of gravity/reputability of this evidence (property 2). Meanwhile, a forgetting factor β permits to privilege fresh evidences rather than the stale evidences that were computed at the previous time slot $\Delta(t-1)$ (property 3). As long as no new activity is observed, the trust value is periodically updated i.e., either increased or decreased until it reaches the initial trust value. This ensures the redemption of nodes over time. Expression (4.2) represents how a node A updates the trust value $T_{\Delta t}(A,B)$ of a node B when no activity is observed/collected about this latter.

$$T_{\Delta t}^{A,B} = \beta T_{\Delta(t-1)}^{A,B} \tag{4.2}$$

In this expression, we note the association between the trust value before and after the fading: the forgetting factor β is formulated such that the higher (resp. the lower) the trust value before the fading, the smaller the decrease (resp. the increase) in the trust value after the fading. Note that the assignment of an initial trust value to a node that joins the network or moves to a new neighborhood is critical: the ignorance of historical activities of the nodes represents a major challenge for the trust model [143]. In fact, being pessimistic implies to assign a low trust value. A side effect is that new nodes may be less interrogated and may lack of opportunities to increase their trust value. Alternatively, the optimistic assignment of a high trust value favors the renewal of untrustworthy nodes which look for novel reputations. We believe that the initial trust value should be adapted according to the criticality of the application for which the trust model is developed: for instance a low initial trust value will be provided to support military applications. In our experiments, we select an initial trust value of 0.4, which is fairly low and ensures the defense nature of our trust model such that re-entering the network with a new identity is not so interesting.

When the observations of A are not sufficient, additional evidences are gathered from other nodes. These evidences are less reliable than the local evidences. Thus, an uncertainty is involved. To compute such uncertainty, the *entropy*, a measure of uncertainty stated in information theory [157], is used. In practice, trust is established through a third party (termed *concatenated propagation*) and through recommendations provided by multiple sources (called *multipath propagation*) [149, 158]. Let a *recommendation* $R_{A,S}$ represent how much A trusts the recommendations generated by S. A builds its belief about B according to the recommendation of S, a third party:

$$Tc_{\Delta t}^{A,B} = R_{\Delta t}^{A,S} T_{\Delta t}^{S,B} \tag{4.3}$$

Note that if A does not have an idea about the trustworthiness of S (i.e., $R_{A,S} = 0$) then the trust between A and B is still unknown (i.e., $Tc_{\Delta t}^{A,B} = 0$). When multiple nodes $S_1, S_2, ..., S_m$ generate a recommendation, A estimates its belief about B by applying the maximal ratio combining on

[49] We apply a Wraparound mechanism such that if the calculated trust value exceeds the upper limit (i.e., 1) or drops down the lower limit (i.e., 0) then the nearest limit is used as a trust value.

the recommendations. In this combination, a recommendation is weighted according to the ratio of its provider's trust value to the sum of trust values of all recommendation providers. Thus, a recommendation that comes from a trusty (resp. unreliable) node is amplified (resp. attenuated):

$$Tm_{\Delta t}^{A,B} = \sum_{i=1}^{m} w_i . T_{\Delta t}^{S_i,B} \tag{4.4}$$

with $w_i = \frac{R_{\Delta t}^{A,S_i}}{\sum_{j=0}^{m} R_{\Delta t}^{A,S_j}}$

The aforementioned trust model is adapted to our IDS so as to protect the intrusion diagnostic from the *blackmail* attacks.

4.2.2 Trusted Intrusion Diagnostic

We adapt the entropy-based trust model in order to build trust relationships between the nodes in the context of detection. The relationship $T_{A,B}$ between two nodes A and B represents to which extend A believes that B would return non falsified answers/evidences during the diagnosis. In order to establish $T_{A,B}$, A relies on the diagnosis-related evidences provided by B: $T_{A,B}$ increases (resp. decreases) every time B provides correct (resp. incorrect) evidence for a diagnosis initiated by A. In addition, A uses the alarms which are broadcasted when an intrusion is taking place. An alarm serves as a second-hand evidence that changes the trust value of the node(s) advertised as intruder(s) through the *concatenated propagation* (Formula 4.3). The amount of change depends on the trustworthiness of the alarm originator. In order to avoid the false accusations, an alarm message is considered only if it agrees with the self belief of the receiver. As usual, if the malicious nodes constitute the majority of the interrogated nodes, then a node A may be vulnerable to a brainwashing [156].

We further exploit the trust model so as to merge the evidences, even the inconsistent ones, during a diagnosis: the evidences are combined using the *multipath propagation* (Formula 4.4). Here, a returned evidence takes the value -1 if it confirms the existence of an attack, otherwise it takes the value 1. It is then weighted according to the trust value of its provider; an evidence that is provided by a trustworthy (resp. untrustworthy) node is amplified (resp. attenuated). The maximal ratio combination of these weighted, positive or negative, evidences represents the diagnostic result. Let illustrate the calculation of a diagnostic result by considering a link spoofing attack that is taking place between a suspicious node I and one of its neighbor S. During such diagnosis, other nodes $(S_1, ..., S_m)$ are interrogated to verify their neighborhood relations with I. Suppose that the interrogated nodes have provided the evidences noted $e^{S_1,I}, ..., e^{S_i,I}, ..., e^{S_m,I}$. These evidences are then combined according to (Formula 4.5) such that $e_i^{S_i,I}$ is pondered with a weighting factor w_i. This factor represents the ratio of the trust value of S_i (the provider of this evidence) to the sum of the trust values of all nodes that have provided answers.

$$Detect_{\Delta t}^{A,I} = \sum_{i=1}^{m} w_i e_i^{S_i,I} \tag{4.5}$$

with $w_i = \frac{T_{A,S_i}}{\sum_{j=0}^{m} T_{A,S_j}}$.

Recall that an evidence $e_i^{S_i,I}$ is either equal to 1 - the link which is advertised by I is correct, meaning that I does not carry a spoofing attack - or to -1 - the advertised link is wrong. If an interrogated node S_i does not return an evidence (before the waiting time elapses) then $e_{S_i} = 0$. Overall, a link spoofing is detected when $Detect_{\Delta t}^{A,I}$ is nearly equal to -1. Once stated, this result is used to update the trustworthiness of I, as well as $S_1, ..., S_m$. Indeed, if $Detect_{\Delta t}^{A,I}$ is less than 0 (i.e., a link spoofing attack is detected) then the attacker I and every interrogated node S_i that denies the existence of a spoofing (i.e., $e_i^{S_i,I}$ is equal to 1) lose a part of their trustworthiness. While the trustworthiness of an interrogated node S_j that confirms the existence of a spoofing (i.e., $e_j^{S_j,I}$ is equal to -1) is increased. In contrast, if there is no spoofing (i.e., $Detect_{\Delta t}^{A,I}$ is more than 0) then the trustworthiness of the nodes that confirm the existence of a spoofing is decreased. While the trustworthiness of the nodes that deny the existence of a spoofing is increased. In both cases, the amount of increase/decrease in trustworthiness is associated to the risk of the detected attack.

4.2.3 Attack Risk

We propose to amplify the preventative nature of our trust model taking into account the attack risk which varies depending on: (i) the damages that may be caused by an attack and (ii) the degree of acomplishment of the attack. In practice, each evidence e (Expression 4.1) is pondered with a degree of gravity α that reflects the risk of the attack. This risk depends of two key factors[50]:

- **Attack evolution:** an attack is seen as an (ordered) sequence of actions that are realized by the attacker (and colluding node(s)) so as to achieve malicious purpose(s) (e.g., falsifying MPR selection in OLSR protocol). An attack tree is a systematic method that has been proposed so as to assess the cause-consequence relationship between the actions composing the attack [159]. The root of this tree represents the first action (or the state) that should be realized by the attacker while the leaves represent the goals of the attack. When the attack evolves, i.e., the attacker goes deeply in the tree and approaches the leaves, the damages become closer. Therefore, the risk is proportional to the level of attack's evolution l. In our model, l is measured as the depth of the tree.

- **Derived attacks:** a successful attack may open the door for others, perhaps more serious, attacks. For instance, gaining maliciously a MPR position may be intended to tamper packets. The degree of gravity is proportional to the number d of potential attacks that can be derived.

The above properties are used to calculate the degree of gravity α for an activity or an evidence e as follows:

$$\alpha = l * d \tag{4.6}$$

Let us illustrate the estimation of the degree of gravity with a link spoofing attack defined in the previous chapter. For this purpose, we build the attack tree (Figure (4.3)), which is composed of

[50]Note that another factor may be considered so as to reflect the subjective point of view of the security engineer. For instance, an attack that leads to violate the confidentiality or a denial of service is considered too risky in an *ad hoc* network developed for military purposes [16].

3 levels. First, the attacker should be a 1-hop neighbor of the victim (root action). The second level contains the three types of link spoofing: non-existing 1-hop neighbor is advertised, existing but non neighboring node is advertised as a 1-hop neighbor, or 1-hop neighbor is omitted. By taking the first and/or the second option, the attacker increases its connectivity and hence can be chosen as MPR by the victim (second leaf). In addition, a legitimate MPR may be excluded from the MPR set of the victim (optional leaf). While the third choice of the link spoofing leads to reduce the connectivity of both the victim and the attacker. This latter is no more chosen as MPR (third leaf). When the attacker succeeds, it may launch other derived attacks. For instance, when the attacker succeeds to be selected as a MPR of the victim then all the traffic of this latter will be forwarded to the attacker. Consequently, the attacker may mis-relay the packets or replaying them in other places by tunneling them toward another colluding node and hence a *wormwhole* is realized. The attacker may also drop all packets (*blackhole* attack) or some selective ones (*grayhole* attack). Both may lead to a DoS attack. The DoS may be realized also when the attacker maliciously excluding himself/herself from the MPR set of the victim. Supposing that an intruder I launches a link spoofing attack against one of its 1-hop neighbors

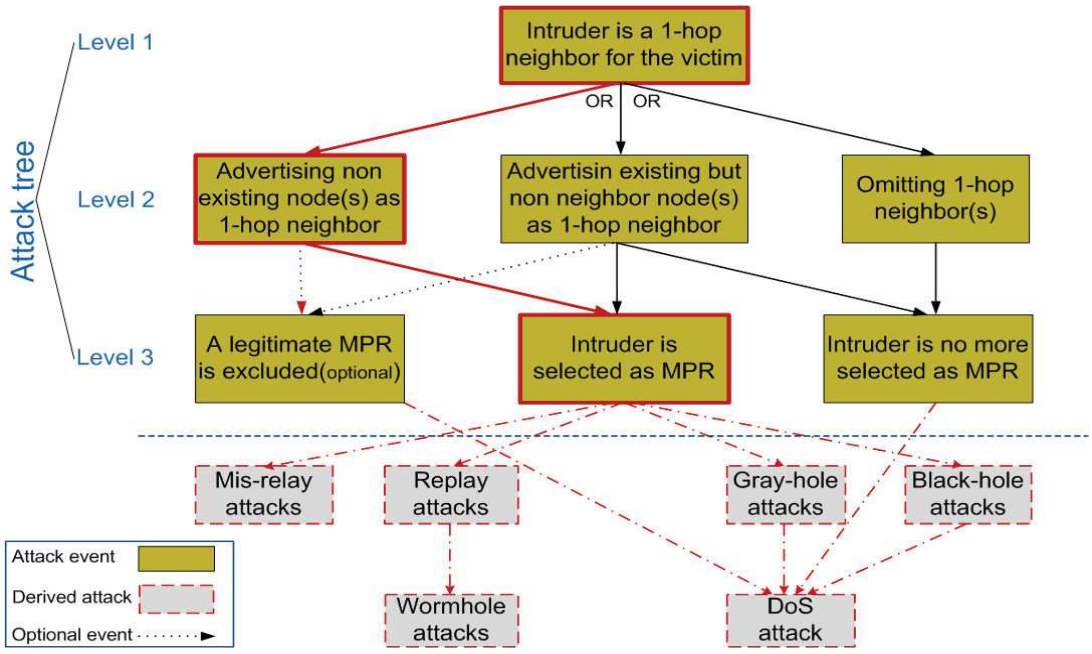


Figure 4.3: Link spoofing attack tree

(i.e., I has already exceeds the first level of the attack tree). For this purpose, I sends a false *hello* message advertising a non-existing node as 1-hop neighbor (i.e., attack is evolved into the second level of the attack tree ($l = 2$)). If the attack succeeds and becomes a MPR then he/she would have the possibility to realize 4 further attacks: mis-relaying, replaying , *blackhole* and/or *grayhole* attacks. According to Formula (4.6), the degree of gravity is $\alpha = 2*4 = 8$. Once stated, this degree of gravity (resp. reputability) is used to update the trust value of the nodes that provided incorrect (resp. correct) evidences during the diagnosis related to the aforementioned scenario of link spoofing attack.

4.2.4 Evaluation

We analytically evaluate [51] the proposed trust model such that the aforementioned formulas are used to estimate: (i) the evolution of the trustworthiness for, both, well-behaving and misbehaving nodes, (ii) the trust fading, and (iii) the aggregation of the evidences. We consider a case study that contains, unless specified, 16 nodes including 1 attacker that performs a link spoofing attack and 4 misbehaving nodes. The misbehaving nodes constitute 26.3% of the network and aim at foiling the detection by confirming that the falsified neighborhood relation advertised by the attacker is true. Initially, each node is assigned a random trust value. The attacker launches a link spoofing attack that, unless specified, takes place during the overall experiment. Similarly, the misbehaving nodes always supply incorrect evidences during the diagnosis. The evolution of trustworthiness as seen by the node under attack is presented

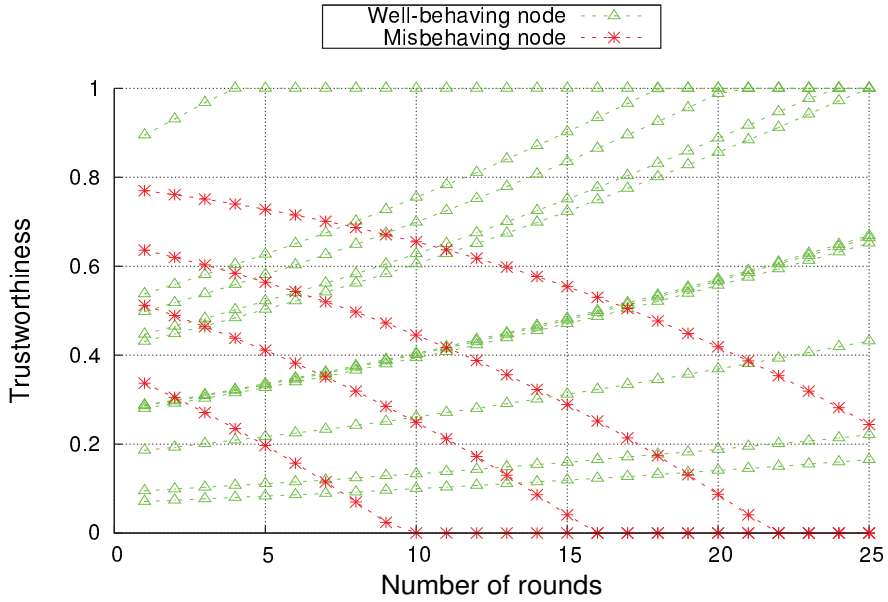


Figure 4.4: Evolution of trustworthiness

in Figure 4.4. The constant maintaining of the well-behaving and misbehaving explains the (monotonous) ascending versus descending rate of the node trustworthiness. One may note the defensive nature of our trust model: the trust associated to a liar falls dramatically regardless of the initial trust, whereas the well-behaving nodes with a low initial trust value gain moderately the trust of others. Then, if the attack ceases (Figure 4.5), both liars and well-behaving nodes recover due to the forgetting factor. One may note that the nodes with a high or medium initial trust value reach the default trust value (herein 0.4) in the the first rounds. Nodes with small initial trust values recover slowly: they are assigned with the default trust value after passing 25 rounds without providing any diagnostic evidence. This represents the defensive nature of our trust model which demands a long misconduct-less duration before trusting a former liar. Figure (4.6) shows the impact of using the trustworthiness during the intrusion diagnostic. Here, the

[51] The trust model is already implemented and integrated into LIDR (Appendix C.3). However, we still need to execute it in our simulator.

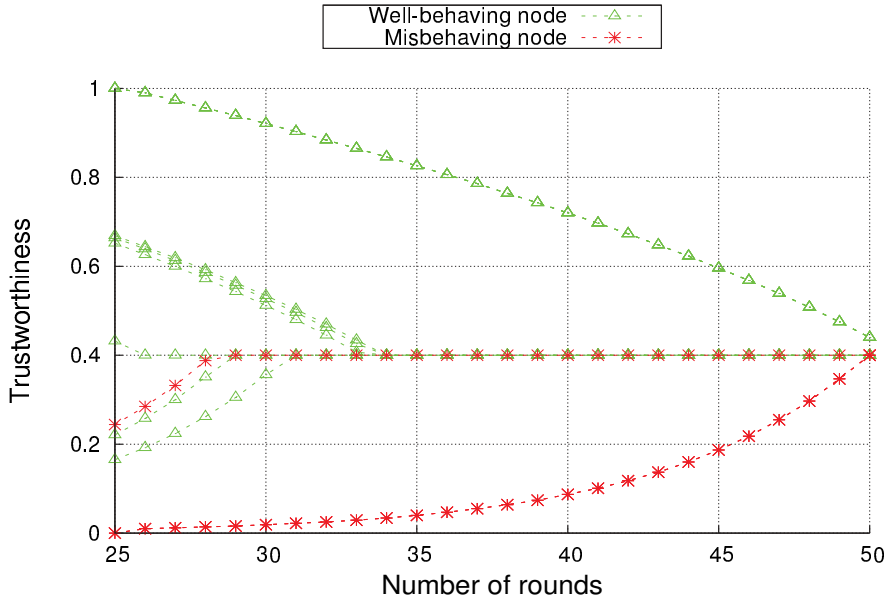


Figure 4.5: Trust fading

diagnosis-related evidences are inconsistent due to the presence of liars. They are aggregated according to Expression 4.5 such that each evidence is pondered with the trust value. When the trust value assigned to each node reflects its true nature, i.e, liars have low trust values while the well-behaving nodes have high trust values, then the diagnosis correctly refers whether an attack takes place. It is worth to mention that the trustworthiness of the nodes evolve correctly, and hence the diagnosis concludes correctly, provided that the impact of the liars is less than the impact of the well-behaving nodes. However, the sum of liars 's weighted evidences should be less than the sum of the well-behaving nodes' evidences. Otherwise, an intoxication takes place and the legitimate nodes starts losing their trustworthiness in favor of the liars. In order to evaluate the impact of the liars, diagnostic result has been calculated for several percentages of liars (Figure 4.7). As expected, augmenting the percentage of liars delays the conclusion of the diagnosis: more interrogation rounds are required until the diagnostic result approaches -1 (meaning that an attack has occurs). However, the diagnostic result continues falling down and approaching -1 along the time even when the liars constitute 43.2% of the nodes. During the last rounds, the detection converges to -0.8 regardless of the percentage of liars: the trust values of the liars diminish dramatically in the last rounds, and thus their influence on the detection almost disappears.

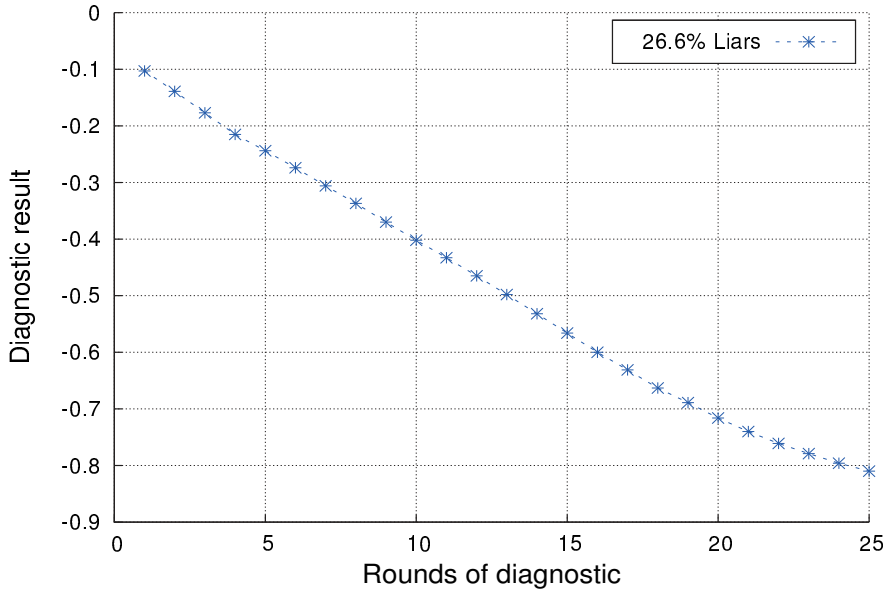


Figure 4.6: Trust-based intrusion detection

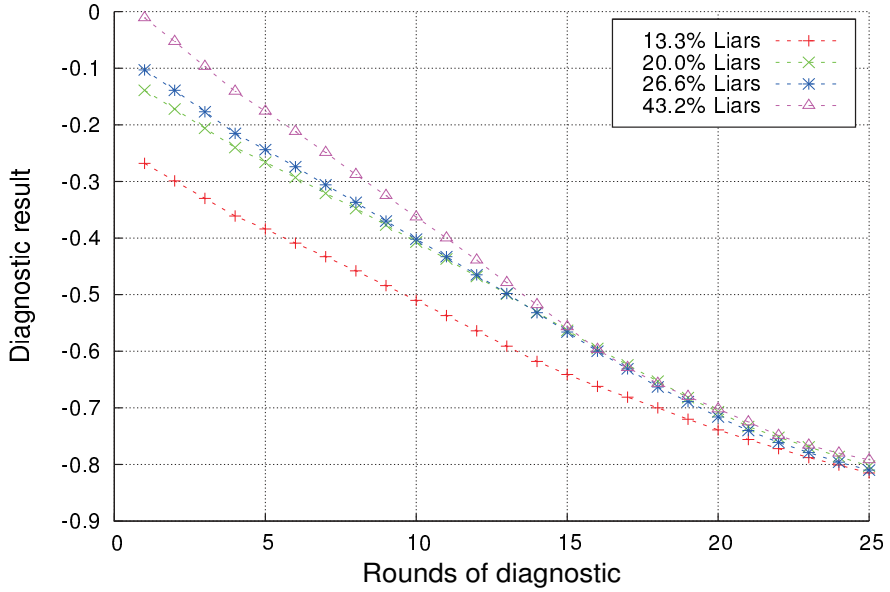


Figure 4.7: Impact of liars on the detection

Overall, the node cannot misbehave and still have a high trust value in the same time. Decreasing the trustworthiness of the misbehaving nodes diminishes their impact on the detection. Obviously, the accuracy of the intrusion detection is enhanced when the trust values of the nodes reflect properly their true nature. However, the presence of inconsistent answers/evidences during the diagnosis always generates a doubt about the diagnostic result. Waiting for more diagnostic and trust estimation rounds before concluding the diagnosis, so as to reduce the level of doubt, means that more evidences are gathered. Consequently, more resources will be

consumed before obtaining the diagnostic result. Therefore, there is a need to find a trade-off between the detection accuracy and the resources consumption.

4.3 Statistics-based Interrogation

In this section, we provide a background about the inferential statistics and the confidence interval. We further detail the mathematical definition and computing of the confidence interval, the confidence level, and the sampling error (4.3.1). In (4.3.2), we present how the confidence interval is integrated in our IDS as a measure of reliability. We then evaluate the efficiency of using the confidence interval in order to find a trade-off between the detection accuracy and the resources consumption (4.3.3).

Statistics are defined in [160] as the science of gathering, organizing, interpreting, and analyzing data so as to make decisions. They refer to the set of mathematical procedures that condense, usually, large quantities of information into a few simple informative facts and figures [7]. These facts and figures answer the questions about a group (or groups) of **individuals**. For example, the effects of the summer job on the average income in France, or the analysis of the political attitudes for men compared to women during the presidential election. In statistical terminology, the entire group of individuals for which statistics are established is called **population**. The population is not uniquely composed of people, but it could be a population of measurements, corporations, or anything. Regarding the size of the population, it may be small e.g., the students in a class, or extremely large e.g., the adults who are registered voters in France. Since the examination of all the individuals forming a large population is very costly in terms of money, time, and resources, the statistical studies are limited to a smaller group of individuals, more manageable, that are typically selected from such population. More precisely, the characteristics of a population (e.g., mean, median, standard deviation) are usually inferred based on a subgroup of the individuals which represent this population. Mostly, this selection is done randomly, i.e., population individuals have equal chance to be selected, although, non-random criteria-based selection could be also used [8]. In statistical terminology, such selected set of individuals is called a **sample**, and is intended to represent the population. The selected sample can vary in size. After finishing examining a sample, the results are then generalized back to the entire population. A population characteristic, e.g., mean or median, is termed a **parameter**; a population parameter is usually a numerical/quantitative value that describes a population. A characteristic of a sample is termed **a statistic** (or an **estimate**); a sample statistic is a numerical/quantitative description of a sample. Figure (4.8) represents the full relation between a population and its sample. Statistics are classified into two general branches: **descriptive** statistics which involve organizing, simplifying, summarizing and displaying data in a more manageable form, and **inferential** (called also inductive) statistics that use probabilistic techniques so as to examine a sample from a population, and then make a generalization on the obtained answers in order to improve our knowledge about this population. In other words, the sample statistics are used as a basis for estimating the population parameters. Inferential statistics allow us to have facts and answers about the large population without need to examine all its individuals. However, to which extend these facts generalized from a sample are accurate? This question should be asked because a sample is still a representative of its population,

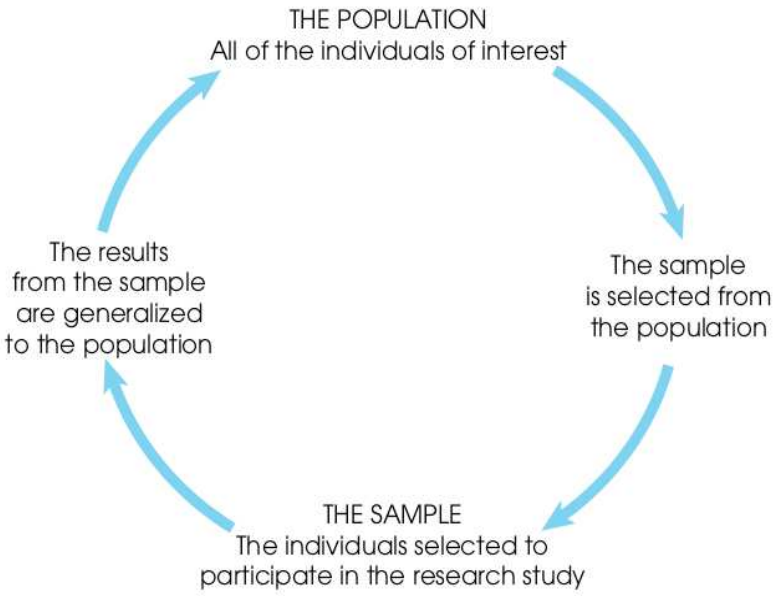


Figure 4.8: The relation between a population and a sample [7]

but it does not necessary give a perfectly accurate reflection about the whole population. A discrepancy is usually existing between a population parameter and the corresponding sample statistic. In statistic, this discrepancy is termed **sampling error**. We illustrate the concept of sampling error in Table 4.1. Let consider a population of 10 students, and further choose randomly two samples from this population, each one is composed of 4 students. Note that the individuals of these samples are distinct. Here, two characteristics are calculated: the average age and the average height, for the population as well as for the selected samples. Since the characteristics of a sample depend on its individuals, the sample statistics vary between the two selected samples. For instance, the average age in the first sample equals to 19.75, while it is equal to 19.0 in the second sample. Moreover, neither the statistics of the first nor the second sample are identical to the population parameters (e.g., the average age in the population is 19.4). It is worth to mention that we present two samples of many of possible samples. Each of these latter typically contain different individuals, thus producing different statistics. Imagine now that we classify the students according to the age, i.e., the population is divided into two groups such that the first group contains the younger students while the second group contains the older students. We then select one sample from each group. Again, the sample statistics will vary from one sample to another, and are typically not identical to the population parameters. Overall, no matter how we select the samples, they will mostly produce different statistics. As aforementioned, this variation of sample statistics from one sample to another, and further from the corresponding population parameters represents the concept of sampling error. Thus, there is a need to evaluate the accuracy of a sample-based estimation of a population parameter. For this purpose, the **confidence interval** is designed to override the effects of sampling error on the precision of a sample statistic [161].

The confidence interval is used so as to represent statistical imprecision or uncertainty associated with the estimation of a population parameter (e.g., proportion, mean) from a sample [162].

Table 4.1: Sampling error

Eric	18	175
Laura	19	165
Mouhannad	20	170
Francoise	22	167
Julien	19	180
Kate	17	177
Ali	21	185
Brian	18	181
Sara	20	161
Kristen	17	172

Eric	18	175
Laura	19	165
Mouhannad	20	170
Francoise	22	167

Kate	17	177
Ali	21	185
Brian	18	181
Sara	20	161

A common example of confidence interval statement is a pollster's claim that he (or she) is, for example, 95% confidant that the percentage of vote for a presidential candidate is, for example, 42% with a margin of error equals to 4%. In this case, it is obvious that the voting population is very large (e.g., the voters registered in the United States). Thus, the previous survey is surely based on a group of possible voters (i.e., a sample). The percentage 42% represents the **point estimate** of the true percentage of votes. However, a point estimate is not recommended with inferential statistics due to the sampling error, i.e., we cannot figure out to which extend the point estimate is close to the population parameter [162]. To override this shortcoming, the pollster uses the **interval estimate**, the true percentage of votes for the mentioned candidate is specified as between two values. In practice, this interval (i.e., $[38\%, 46\%]$ in our example) is estimated by adding (resp. subtracting) the margin of error (i.e., 4%) to (resp. from) the percentage of votes (i.e., 42%). Since the pollster specifies 95% as a degree of confidence (i.e. he or she is 95% sure that the true percentage of votes is within the estimated interval), then the interval $[38\%, 46\%]$ is considered as a 95% confidence interval. This means that under repeated and random sampling under identical conditions, this confidence interval will contain the sample-based percentage of votes 95% of the times. In other words, the pollster is 95% sure that the interval $[38\%, 46\%]$ would contain the true percentage of votes if all possible voters (i.e. all the individuals in the population) are included in his/her survey. The confidence interval is seen as a measure of certainty around a sample statistic, telling us the range of values within which the true population parameter lies with a given degree of confidence. This degree of confidence is termed as **confidence level** (*cl* for short). The confidence interval helps in estimating, with a specific degree of confidence, the opinion of a population based on a limited-size sample from this population. We employ the confidence interval in our IDS such that instead of gathering and processing all the evidences, a subset of these latter is used to estimate whether or not an attack takes place. Thus, we reduce the amount of gathered and processed evidences that are necessary to detect the attack. In addition, since the confidence interval is based on a specific confidence

level, we guarantee the required detection accuracy. As a result, the confidence interval permits to find a compromise between the detection accuracy and the resources consumption (Figure 4.9).

Figure 4.9: Using the confidence interval to estimate a population's opinion.

The confidence interval is largely used in election surveys in order to estimate the opinion of the voters about a political candidate. Since interrogating all the voters is impossible, such estimation is based on a sample of voters. We adapt this concept in order to estimate, based on only a subset of evidences, the opinion of the network about a suspicious device. This estimation is related to a degree of certainty and helps in reducing time and resources consumption in the intrusion detection.

Before delving into the utilization of the confidence interval with our IDS, let us introduce the mathematical background on the confidence interval.

4.3.1 Confidence Interval

We hereafter describe the computation of the confidence interval for the mean which is the most used population parameter [162]. The confidence interval for other population parameters, e.g., proportion, median, is similarly computed. Given the sample mean (i.e., the point estimate) μ_s and the sampling error ε (also called standard error) that is calculated according to a specified confidence level (cl for short), the population mean μ is delimited by a lower limit ($\mu_s - \varepsilon$) and an upper limit ($\mu_s + \varepsilon$), i.e., $\mu_s - \varepsilon < \mu < \mu_s + \varepsilon$. The interval $[\mu_s - \varepsilon, \mu_s + \varepsilon]$ is the $cl\%$ confidence interval of the population mean. It is obvious that the key element during the estimation of the confidence interval is the sampling error of the mean. When the sampling distribution of the mean follows a normal law, the sampling error is calculated as follows [161]:

$$\varepsilon = z_{\alpha/2} \frac{\sigma_s}{\sqrt{n}} \tag{4.7}$$

With a sample size denoted n, and a standard deviation σ_s of the sample that is calculated as follows:

$$\sigma_s = \sqrt{\frac{\sum_{i=0}^{n}(\mu_s - x_i)^2}{n-1}} \tag{4.8}$$

with $x_1, x_2, ..., x_n$ correspond to the sample individuals.
The factor $z_{\alpha/2}$ is related to both the required confidence level and the form of the sampling distribution. The confidence level refers to the expected percentage of times where the estimated confidence interval would contain the sample mean, under repeating sampling from the population [8]. Hence, it refers to the probability that the true value of the population mean falls into the estimated confidence interval. Thus, the larger the confidence level, the higher the certainty that the confidence interval contains the true value of the population parameter. The confidence level is usually written as $100(1-\alpha)\%$ where α, which is less than or equal to 1, represents the portion of the sampling distribution outside the confidence interval. In order to understand the relation between the confidence level and α, let introduce some terms and theorems. In statistics, the probability distribution of a variable x specifies the likelihood or the percentage for each value that x can take within a range. This range is limited between the minimum and the maximum statistically possible values of x. Moreover, the probability that x would assume a value within a specific interval can be obtained by determining the area under the probability distribution graph that is limited by this interval. For instance, assuming that Figure 4.10 shows the graph of the probability distribution of a variable x, the percentage of the area between a and b refers to the probability that x has a value within the interval $[a, b]$, i.e., $Pr(a < x < b)$. The sampling distribution of a sample mean (or any other sample statistic) is

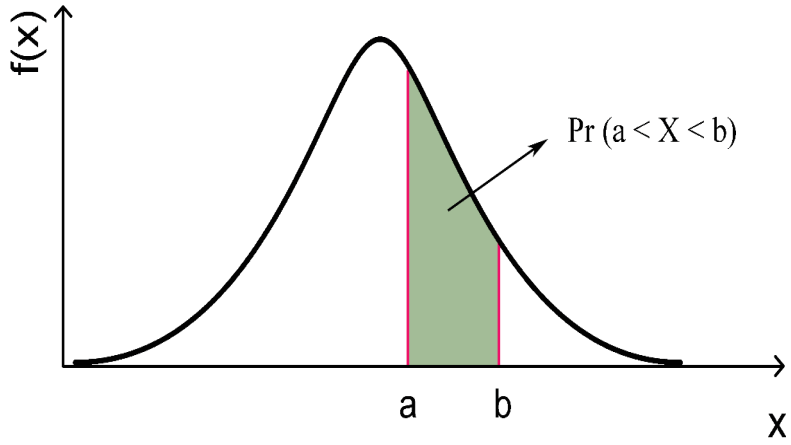


Figure 4.10: Probability distribution of X

the probability distribution of the mean obtained under repeated sampling of the population [8]. Supposing that the sampling distribution of the mean has a normal distribution form (Figure 4.11), the percentage of the area under the distribution graph, for example, from $\mu - \varepsilon$ to $\mu + \varepsilon$ represents the probability that the mean has a value within the interval $[\mu - \varepsilon, \mu + \varepsilon]$. In other words, this area contains the samples whose means fall into the interval $[\mu - \varepsilon, \mu + \varepsilon]$. Thus, the interval $[\mu - \varepsilon, \mu + \varepsilon]$ is considered as a confidence interval with a confidence level equals to the

percentage of the area from $\mu - \varepsilon$ to $\mu + \varepsilon$. The lower and upper tails of the distribution graph, where the mean is less than $\mu - \varepsilon$ or more than $\mu + \varepsilon$ respectively, represents the probability that the sample mean is not within the interval $[\mu - e, \mu + e]$. In other words, they contain the samples which their means are not within the confidence interval. Thus, they reflect the sampling error. It is clear that by increasing the width of the interval $[\mu - \varepsilon, \mu + \varepsilon]$ (the tails of sampling error henceforth become narrower), the probability that a sample mean is within the confidence interval increases. Therefore, the width of the confidence interval increases along with the confidence level. In Figure 4.11, we assume that the interval $[\mu - \varepsilon, \mu + \varepsilon]$ constitutes 95% (or $100(1 - 0.05)\%$) of the distribution area. Thus, the area outside this interval constitutes $100\% - 95\% = 5\%$ of the total distribution area such that each tail covers 0.025 of the distribution. α in (Formula 4.7) refers to the total area in both tails of the distribution graph, and $\alpha/2$ (in our example $\alpha/2 = 0.025$) refers to the area in each one of the tails. Since the sampling distribution of the mean has a normal form, we determine the percentage for every area specified by an interval through the Table of areas for the standard normal distribution also called Z-distribution (Appendix B.1) [160]. The Standard normal distribution is a special case of the normal distribution with mean 0 and standard deviation 1. Furthermore, if the percentage

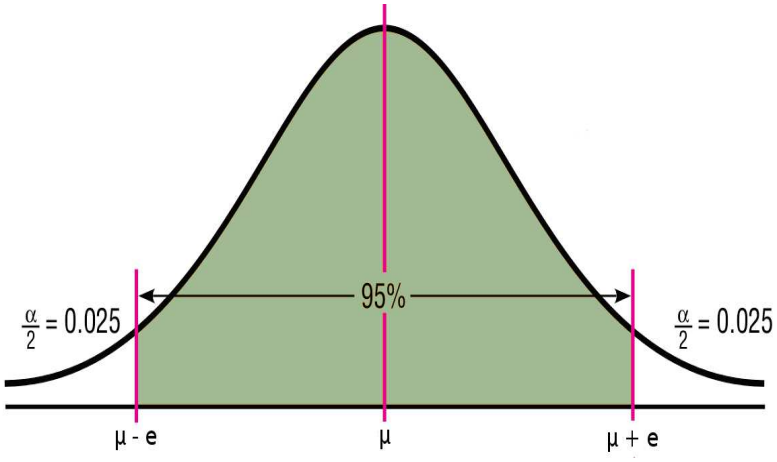


Figure 4.11: Sampling distribution of the mean

of an area (i.e., the required confidence level) is known, then the corresponding interval (called z-score) can be taken from the standard normal distribution table. In Formula 4.7, $z_{\alpha/2}$ represents the z-score that corresponds to the area of the Standard Normal distribution where $\alpha/2$ is excluded from each tail. For instance, if the required confidence level is 95% (i.e., $\alpha/2 = 0.025$), then the corresponding z-score is $z_{0.025} = 1.96$. Table (4.2), contains the z-scores for the most commonly used confidence levels.

So far so good, but can we use the z-score with the sampling distributions that are not normal? To answer this question, one should discuss the size of the sample[52]. If all the possible samples of the population are considered, then the mean of the sampling distribution of the mean is equal

[52]If the standard deviation of the population is known, then the z-score is used regardless of the sample size and the sampling distribution form. However, in this case, it is practically meaningless to estimate the confidence

Table 4.2: z-scores for the most commonly used confidence levels.

80%	1.28
90%	1.65
95%	1.96
99%	2.58
99.9%	3.29

to the population mean [160]. However, considering all the possible samples from an infinite or large population is practically impossible. Therefore, the mean of the sampling distribution of the mean differs from the population mean. The **law of large numbers** [163] states that this difference which reflects the sampling error, decreases along with the sample size [8]. More precisely, the sampling error decreases as the sample size increases. In addition, according to the central limit theorem introduced by Pierre Simon Laplace [164], as the sample size gets larger, the sampling distribution of the mean begins to look like a normal distribution. In practice, regardless of the population form, once the sample size exceeds 30, then the sampling distribution of the mean can be approximated by a normal distribution [160]. Thus, the z-score is used when the sample size exceeds 30. Otherwise, with small samples, the sampling distribution of the mean is considered as a t-distribution (also called Student's t-distribution, or Gosset distribution) [8]. The t-distribution is quite similar to the normal distribution, but the former has larger tails comparing to the latter as it is illustrated in Figure 4.12. This means that the confidence interval based on t-distribution is wider than it would be if this confidence interval was based on the normal distribution. Thus, the t-distribution is less precise than the normal distribution. However, as the sample size increases, the difference between the two decreases

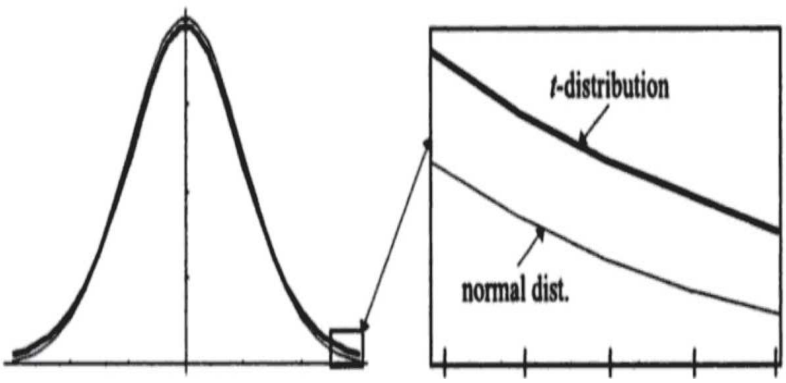


Figure 4.12: Normal distribution compared to a t-distribution [8]

until it is not of practical importance. It is worth to mention that each particular curve of the t-distribution is determined by a degree of freedom (*df* for short), such that changing the

interval of the mean because knowing the standard deviation of the population means that the true population mean is already known.

value of *df* changes the shape of the t-curve [160]. Given n the size of the sample on which the estimation is based, then the *df* of the corresponding t-distribution is $n-1$. Therefore, when $n < 30$ then the sampling error is calculated as follows:

$$\varepsilon = t_{\alpha/2}\frac{\sigma_s}{\sqrt{n-1}} \tag{4.9}$$

Where the value of $t_{\alpha/2}$ can be taken from the table of areas for the t-distribution (see Appendix B.2), with $df = n-1$.

Overall, the estimation of the confidence interval of a population mean involves the following steps:

1. Collect a sample from the population.
2. Calculate the sample mean μ_s and the sample standard deviation σ_s.
3. In order to ascertain the portion of the sampling distribution that is excluded from the confidence interval, subtract the required confidence level to 100 and divide the result by 2 so as to obtain $\alpha/2$.
4. When the sample size n is more than or equal to (resp. less than) 30, select the $z_{\alpha/2}$ (resp. $t_{\alpha/2}$ with $df = n-1$) from the z-distribution (resp. the t-distribution) table.
5. Use Formula 4.7 (resp. Formula 4.9 when $n < 30$) to calculate the sampling error ε.
6. Estimate the confidence interval as $[\mu_s - \varepsilon, \mu_s + \varepsilon]$

Increasing the confidence level leads to increase the certainty that the true population mean is within the corresponding confidence interval. However, it renders the confidence interval wider, thus the precision is reduced. Similarly, increasing the sample size reduces the sampling error and the confidence interval becomes narrower while its precision rises.

4.3.2 Lightweight and Reliable Evidence Gathering

The computation of the confidence interval during the diagnostic permits our IDS to: (i) minimize the number of gathered and processed evidences while maintaining the required detection accuracy, and (ii) measure to what extend the diagnosis is reliable, especially in the presence of inconsistent evidences. In other words, finding a compromise between the resource consumption and the detection accuracy implies to be informed about when gathering more evidences is redundant and does not change the obtained result. For this purpose, LIDR is enriched in order to exploit the confidence interval as a measure of reliability (Figure 4.13). In practice, the first steps of detection, which consist in extracting the evidences from the logs and matching those evidences against the intrusion signatures, remain unchanged. The change takes place when remote evidences are gathered from other nodes: rather than gathering all the evidences, only a subset/sample of those evidences are gradually gathered. This subset is gathered by randomly and uniformly interrogating a group of candidates that provide the evidences about the suspicious node. Based on the gathered evidences, a diagnostic result is calculated as usual,

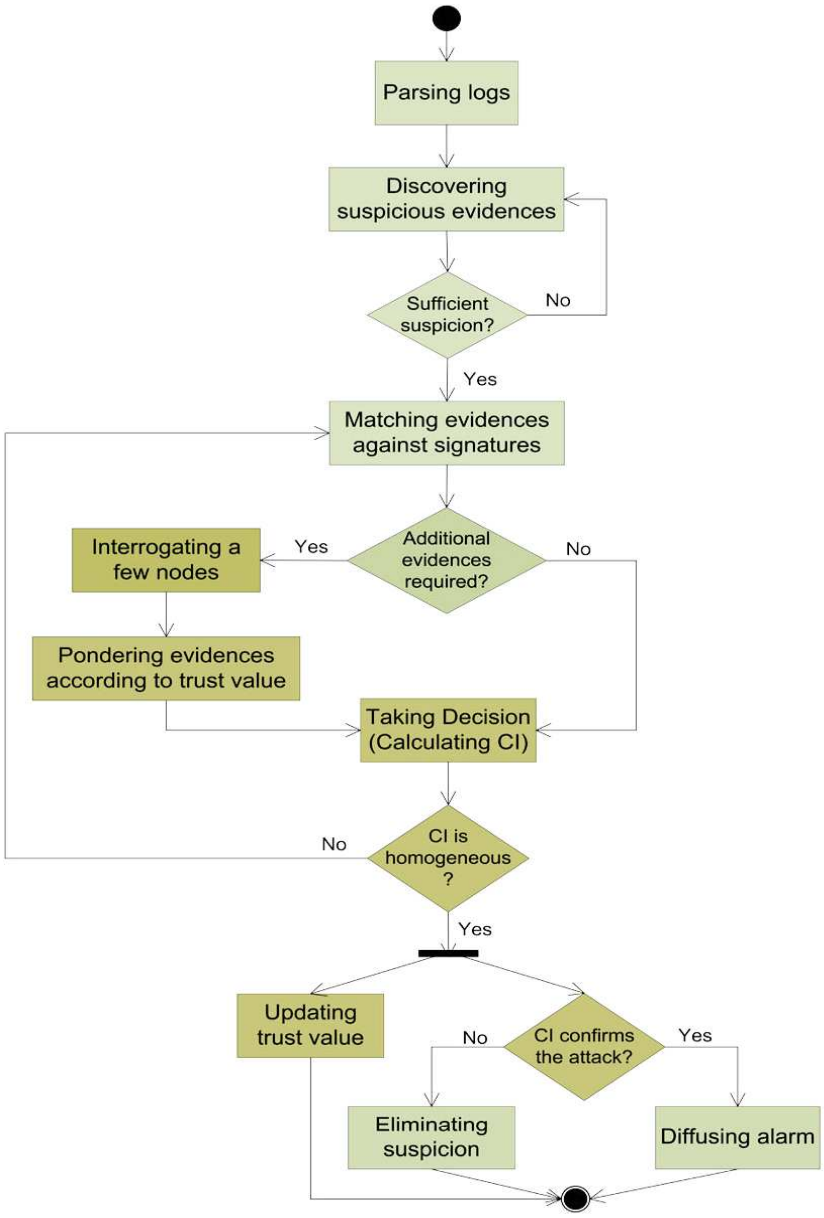


Figure 4.13: Integrating the confidence interval into the diagnosis

i.e., the gathered evidences are pondered according to the trustworthiness of their sources and further aggregated according to Formula 4.5. The result represents a sample statistic that may differ from the true diagnosis (i.e., the one that would be obtained if all evidences in the network are gathered and processed). Therefore, the sampling error of the obtained diagnostic result is calculated. Formula 4.7 (resp. Formula 4.9 with a number of gathered evidences bellow 30) is used. Here, the applied confidence level reflects the required detection accuracy, i.e., if the required accuracy is high then a high confidence level is chosen and vice versa. Then, the confidence interval of the diagnostic result is estimated based on the computed sample error. The

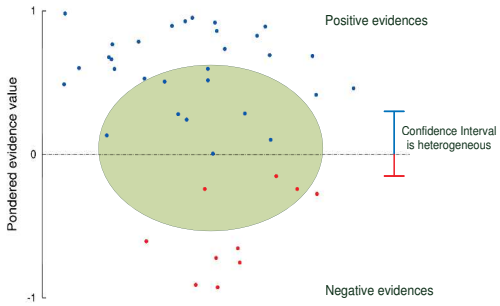
true diagnostic result[53] should be within the estimated confidence interval. Thus, the sample-based result is accepted only if the estimated interval is homogeneous, i.e., all its values deny (Figure 4.14(c)) (resp. confirm (Figure 4.14(d))) that the suspicious node is an intruder. If the confidence interval is heterogeneous, then the true diagnostic result can either confirm or deny the attack. This means that there remains a doubt about the obtained diagnostic result and thus, more evidences should be gathered before completing the diagnosis (Figure 4.14(a), 4.14(b)). For a better understanding, given $Detect_{A,I}$ as the result of a diagnosis initiated by A as to verify whether a suspicious node I is an intruder or not. $Detect_{A,I}$ is calculated based on a group of gathered evidences $e_1, e_2, ..., e_n$. Let ε the sampling error of $Detect_{A,I}$ and 95% is the used confidence level. Then, the 95% confidence interval of the diagnostic result is $[Detect_{A,I} - \varepsilon, Detect_{A,I} + \varepsilon]$ such that $Detect_{A,I} - \varepsilon$ and $Detect_{A,I} + \varepsilon$ are, respectively, the lower and the upper limits of the interval. Recall that a diagnostic result takes a value between -1 and 1 (i.e., $-1 \leq Detect_{A,I} \leq 1$) such that a negative (resp. a positive) result means that the suspicious node is an intruder (resp. a legitimate node). When the confidence interval contains only negative values, i.e. $Detect_{A,I} + \varepsilon \leq 0$, then 95% of the evidence samples, under repeating random sampling from the evidence population, confirm that the suspicious node is an intruder. Similarly, when the confidence interval contains only positive values, i.e. $Detect_{A,I} - \varepsilon > 0$, then 95% of the evidence samples, under repeating random sampling from the evidence population, confirm that the suspicious node is a well-behaving node. In both cases, we are 95% sure about the diagnostic result, thus no more evidences have to be gathered and the diagnosis is concluded. On the other hand, if the confidence interval contains, both, negative and positive values, i.e. $((Detect_{A,I} - \varepsilon \leq 0) \wedge (Detect_{A,I} + \varepsilon > 0))$, then some evidence samples confirm the existence of an attack while others deny it. This means that the diagnostic result is not yet reliable and thus, more evidences should be gathered and processed. It is clear that when the confidence interval is wide, i.e., it offers imprecision estimation of the diagnostic result, then there is less chance to conclude this diagnosis. Recall that the larger the sampling error, and thus the wider the confidence interval, the lower the precision of this interval. Several factors impact the sampling error:

- Sample size: increasing the number of evidences that are used in calculating the diagnostic result leads to reduce the sampling error.

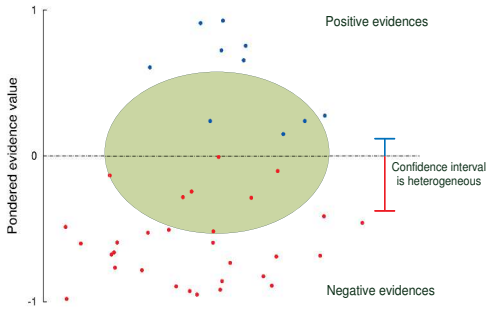
- Detection accuracy: the higher the required detection accuracy, i.e., the used confidence level, the larger the sampling error. In our IDS, the detection accuracy is fixed at 95%.

- Inconsistency of evidences: the higher the variance between the sample evidences, and hence the larger the sample standard deviation, the larger the sampling error. Here, the inconsistency of evidences represents the difference, in terms of the impact on the diagnostic result, between the trust-based pondered negative evidences, which confirm the existence of an attack, and the trust-based pondered positive evidence, which deny the existence of an attack. As this difference approaches 0, as the inconsistency increases. Hence, the confidence interval becomes wider.

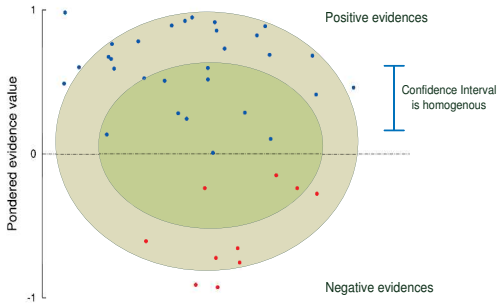
[53]Note that the true diagnostic result, which is based on all evidences, answers correctly whether or not an attack takes place providing that the falsified evidences constitute less than the half of the evidences population.



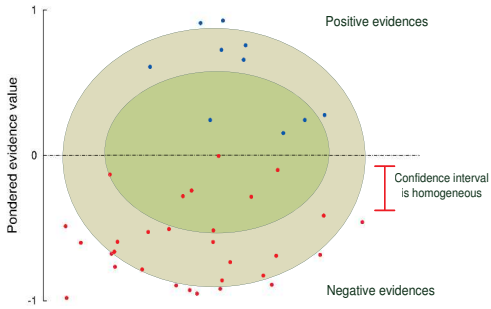
(a) Confidence interval is heterogeneous, denying the attack is not reliable



(b) Confidence interval is heterogeneous, confirming the attack is not reliable



(c) Confidence interval is homogeneous, thus the attack is reliably denied



(d) Confidence interval is homogeneous, thus the attack is reliably confirmed

Figure 4.14: Employing the confidence interval as a measure of detection reliability.

It was aforementioned that when the confidence interval is heterogeneous, i.e., it contains, both, negative and positive values, more evidences are gathered so as to obtain a homogeneous interval. More gathered and processed evidences make the corresponding confidence interval becomes narrower, i.e., its precision increases. This leads mostly[54] to a homogeneous interval, which terminates the diagnosis. However, when the inconsistency of evidences is high, many evidences, and consequently a long diagnosis, are required before obtaining a homogeneous interval. More precisely, if the difference between the positive and the negative pondered evidences is always near to 0 even after gathering new evidences, then the confidence interval becomes narrower but it still contain both negative and positive values. In such cases, continuing gathering blindly more evidences wastes resources. To tackle this issue, we propose to use a precision threshold p_{thr},

[54]Supposing that all evidences in the network have been gathered, if the diagnostic result is 0, which means that the positive and negative evidences are completely equivalent, then it is not possible to decide whether the suspicious node is an intruder or not.

which reflects the precision of the confidence interval. If the sampling error falls below p_{thr}, i.e., the estimated confidence interval achieves the required precision, then the diagnosis terminates regardless whether this interval is homogeneous or not. When the diagnosis is terminated with a confidence interval still heterogeneous, then the suspicious node is not confirmed neither as an intruder nor as a legitimate node (Expression 4.10). However, its trust value is decreased. The value of p_{thr} is an operational parameter that is specified by the end user. The lower the value of p_{thr}, the higher the detection accuracy, but the larger the resources consumption and diagnosis duration.

$$\begin{cases} I \text{ is an intruder} & \text{if } Detect_{A,I} + \varepsilon \leq 0 \\ I \text{ is a legitimate node} & \text{if } Detect_{A,I} - \varepsilon > 0 \\ \text{More evidences are required} & \text{if } (Detect_{A,I} - \varepsilon \leq 0 \wedge Detect_{A,I} + \varepsilon > 0) \wedge (\varepsilon > p_{thr}) \\ I \text{ is unrecognizable} & \text{if } (Detect_{A,I} - \varepsilon \leq 0 \wedge Detect_{A,I} + \varepsilon > 0) \wedge (\varepsilon \leq p_{thr}) \end{cases} \tag{4.10}$$

4.3.3 Evaluation of the Reliability-based Diagnosis

We evaluate to which extend employing the confidence interval allows to find an efficient trade-off between detection accuracy and resource maintaining. For this purpose, we evaluate the impact of the number of evidences (i.e., the sample size) on the sampling error, and hence on the width of the confidence interval. We further show that estimating the confidence interval enables specifying accurately when the diagnostic result is reliable enough. Thus, the collect of redundant evidences is avoided. In addition, we present the large enhancement, in terms of economizing resources consumption, that is achieved thanks to the employment of the confidence interval in our IDS. The aforementioned evaluations are mathematically performed[55]. We consider a case study where a node initiates a diagnosis so as to detect a link spoofing attack. During this diagnosis, this node sequentially interrogates $4, 8, 12, 16$ and 20 other nodes such that each interrogated node provides only one evidence. In other words, there are five rounds of diagnostic. In each round the diagnostic is calculated based on the up-to-now gathered evidences. Moreover, the corresponding 95% confidence interval is computed[56]. The interrogated nodes are initially provided random trust values. We consider that 25% of the nodes are misbehaving. This means, there is always evidences inconsistency in the five diagnostic rounds.

Figure 4.15 shows the evolution of the sampling error along with the increase of the number of evidences. The sampling error decreases as long the number of used evidences is increased. In fact, there is a dramatical decline of the sampling error when the number of used evidences rises from 4 to 8. After that, this decline continues but in a slight manner. This means that increasing the gathered and processed evidences always leads to reduce the sampling error and thus, the precision of the confidence interval goes up. However, after a specific number of evidences (in our scenario 8), the precision enhancement falls down until it becomes negligible. But, when the evidences gathering should be stopped? To answer this question, we analyze the diagnostic result and the corresponding confidence interval at each diagnostic round (Figure 4.16). The

[55] Integrating the confidence interval into our IDS is ready (Appendix C.3) and will be soon evaluated in our environment of simulation.

[56] Since the number of evidences, i.e., the sample size, is less than 30 then t-distribution is used to compute the sampling error.

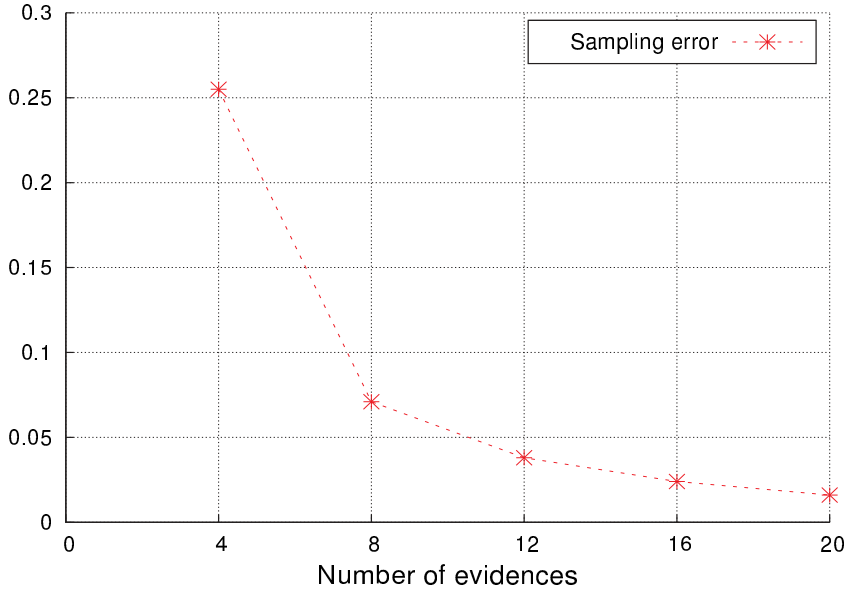


Figure 4.15: Evolution of the sampling error during the confidence interval establishment.

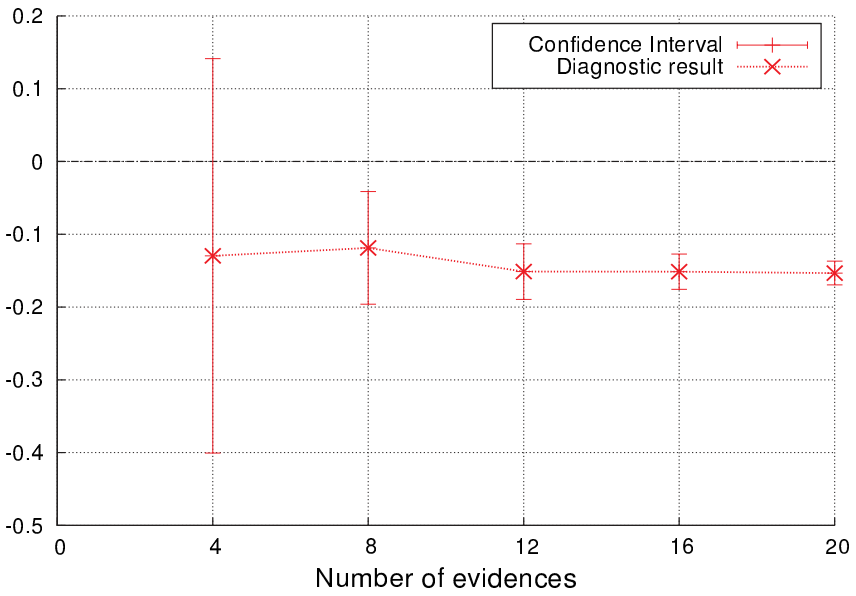


Figure 4.16: Evaluation of the detection reliability based on the confidence interval.

diagnostic result has always a negative value that confirms the existence of an attack. Recall that the diagnostic result ranges from -1 to 1 whereas a negative (resp. positive) result means that the suspicious node is an intruder (resp. a well-behaving node). When the diagnostic result is based on only 4 evidences, the corresponding confidence interval is heterogeneous (i.e., it contains both negative and positive values) and the diagnosis is not yet reliable. Based on 8 or more evidences makes the confidence interval of the diagnostic result homogeneous and the existence of an attack can be reliably confirmed from the second round of diagnosis, i.e., with 8 evidences. Moreover, the confidence interval becomes narrower, i.e., its precision rises, along with the number of evidences. But, the final result (i.e., an attack has taken place) will not be changed. Therefore, gathering more than 8 evidences is a waste of resources. To illustrate the improvement in resources consumption, we study the impact of employing the confidence interval on the number of detection-related messages that are exchanged in the network. Figure 4.17 shows the number of messages that are necessary to gather the evidences under three scenarios: (i) the node charged of detection diagnoses about one suspicious case, (ii) 10 independent suspicious cases, and (iii) 20 independent suspicious cases. Under the three scenarios, we propose that

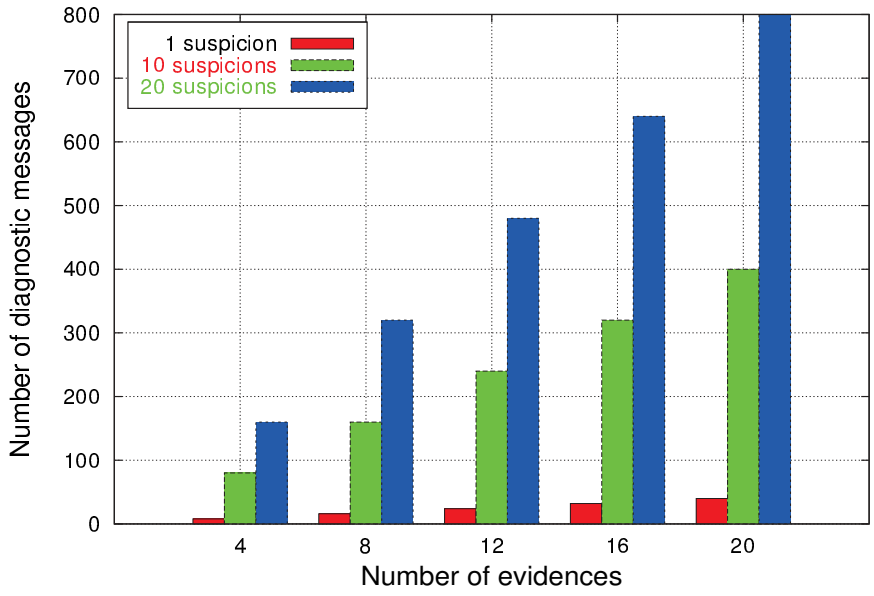


Figure 4.17: Economizing detection messages thanks to confidence interval.

the gathering process is optimal such that gathering one evidence requires only two messages (request/answer). This means there is no consideration for message dropping and re-sending due to, e.g., the collision in the network. For instance, terminating the diagnosis after 8 evidences instead of 20 economizes, at minimum, $24,240,480$ messages when there is $1, 10, 20$ suspicious cases respectively.

4.4 Summary

In this chapter, we tackled three key challenges facing our IDS: (i) its vulnerability to *blackmail* attack, (ii) the necessity of finding a compromise between detection accuracy and resources con-

sumption, and (iii) the necessity of guaranteeing the reliability of detection.
During the intrusion diagnostic, misbehaving node may provide incorrect evidences. Thus, the diagnostic may blame (resp. exonerate) a legitimate node (resp. an intruder). In order to make the diagnostic robust against such falsified evidences, an entropy-based trust model has been proposed and integrated into our IDS. Indeed, an evidence is first pondered by the trustworthiness of its source before being used in the diagnostic. Thus, the evidences coming from distrustful (resp. trustful) nodes have less (resp. more) impact on the diagnostic result. The trustworthiness of a node is increased (resp. decreased) each time this latter provides a correct (resp. an incorrect) diagnostic-related evidence. Therefore, the more the node misbehaves and provides incorrect evidences, the lower is its trustworthiness, and hence the less is its damaging impact on the diagnosis. During the estimation of a node's trustworthiness, its recent participations in the intrusion diagnostics are privileged over the stale ones. This privilege helps in avoiding the effects of the intoxication that takes place when: (i) a legitimate node with a high trust value becomes compromised and starts providing incorrect evidences, or (ii) a malicious node tries to gain the trust of others by providing correct evidences for a while before it starts participating maliciously in the diagnostics. The trustworthiness of a node is also associated with its role in detecting the serious attacks. More precisely, the higher the risk of an attack, the larger the increase (resp. decrease) of the trustworthiness of the nodes that have participated in detecting (resp. hiding) this attack. This means that the preventative nature of our IDS is amplified along with the level of danger in the network. The risk of an attack increases as: (i) it evolves and approaches its final goals, and (ii) more vulnerabilities or attacks can be derived from it. The evaluation of trust-based intrusion detection shows that the impact of *blackmail* attacks fades along with the evolution of trust relations between the nodes.
We are inspired from the inferential statistics so as to estimate the opinion of the entire network about a suspicious node. Indeed, the confidence interval of the diagnostic result has been employed so as to achieve a compromise between resources consumption and detection reliability. During a diagnostic, the gathered evidences are used to compute the confidence interval of the obtained diagnostic result. This interval is computed according to a specific confidence level that reflects the required detection accuracy. It further contains the true diagnostic result that would be obtained if all the evidences in the network are gathered. More evidences are gradually gathered and processed as long the corresponding confidence interval is heterogeneous, i.e., it contains, both, values that confirm the attack and values that deny it. Increasing the number of used evidences lead to rise the precision of the confidence interval, and further the reliability of the diagnostic result. When the confidence interval becomes homogeneous, i.e., all its values confirm (or deny) the attack, the diagnostic result is considered sufficiently reliable. Here, no more evidences are gathered since they will not changed the obtained result. However, in the presence of highly contrary evidences, obtaining a homogeneous confidence interval may require gathering many evidences. To override this problem, we proposed to terminate the diagnostic when the precision of the corresponding confidence interval rises over a predefined threshold. Overall, the confidence interval enables our IDS to specify accurately when the required reliability is achieved, hence more evidences are redundant and should not be gathered. Moreover, employing the confidence interval guarantees the reliability of detection. The evaluation of the proposed statistical-based diagnostic proves that the confidence interval reduces significantly the

amount of the diagnostic-related messages.

CHAPTER 5

CONCLUSION

In this thesis, I have proposed a **l**ightweight and robust **i**ntrusion **d**etection system for *ad hoc* **r**outing protocols (LIDR for short). More precisely, this IDS aims at detecting the attacks targeting the OLSR routing protocol. For this purpose, it extracts evidences and signs of attacks from the local routing logs. Then it correlates them with the predefined intrusion signatures. An attack is confirmed when a sequence of observed evidences matches an intrusion signature. If the local evidences are not sufficient to confirm or affirm an attack, an in-depth diagnosis is triggered. This in-depth diagnosis consists in asking other nodes to gather more evidences about the suspicious node(s). An evidence provided by another node is first weighted according to the trustworthiness of its provider. The trustworthiness of a node is established by an entropy-based and risk-aware trust system that increases (resp. decreases) the trustworthiness of a node each time this node provides correct (resp. incorrect) evidence. The amount of such increase/decrease is related to the risk of the attack that reflects both the evolution of this attack and its possible consequences/damages. In addition, the confidence interval of the diagnosis is used as a measure of reliability that serves in determining whether the suspicious node is an intruder or not. In particular, during the diagnostic phase, gathering and processing more evidences is repeated as long as the corresponding confidence interval is heterogeneous, i.e., some of its values confirm the attack while others affirm it. When all the values in the confidence interval confirm (resp. affirm) the attack, then the diagnosis is terminated by confirming the suspicious node as an intruder (resp. a well-behaved node). Some experiments have been realized so as to evaluate the performance of this IDS and its suitability for MANET. The proposed IDS is well adapted to the dynamic topology of MANET and offers a high detection accuracy. In fact, this IDS has a distributed nature; each node is charged with analyzing its local logs and protecting itself. In other words, all nodes perform similar functions and there is no need for central node(s) to be charged with special functions. This means that this IDS does not contain failure points resulting when, for example, a central node moves away or leaves the network. In addition to ensuring its own security, each node cooperates with others to exchange and to match the evidences with the intrusion signatures. This cooperation offers a global point of view about the activities of the node under investigation, and hence it enhances significantly the detection accuracy. Furthermore, considering the temporal dimension of the evidences during the diagnosis

enables detecting the long-term attacks. The detection accuracy of the proposed IDS has been evaluated in a simulated MANET wherein different scenarios of density and mobility are applied. The analysis of the results shows that the density of the network has a small impact on the performance of the proposed IDS. For instance, when the density[57] reaches up to 16, the average of the detection rate exceeds 81% and the average of the false alarms rate is less than 6%. Similarly to the density, the mobility has a small impact on the false alarm rate: increasing the mobility leads to a small decrease in the detection rate. The detection rate remains relatively high; it exceeds 70% for a node speed equals to 8m/s (i.e., 28.8 km/h).

The proposed IDS efficiently saves the available resources, e.g., battery life, computing capabilities, and bandwidth. Hence, it is easily tolerated by the limited-resources devices. This conservation results from a threefold reason:

- The restriction of, both, the number and the duration of the in-depth diagnosis. In fact, the in-depth diagnosis includes gathering and correlating global evidences and therefore it is a costly operation in terms of resources. Therefore, this diagnosis is carefully planned. In practice, the evidences are categorized according to their gravity so that only the evidences with a sufficient level of gravity activate the in-depth diagnosis. In addition, the in-depth diagnosis is completed as soon as an evidence, which can confirm (or affirm) the attack, is provided.

- Minimizing the resources that are consumed due to evidence gathering and processing. In fact, the proposed IDS extracts the evidences of attacks from local logs that are carefully selected. It avoids traffic sniffing because this latter leads to: (i) a permanent strain of energy as a result of keeping the network interface in the promiscuous mode, and (ii) a significant strain of memory and CPU processing as a result of applying packet-level analysis on all the packets, including those which are not related to the diagnosis.

- Avoiding the waste of resources resulting from gathering and processing the redundant evidences. The redundant evidences are those that do not change the diagnostic result (or for which the change is negligible). In order to identify such evidences, the confidence interval is employed as a measure of detection reliability. A reliable diagnostic will not be changed even though new evidences are gathered and processed, and hence such evidences become redundant. Therefore, when the confidence interval refers to a reliable diagnosis, this latter is directly terminated.

Results of computational experiments show that the traffic overhead, which is generated by the proposed IDS, is very limited and could be ignored compared to the routing-related traffic. In addition, this IDS imposes a limited increase of the memory usage that ranges from 17.6MB to 22.2MB. Furthermore, the employment of the confidence interval shows that more than 50% of gathered evidences are redundant and can thus be avoided.

The proposed IDS offers, for the first time, an indicator of the detection reliability. This indicator is based on the confidence interval of the diagnosis and is calculated according to the required detection accuracy. It helps in eliminating the doubt about the detection result,

[57] Recall that the network density reflects the average number of the neighbors count

especially in the presence of inconsistent evidences. Thus, the detection-related decisions, e.g., launching a counter measure, are safely made.

The proposed IDS is also protected against misbehaving nodes which aim at spoiling the detection by providing falsified evidences. This protection results from coupling this IDS with the entropy-based trust model. Recall that this trust model establishes and maintains the trust relationships between the nodes in the detection context. It further amplifies its preventive nature when the level of risk is high. Such amplification is realized by largely decreasing the trustworthiness of the misbehaving nodes. In addition, the trust model resists to intoxication that happens when: (i) a legitimate node with a high trust value is compromised and starts providing incorrect evidences, or (ii) a malicious node tries to gain the trust of others before beginning to provide falsified evidences. This resistance is due to the fact that the newest evidences are privileged. The performance evaluation of the trust model shows that the trust relationships are correctly evolving. More precisely, the misbehaving nodes continually lose their trustworthiness, and hence their impact on the detection gradually disappears. As a result, the proposed IDS is able to detect the attacks even when the misbehaving nodes constitute up to 43% of the network.

Perspectives and Future Works: our evaluation highlighted the performances of our IDS, as well as its high suitability for the limited-resources devices. Nevertheless, this evaluation is intended to be enriched with additional attacks threatening OLSR and potentially other routing protocols, e.g., AODV. Moreover, other evaluation criteria such as the energy consumption or the CPU processing[58] will prove the lightweight of the detection. Ongoing work includes new series of experiments focused on measuring the reliability, i.e., the confidence interval, under a real MANET. For this purpose, I am using several Nexus S smartphones[59] embedding the Android operating system[60]. These smartphones are configured to support *ad hoc* networking, and, a compatible version of OLSR [61] has been installed. It is worth to mention that the implementation of our IDS, as used during the simulation, can be transparently transferred to the smartphones. However, there are still several challenges of the performance evaluating in a real MANET. For instance, this includes making the smartphones following the required model(s) of mobility.

In the near future, two main research directions are envisioned. The former lies in tuning the risk evaluation depending on the context and the latter refers to improving the event sampling by taking into account predefined criteria. More precisely, we have proposed an indicator of the attack risk that depends of the degree of evolution of the attack and the predictable vulnerabilities. The concept of risk is inadequately treated in MANET; to the best of our knowledge, this is the first time the risk of the attacks targeting the *ad hoc* routing protocols is proposed. One idea would be to relate the attack risk with the operating environment and the nature of the mission (e.g., military purpose). This implies to consider a subjective dimension during the risk assessment. Another improvement would consist in providing, instead of the random and uniform sampling, a stratified random sampling [8]. In this sampling, the nodes are stratified

[58] Previous attempts to measure the CPU usage failed due to an bug in the LXC virtual machine. As planned in the near future, the use of real smartphones will permit to monitore the CPU processing.

[59] www.samsung.com/us/mobile/cell-phones/GT-I9020FSTTMB

[60] http://www.android.com

[61] www.olsr.org/releases/0.5/olsrd-0.5.6-android-samsung-galaxy.tgz

into n groups according to e.g., the trust value, the neighborhood, the battery level, etc. During the diagnosis, the sample is selected by gathering evidences from the n groups according to predefined percentages $(p_1\%, p_2\%, ..., p_n\%)$. However, the selection inside a group is randomly. As a result, the stratified random sampling ensures that the sample has a certain combination and respects predefined conditions. For instance, it enables having a sample such that the majority of evidences are gathered from the trustful nodes. At the same time, every node, even those that are, for instance, distrustful, still has the chance to offer evidences for the diagnosis.

Future Tracks: According to the International Telecommunication Unit (ITU) [165, 166], the Internet of Things (IoT) can be envisioned as providing all the objects and devices with the ability to connect to a network termed IoT. Thus, object and devices can exchange information, produce/consume services within a digital world. The IoT is one of the most promising technologies that would change many domains, e.g. logistics, business/process management, assisted living, and e-health. But several social and technical challenges remain to fulfill the idea of IoT, among them the security [167] which is of prime importance. Security solutions for IoT networks should take into account, among others: (i) the existence of heterogeneous routing technologies where the multi-hop *ad hoc* routing occupies a non negligible part [168], and (ii) the severe constraints on, both, the computing capacity and the energy of the many things/objects [167]. Since *ad hoc* routing and limited resources are key elements in our IDS, it would be interesting to study the adapting of our system for detecting the attacks in IoT networks.
Meanwhile issues e.g., gathering or correlating a huge amount of evidences are shared not only with the IoT but also with **G**rid and **C**loud computing [169]. Evidences are gathered from the different, usually geographically-separated, administrative domains. Consequently, detecting attacks in such networks does not only consume many resources but also requires a long time. Note that the longer the duration of detection, the larger the damages of the attack. The use of confidence interval as a measure of detection reliability ensures minimizing the number of gathered evidences, and henceforth the duration of detection, along with maintaining the required level of accuracy. We believe that the confidence interval should be considered during the correlation in a Grid Security Operation Center (GSOC) [170], a security event manager for grid computing proposed in FEMTO-ST/DISC laboratory[62] .

[62] www.femto-st.fr/fr/Departements-de-recherche/DISC/Presentation/

List of Publications

INTERNATIONAL JOURNALS :

Mouhannad Alattar, Françoise Sailhan, and Julien Bourgeois
On Lightweight Intrusion Detection : Modeling and Detecting Intrusions dedicated to OLSR Protocol. International Journal of Distributed Sensor Networks, Special Issue on Intrusion Detection and Security Mechanisms for Wireless Sensor Networks (IDS), 2013, Note : 18 pages, To appear.

INTERNATIONAL CONFERENCES :

Mouhannad Alattar, Françoise Sailhan, and Julien Bourgeois
Log-based Intrusion Detection for MANET. In IWCMC'12, 8-th IEEE Int. Conf. on Wireless Communications and Mobile Computing, Limassol, Cyprus, pages 697–702, August 2012. IEEE Computer Society.

Mouhannad Alattar, Françoise Sailhan, and Julien Bourgeois
Reliable Trust Estimation in AD HOC Networks using Confidence Interval. In SecurIT 2012, 1st Int. Conf. on Security of Internet of Things, Kerala, India, pages ***–***, August 2012. ACM Press. Note : 8 pages.

Mouhannad Alattar, Françoise Sailhan, and Julien Bourgeois
Trust-enabled Link Spoofing Detection in MANET. In WWASN 2012, 9-th Workshop on Wireless Ad Hoc and Sensor Networks, workshop of ICDCS 2012, the 32-nd IEEE Int. Conf. on Distributed Computing Systems, Macau, China, pages 237–244, June 2012.

NATIONAL CONFERENCES :

Mouhannad Alattar, Françoise Sailhan, and Julien Bourgeois
Détection de liens falsifiés basée sur les traces en réseau mobile ad hoc. In JDIR'11, 12èmes Journées Doctorales en Informatique et Réseaux, Session algorithmes sur les graphes, Belfort, France, November 2011.

Mouhannad Alattar, Françoise Sailhan, and Julien Bourgeois
Modeling and Detecting Intrusions in ad hoc Network Routing Protocols. 3SL, Journée Sécurité des Systémes et Sureté des Logiciels, Saint-Malo, France, May 2011.

RESEARCH REPORTS :

M. Alattar, J. Bourgeois, and F. Sailhan
Log based link spoofing detection in MANET, CEDRIC laboratory, CNAM-Paris, France, February 2012.

BIBLIOGRAPHY

[1] C.H. Tseng, S. Tao, P. Balasubramanyam, and al. A specification-based intrusion detection model for OLSR. In *RAID*. 2008.

[2] C. Tseng, P. Balasubramanyam, R. Limprasittiporn, and al. A specification-based intrusion detection system for AODV. In *ACM SASN workshop*, 2003.

[3] P. Yi, Y. Zhong, and S. Zhang. A novel intrusion detection method for mobile ad hoc networks. In *Advances in Grid Computing - EGC 2005*, volume 3470. Springer Berlin / Heidelberg, 2005.

[4] G. Vigna, S. Gwalani, K. Srinivasan, and al. An intrusion detection tool for AODV-based ad hoc wireless networks. In *ACSAC*, 2004.

[5] R.S. Puttini, J.M. Percher, L. Mé, and al. A modular architecture for distributed ids in manet. In *Proceedings of the 2003 international conference on Computational science and its applications: PartIII*, 2003.

[6] F.Cuppens, N.Cuppens-Boulahia, S.Nuon, and al. Property based intrusion detection to secure OLSR. In *IEEE ICWMC*, 2007.

[7] F.J. Gravetter and L.B. Wallnau. *Essentials of statistics for the behavioral sciences*. Wadsworth Publishing Company, 2008.

[8] M.J. Smithson. *Statistics with confidence: an introduction for psychologists*. Sage Publications Limited, 2000.

[9] T. Clausen and P. Jacquet. Optimized link state routing protocol (OLSR). IETF experimental RFC 3626, Oct. 2003.

[10] B.C. Cheng and R.Y. Tseng. A context adaptive intrusion detection system for manet. *Computer Communications*, 34(3), 2011.

[11] A. Adnane, C. Bidan, and R.T.S. Junior. Trust-based countermeasures for securing olsr protocol. In *IEEE CSE*, 2009.

[12] P. Mohapatra and S. Krishnamurthy. *Ad hoc networks: technologies and protocols*. Springer, 2004.

[13] J. Loo, J.L. Mauri, and J.H. Ortiz. *Mobile Ad Hoc Networks: Current Status and Future Trends*. Taylor & Francis Group, 2012.

[14] S. Sen and J.A. Clark. Intrusion detection in mobile ad hoc networks. In *Guide to Wireless Ad Hoc Networks*. Springer London, 2009.

[15] S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic. *Mobile ad hoc networking*. Wiley-IEEE press, 2004.

[16] J.P. Hubaux, L. Buttyán, 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 & computing*, 2001.

[17] B. Mukherjee, L.T. Heberlein, and K.N. Levitt. Network intrusion detection. *Network, IEEE*, 8(3), 1994.

[18] H. Debar and J. Viinikka. Intrusion detection: Introduction to intrusion detection and security information management. *Foundations of security analysis and design III*, 2005.

[19] B. Fraser. Site security handbook. IETF experimental RFC 2196, Sept. 1997.

[20] A. Adelsbach, C. Cachin, S. Creese, and al. Conceptual model and architecture of maftia. 2003.

[21] P. Brutch and C. Ko. Challenges in intrusion detection for wireless ad-hoc networks. In *Proceedings of the Symposium on Applications and the Internet Workshops (SAINT'03 Workshops)*, 2003.

[22] F.H. Wai, Y. Nwe Aye, and N.H. James. Intrusion detection in wireless ad-hoc networks. In *CS4274 Introduction to mobile computing*, 2004.

[23] C. Xenakis, C. Panos, and I. Stavrakakis. A comparative evaluation of intrusion detection architectures for mobile ad hoc networks. *Computers & Security*, 30(1), 2011.

[24] V. Cerf. The internet activities board. IETF experimental RFC 1120, May 1990.

[25] S.Corson and J.Macker. Mobile ad hoc networking (manet): Routing protocol performance issues and evaluation considerations. IETF RFC2501, Jan. 1999.

[26] E. Atallah and S. Chaumette. A smart card based distributed identity management infrastructure for mobile ad hoc networks. In *Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems*. Springer Berlin Heidelberg, 2007.

[27] H.C. Jang, Y.N. Lien, and T.C. Tsai. Rescue information system for earthquake disasters based on manet emergency communication platform. In *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly*, IWCMC '09. ACM, 2009.

[28] A. Vasiliou and A.A. Economides. Manets for environmental monitoring. In *Telecommunications Symposium, 2006 International*, Sept. 2006.

[29] B.C. Seet, G. Liu, B.-S.g Lee, and al. A-star: A mobile ad hoc routing strategy for metropolis vehicular communications. In *Proc. NETWORKING 2004*, 2004.

[30] S. Chaumette, R. Laplace, C. Mazel, and A. Godin. Secure cooperative ad hoc applications within uav fleets position paper. In *Military Communications Conference. MILCOM 2009. IEEE*, 2009.

[31] V. Karyotis and S. Papavassiliou. Risk-based attack strategies for mobile ad hoc networks under probabilistic attack modeling framework. *Comput. Netw.*, 51(9), 2007.

[32] S. Ruffino, P. Stupar, T. Clausen, and S. Singh. Connectivity scenarios for manet. IETF draft-ruffino-autoconf-conn-scenarios-00, Jan. 2006.

[33] M. Felegyhazi, J.P. Hubaux, and L. Buttyan. Nash equilibria of packet forwarding strategies in wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 5, 2006.

[34] J. Wang, R. Prasad, and I. Niemegeers. Self-configuration in manets: Different perspectives. In *Self-Organizing Systems*, volume 4725. Springer Berlin / Heidelberg, 2007.

[35] J. Tyan and Q.H. Mahmoud. A network layer based architecture for service discovery in mobile ad hoc networks. In *Electrical and Computer Engineering, 2004. Canadian Conference on*, 2004.

[36] S. Helal, N. Desai, V. Verma, and C. Lee. Konark-a service discovery and delivery protocol for ad-hoc networks. In *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, 2003.

[37] O. Ratsimor, D. Chakraborty, A. Joshi, and T. Finin. Allia: Alliance-based service discovery for ad-hoc environments. In *Proceedings of the 2nd international workshop on Mobile commerce*, 2002.

[38] M. Mohsin and R. Prakash. Ip address assignment in a mobile ad hoc network. In *MILCOM 2002. Proceedings*, 2002.

[39] A. Yousef, P. Driess, A. Diab, and A. Mitschele-Thiel. Comparative analysis of lha, manetconf and prophet stateful address auto-configuration protocols in ad hoc networks. In *Proceedings of the 6th ACM symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*. ACM, 2009.

[40] H. Zhou, L.M. Ni, and M.W. Mutka. Prophet address allocation for large scale manets. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, 2003.

[41] C. Tschudin, P. Gunningberg, H. Lundgren, and E. Nordström. Lessons from experimental manet research. *Ad Hoc Networks*, 3(2), 2005.

[42] S. Sesay, Z. Yang, and J. He. A survey on mobile ad hoc wireless network. *Information Technology Journal*, 3(2), 2004.

[43] Y.S. Chen, C.S. Hsu, and S.J. Jan. A shoelace-based qos routing protocol for mobile ad hoc networks using directional antenna. *Wireless Personal Communications*, 54, 2010.

[44] G. Holland and N. Vaidya. Analysis of tcp performance over mobile ad hoc networks. *Wirel. Netw.*, 8(2/3), March 2002.

[45] C. K. Toh. *Ad Hoc Mobile Wireless Networks: Protocols and Systems*. Prentice Hall PTR, 2001.

[46] C. Perkins, E Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing. IETF experimental RFC 3561, Jul. 2003.

[47] C.E. Perkins and E.M. Royer. Ad-hoc on-demand distance vector routing. In *Mobile Computing Systems and Applications. Proceedings. WMCSA '99. Second IEEE Workshop on*, 1999.

[48] C. Perkins and I. Chakeres. Dynamic manet on-demand (aodvv2) routing. IETF Active Internet-Draft draft-ietf-manet-dymo-22, Mar. 2012.

[49] D. Johnson, Y. Hu, and D. Maltz. The dynamic source routing protocol (dsr) for mobile ad hoc networks for ipv4. IETF experimental RFC 4728, Feb. 2007.

[50] D.B. Johnson and D.A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, volume 353. 1996.

[51] P. Jacquet, P. Muhlethaler, T. Clausen, and al. Optimized link state routing protocol for ad hoc networks. In *Multi Topic Conference. IEEE INMIC2001. Technology for the 21st Century. Proceedings. IEEE International*, 2001.

[52] C.E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *ACM SIGCOMM Computer Communication Review*, 1994.

[53] R. Ogier, F. Templin, and M. Lewis. Topology dissemination based on reverse-path forwarding (tbrpf). IETF experimental RFC 3684, february 2004.

[54] L. Wang and S. Olariu. A two-zone hybrid routing protocol for mobile ad hoc networks. *Parallel and Distributed Systems, IEEE Transactions on*, 15(12), 2004.

[55] M. Jiang, J. Li, and Y.C. Tay. Cluster based routing protocol(cbrp). IETF draft-ietf-manet-cbrp-spec-01, Jul. 1999.

[56] P.Albers, O.Camp, J.Percher, and al. Security in ad hoc networks: a general intrusion detection architecture enhancing trust based approaches. 2002.

[57] Z. Zhao, H. Hu, G.J. Ahn, and R. Wu. Risk-aware mitigation for manet routing attacks. *Dependable and Secure Computing, IEEE Transactions on*, 9(2), 2012.

[58] A. Irshad, W. Noshairwan, M. Shafiq, and al. Security enhancement for authentication of nodes in manet by checking the crl status of servers. In *Security-Enriched Urban Computing and Smart Grid*. 2010.

[59] T. Eissa, S. Razak, and M. Ngadi. Towards providing a new lightweight authentication and encryption scheme for manet. *Wireless Networks*, 17, 2011.

[60] N.A. Jourabbaf and A. Movaghar. Id-nac: Identity-based network access control for manets. In *Networks, 2008. ICON 2008. 16th IEEE International Conference on*, 2008.

[61] H. Lee and Y. Mun. Design of modified cga for address auto-configuration and digital signature in hierarchical mobile ad-hoc network. In *Information Networking. Advances in Data Communications and Wireless Networks*. Springer Berlin / Heidelberg, 2006.

[62] S. Xu, Y. Mu, and W. Susilo. Online/offline signatures and multisignatures for aodv and dsr routing security. In *Information Security and Privacy*. Springer Berlin / Heidelberg, 2006.

[63] C. Li, Z. Wang, and C. Yang. Seaodv: A security enhanced aodv routing protocol for wireless mesh networks. In *Transactions on Computational Science XI*. Springer Berlin / Heidelberg, 2010.

[64] I. Doh, K. Chae, H. Kim, and K. Chung. Security enhancement mechanism for ad-hoc olsr protocol. In *Information Networking. Advances in Data Communications and Wireless Networks*. Springer Berlin / Heidelberg, 2006.

[65] P. Kunwar, S. Sofat, and D. Bansal. Comparison of secure olsr routing protocol. *International Journal of Engineering Science*, 3, 2011.

[66] S. Sen, J.A. Clark, and J.E. Tapiador. Security threats in mobile ad hoc networks, 2009.

[67] D.E. Denning. An intrusion-detection model. *IEEE Transactions on software engineering*, 13, 1987.

[68] Y. Zhang and W. Lee. Intrusion detection in wireless ad-hoc networks. In *ACM MobiCom conference*. ACM, 2000.

[69] R. Heady, G. Luger, A. Maccabe, and M. Servilla. The architecture of a network level intrusion detection system. Technical report, LA-SUB–93-219, Los Alamos National Lab., NM (United States); New Mexico Univ., Albuquerque, NM (United States). Dept. of Computer Science, 1990.

[70] A.A. El Kalam, J.y Briffaut, C. Toinard, and M. Blanc. Intrusion detection and security policy framework for distributed environments. In *Collaborative Technologies and Systems. Proceedings of the 2005 International Symposium on*, 2005.

[71] W. Wang, H. Man, and Y. Liu. A framework for intrusion detection systems by social network analysis methods in ad hoc networks. *Security and Communication Networks*, 2(6), 2009.

[72] S. Marti, T.J. Giuli, K. Lai, and al. Mitigating routing misbehavior in mobile ad hoc networks. In *ACM Mobicom conference*, 2000.

[73] A.P. Lauf, R.A. Peters, and W.H. Robinson. A distributed intrusion detection system for resource-constrained devices in ad-hoc networks. *Ad Hoc Networks*, 8(3), 2010.

[74] D. Sterne, P. Balasubramanyam, D. Carman, and al. A general cooperative intrusion detection architecture for manets. 2005.

[75] S.A. Razak, S.M. Furnell, N.L. Clarke, and P.J. Brooke. Friend-assisted intrusion detection and response mechanisms for mobile ad hoc networks. *Ad Hoc Networks*, 6(7), 2008.

[76] Y. Huang and W. Lee. Attack analysis and detection for ad hoc routing protocols. In *Recent Advances in Intrusion Detection*, volume 3224. Springer Berlin / Heidelberg, 2004.

[77] J.F.C. Joseph, A. Das, B.C. Seet, and B.S. Lee. Opening the pandora's box: Exploring the fundamental limitations of designing intrusion detection for manet routing attacks. *Computer Communications*, 31(14), 2008.

[78] T. Joachims. *Making large-scale support vector machine learning practical*. MIT Press Cambridge, 1999.

[79] Y. Zhang, W. Lee, and Y.A. Huang. Intrusion detection techniques for mobile wireless networks. *Wirel. Netw.*, 9(5), 2003.

[80] W.W. Cohen. Fast effective rule induction. In *ICML*, 1995.

[81] H. Deng, R. Xu, J. Li, and al. Agent-based cooperative anomaly detection for wireless ad hoc networks. In *Parallel and Distributed Systems. ICPADS 2006. 12th International Conference on*, 2006.

[82] Y. Yajima and T.F. Kuo. Efficient formulations for 1-svm and their application to recommendation tasks. *Journal of Computers*, 1(3), 2006.

[83] S. Kurosawa, H. Nakayama, N. Kato, and al. Detecting blackholes attack on AODV-based mobile ad hoc networks by dynamic learning method. *International journal of network security*, 5(3), 2007.

[84] E.F. Schuster. Estimation of a probability density function and its derivatives. *The Annals of Mathematical Statistics*, 40(4), 1969.

[85] Y. Huang, W. Fan, W. Lee, and al. Cross-feature analysis for detecting ad-hoc routing anomalies. In *IEEE ICDCS*, 2003.

[86] J.R. Quinlan. *C4.5: programs for machine learning*. M. Kaufmann, 1993.

[87] J.B.D. Cabrera, C. Gutierrez, and R.K. Mehra. Ensemble methods for anomaly detection and distributed intrusion detection in mobile ad-hoc networks. *Information Fusion*, 9, 2008.

[88] S. Bose, S. Bharathimurugan, and A. Kannan. Multi-layer integrated anomaly intrusion detection system for mobile adhoc networks. In *Signal Processing, Communications and Networking, 2007. ICSCN '07. International Conference on*, 2007.

[89] LLC Books. *Statistical Classification: Naive Bayes Classifier, Support Vector Machine, Pattern Recognition, Linear Classifier*. General Books LLC, 2010.

[90] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2), June 1993.

[91] B. Sun, K. Wu, and U.W. Pooch. Routing anomaly detection in mobile ad hoc networks. In *Computer Communications and Networks. ICCCN 2003. Proceedings. The 12th International Conference on*, 2003.

[92] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28, Jan 1980.

[93] B. Sun, K. Wu, Y. Xiao, and R. Wang. Integration of mobility and intrusion detection for wireless ad hoc networks. *International Journal of Communication Systems*, 20, June 2007.

[94] A. Mitrokotsa and C. Dimitrakakis. Intrusion detection in manet using classification algorithms: The effects of cost and model selection. *Ad Hoc Networks*, (0), 2012.

[95] D.W. Ruck, S.K. Rogers, M. Kabrisky, and al. The multilayer perceptron as an approximation to a bayes optimal discriminant function. *Neural Networks, IEEE Transactions on*, 1(4), 1990.

[96] N. Bshouty and P. Long. Linear classifiers are nearly optimal when hidden variables have diverse effects. *Machine Learning*, 86, 2012.

[97] C..E. Rasmussen. The infinite gaussian mixture model. In *In Advances in Neural Information Processing Systems 12*, 2000.

[98] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2*, 1995.

[99] G. A. Jacoby and N. J. Davis. Mobile host-based intrusion detection and attack identification. *Wireless Commun.*, 14(4), August 2007.

[100] W. Wang, H. Wang, B. Wang, and al. Energy-aware and self-adaptive anomaly detection scheme based on network tomography in mobile ad hoc networks. *Information Sciences*, (0), 2012.

[101] A. Tsang, M. Coates, and R.D. Nowak. Network delay tomography. *IEEE Transactions on Signal Processing*, 51, 2003.

[102] T.HEIZO and F.KIKURO. Self-organization map and its application. *Brain & Neural Networks*, 10, 2003.

[103] L.C. Freeman, S.P. Borgatti, and D.R. White. Centrality in valued graphs: A measure of betweenness based on network flow. *Social Networks*, 13(2), 1991.

[104] S.Buchegger and J-Y.Le Boudec. Performance analysis of the confidant protocol: Cooperation of nodes - fairness in dynamic ad-hoc networks). In *3rd ACM international symposium on Mobile ad hoc networking & computing*, 2002.

[105] P. Michiardi and R. Molva. Core: A collaborative reputation mechanism to enforce node cooperation. In *in Mobile Ad Hoc Networks. Communication and Multimedia Security*, 2002.

[106] S. Bansal and M. Baker. Observation-based cooperation enforcement in ad hoc networks. *CoRR*, cs.NI/0307012, 2003.

[107] M-Y Su. Prevention of selective black hole attacks on mobile ad hoc networks through intrusion detection systems. *Computer Communications*, 34, 2011.

[108] A.M. Abdalla, I.A. Saroit, A. Kotb, and al. Misbehavior nodes detection and isolation for MANETs OLSR protocol. *Procedia Computer Science*, 3, 2011.

[109] N. Marchang and R. Datta. Collaborative techniques for intrusion detection in mobile ad-hoc networks. *Ad Hoc Networks*, 6(4), 2008.

[110] B. V. Ram Naresh Yadav, B. Satyanarayana, and O. B. V. Ramanaiah. An efficient intrusion detection system for mobile ad hoc networks. In *Emerging Trends in Computing, Informatics, Systems Sciences, and Engineering*, volume 151. Springer New York, 2013.

[111] J.M. Orset, B. Alcalde, and A. Cavalli. An efsm-based intrusion detection system for ad hoc networks. In *Automated Technology for Verification and Analysis*, volume 3707. Springer Berlin / Heidelberg, 2005.

[112] B. Alcalde, A. Cavalli, D. Chen, and al. Network protocol system passive testing for fault management: A backward checking approach. In *Formal Techniques for Networked and Distributed Systems-FORTE 2004*, volume 3235. Springer Berlin / Heidelberg, 2004.

[113] M. Wang, L. Lamont, P. Mason, and al. An effective intrusion detection approach for OLSR MANET protocol. *IEEE ICNP workshop*, 2005.

[114] M. Ayachi, C. Bidan, and N. Prigent. A trust-based ids for the aodv protocol. In *Information and Communications Security*, volume 6476. 2010.

[115] A. Adnane, R.T. Sousa, C. Bidan, and al. Autonomic trust reasoning enables misbehavior detection in OLSR. In *ACM SAP*, 2008.

[116] S. Şen and J.A. Clark. A grammatical evolution approach to intrusion detection on mobile ad hoc networks. In *Proceedings of the second ACM conference on Wireless network security*, 2009.

[117] B. Sun, L. Osborne, Yang Xiao, and S. Guizani. Intrusion detection techniques in mobile ad hoc and wireless sensor networks. *IEEE Wireless Communications*, 40(10), Oct. 2002.

[118] L.M. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *IEEE INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and communications Societies*, 2001.

[119] D. Subhadrabandhu, S. Sarkar, and F. Anjum. A framework for misuse detection in ad hoc networks-part i. *IEEE Journal on Selected Areas in Communications*, 24(2), 2006.

[120] B. Wu, J. Chen, J. Wu, and M. Cardei. A Survey of Attacks and Countermeasures in Mobile Ad Hoc Networks. In *Wireless Network Security*. Springer US, 2007.

[121] S. Sen, J.A. Clark, and J.E. Tapiador. *Security Threats in Mobile Ad Hoc Networks*. Auerbach Publications (CRC Press), 2010.

[122] N. Peng and S. Kun. How to misuse AODV: a case study of insider attacks against mobile ad-hoc routing protocols. *Ad Hoc Netw.*, 3(6), 2005.

[123] P. Jacquet, P. Muhlethaler, A. Qayyum, and al. Optimized link state routing protocol. IETF Internet-Draft draft-ietf-manet-olsr-04.txt, March 2001.

[124] T. Clausen, C. Dearlove, P. Jacquet, and U. Herberg. The optimized link state routing protocol version 2. IETF Active Internet-Draft draft-ietf-manet-olsrv2-15, May 2012.

[125] U. Herberg and T. Clausen. Security issues in the optimized link state routing protocol version 2 (olsrv2). *International Journal of Network Security & Its Applications (IJNSA)*, 2(2), 2010.

[126] Y. Owada, T. Maeno, H. Imai, and K. Mase. Olsrv2 implementation and performance evaluation with link layer feedback. In *IWCMC*, 2007.

[127] O. Gonzalez, M Howarth, and G Pavlou. Detection of packet forwarding misbehavior in mobile ad-hoc networks. In *Wired/Wireless Internet Communications*, volume 4517. Springer Berlin Heidelberg, 2007.

[128] B. Wu, J. Chen, and al. A survey of attacks and countermeasures in mobile ad hoc networks. In *Wireless Network Security*. Springer US, 2007.

[129] C. Adjih, D. Raffo, and P. Mühlethaler. Attacks against OLSR: distributed key management for security. In *OLSR interop workshop*, 2005.

[130] G. Cervera, M. Barbeau, J. Garcia-Alfaro, and E. Kranakis. Mitigation of topology control traffic attacks in olsr networks. In *Risks and Security of Internet and Systems (CRiSIS), 2010 Fifth International Conference on*, 2010.

[131] P.M. Jawandhiya, M.M. Ghonge, M.S. Ali, and al. A survey of mobile ad hoc network attacks. *International Journal of Engineering Science and Technology*, 2, 2010.

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

[133] G.F. Riley and T.R. Henderson. The ns-3 network simulator. In *Modeling and Tools for Network Simulation*. Springer Berlin Heidelberg, 2010.

[134] S. Bhattiprolu, E.W. Biederman, S. Hallyn, and al. Virtual servers and checkpoint/restart in mainstream linux. *SIGOPS Oper. Syst. vol. 42*, 2008.

[135] J. Zhang, J. Xing, and Z. Qin. Taprouter: An emulating framework to run real applications on simulated mobile ad hoc network. In *ANSS*, 2011.

[136] G. Anastasi, E. Borgia, M. Conti, and E. Gregori. Wi-fi in ad hoc mode: a measurement study. In *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on*, 2004.

[137] D. Broyles, A. Jabbar, and J. P. G. Sterbenz. Design and analysis of a 3-d gauss-markov mobility model for highly-dynamic airborne networks. In *Proceedings of the International Telemetering Conference (ITC),San Diego, CA*, 2010.

[138] J. Haerri, F. Filali, and C. Bonnet. Performance comparison of AODV and OLSR in vanets urban environments under realistic mobility patterns. In *IFIP Med-Hoc-Net workshop*, 2006.

[139] M. Fiore, J. Harri, F. Filali, and C. Bonnet. Vehicular mobility simulation for vanets. In *Simulation Symposium, 2007. ANSS'07. 40th Annual*. IEEE, 2007.

[140] P. Ranjan and K.K. Ahirwar. Comparative study of vanet and manet routing protocols. In *Proceedings of the International Conference on Advanced Computing and Communication Technologies (ACCT 2011)*, 2011.

[141] K.S. Cook. *Trust in society*, volume 2. Russell Sage Foundation Publications, 2001.

[142] D. Gambetta. Can we trust trust. *Trust: Making and breaking cooperative relations*, 2000, 2000.

[143] V. Balakrishnan, V. Varadharajan, and U. Tupakula. Trust management in mobile ad hoc networks. In *Guide to Wireless Ad Hoc Networks*. Springer London, 2009.

[144] Y. Wang and J. Vassileva. Trust and reputation model in peer-to-peer networks. In *Peer-to-Peer Computing, 2003.(P2P 2003). Proceedings. Third International Conference on*, 2003.

[145] F. Azzedin and M. Maheswaran. Evolving and managing trust in grid computing systems. In *Electrical and Computer Engineering, 2002. IEEE CCECE 2002. Canadian Conference on*, 2002.

[146] E. Chang, F. Hussain, and T. Dillon. *Trust and Reputation for Service-Oriented Environments: Technologies For Building Business Intelligence And Consumer Confidence*. John Wiley & Sons, 2005.

[147] J.H. Cho, A. Swami, and R. Chen. A survey on trust management for mobile ad hoc networks. *Communications Surveys & Tutorials, IEEE*, 13(4), 2011.

[148] X. Li, M.R. Lyu, and J. Liu. A trust model based routing protocol for secure ad hoc networks. In *IEEE Aerospace Conference*, 2004.

[149] M.N.K.Babu, A.A.Franklin, and C.S.R.Murthy. On the prevention of collusion attack in olsr-based mobile ad hoc networks. In *Networks, 2008. ICON 2008. 16th IEEE International Conference on*, 2008.

[150] J. Mundinger and J.Y. Le Boudec. Analysis of a reputation system for Mobile Ad-Hoc Networks with liars. *Performance Evaluation*, 65(3-4), 2008.

[151] K.Ren, T.Li, Z.Wan, and al. Highly reliable trust establishment scheme in ad hoc networks. *Computer Networks*, 45(6), 2004.

[152] A.A. Pirzada, C. McDonald, and A. Datta. Performance comparison of trust-based reactive routing protocols. *Mobile Computing, IEEE Transactions on*, 5(6), 2006.

[153] H. Luo, J. Kong, P. Zerfos, and al. Ursa: ubiquitous and robust access control for mobile ad hoc networks. *IEEE/ACM Transactions on Networking (ToN)*, 12(6), 2004.

[154] G.C. Hadjichristofi, W.J. Adams, and N.J. Davis IV. A framework for key management in mobile ad hoc networks. In *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, 2005.

[155] N. Deb and N. Chaki. Tids: Trust-based intrusion detection system for wireless ad-hoc networks. In *Computer Information Systems and Industrial Management*, volume 7564. Springer Berlin Heidelberg, 2012.

[156] M.A. Azer, S.M. El-Kassas, A.W.F. Hassan, and M.S. El-Soudani. A survey on trust and reputation schemes in ad hoc networks. In *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, 2008.

[157] T. M. Cover and J. A. Thomas. *Elements of information theory*. John Wiley and Sons, 2006.

[158] YL.Sun, S.Member, Z.Han, and al. Information theoretic framework of trust modeling and evaluation for ad hoc networks. *IEEE Journal on Selected Area in Communications*, 24, 2006.

[159] I. Ray and N. Poolsapassit. Using attack trees to identify malicious attacks from authorized insiders. In *Computer Security-ESORICS 2005*, volume 3679. Springer Berlin / Heidelberg, 2005.

[160] M.O. Asadoorian and D. Kantarelis. *Essentials of Inferential Statistics*. University Press of Amer, 2005.

[161] R.G. Newcombe. *Confidence Intervals for Proportions and Related Measures of Effect Size*. CRC Press, 2012.

[162] M. Smithson. *Confidence Intervales*. Sage University Papers Series on Quantitative Applications in Social Sciences, 2003.

[163] P. Erdös and A. Rényi. On a new law of large numbers. *Journal d'Analyse Mathématique*, 23, 1970.

[164] M. Rosenblatt. A central limit theorem and a strong mixing condition. *Proceedings of the National Academy of Sciences of the United States of America*, 42(1), 1956.

[165] International Telecommunication Unit. IUT Internet Reports 2005: The Internet of Things, 2005.

[166] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer Networks*, 54(15), 2010.

[167] R. Roman, P. Najera, and J. Lopez. Securing the internet of things. *Computer*, 44(9), 2011.

[168] Y. Ahn, Y. Zhao, U. He, and J. Choi. Efficient ad hoc routing technique for connecting geographically distributed heterogeneous networks. In *Internet of Things*. Springer Berlin Heidelberg, 2012.

[169] I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE'08*. Ieee, 2008.

[170] R. H. Syed, M. Syrame, and J. Bourgeois. Protecting grids from cross-domain attacks using security alert sharing mechanism. *Future Generation Computer System*, 29(2), 2013.

Appendices

APPENDIX A

OLSR FUNCTIONS

Olsr performs the following core functionalities:

1. Link sensing and neighborhood discovery
2. Multipoint Relay selection
3. Declaration of the MPR and of the topology-related information
4. Packet formating and forwarding
5. Route calculation

The first three functionalities have been explained in Chapter 3. We hereafter detail the other functions, namelly the packet formating and forwarding and the route calculation.

Packet Format and Forwarding. The unique format applied for all Olsr packet is provided in Figure A.1. Each packet may encapsulate several messages of potentially different types (e.g., *hello*, Tc). A packet is further embedded into a Udp datagram. Upon reception of a packet, the included messages are treated individually. In practice, the receiver first checks whether the message is generated by itself or has been expired (i.e., the packet Time To Live is set to 0). If not, the node verifies whether the message has already been received and processed. For this purpose, the node maintains a *duplicate tuple* composed of the address of the message's originator, the Message Sequence Number (Msn), a boolean indicating whether this message is already retransmitted, the interface(s) on which this message has been received, and the tuple expiration time. If the message is received for the first time, then no corresponding *Duplicate Tuple* exists and the message is processed. The message type precises also whether the received message should be considered for forwarding or not. Recall that the Tc, Hna, and Mid messages are considered to be forwarded, while the *hello* message is not. It is worth to mention that the Olsr's packets are transmitted only between 1-hop neighbor nodes while the olsr's message may be flooded within a specific diameter, in terms of number of hops, or to the entire network.

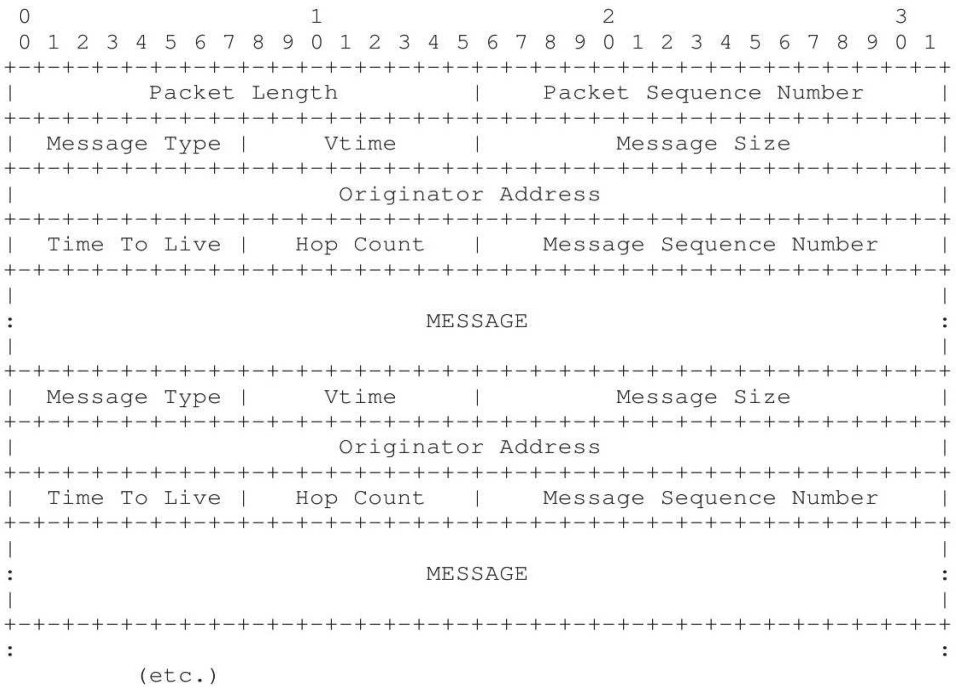


Figure A.1: OLSR packet format [9]

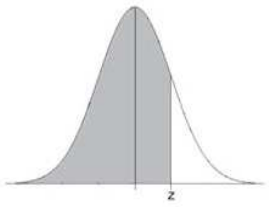
Calculating Routing Table. In OLSR, each node maintains a routing table. An entry in the routing table consists of the the destination address, the next hop address, and the estimated distance to the destination. When a node wants to forward packet(s) toward a destination, it selects the relevant route (i.e., the routing entry), if exist, with the minimal distance. The routing table is re-calculated when a change in the symmetric neighbors takes place, or a used topology entry is expired. Note that re-calculating the routing table does not trigger any type of transmission. A route is seen as a sequence of hops through the MPRs from the source to the destination. Routes are calculated based on the topology information diffused and included in the TC messages. More precisely, each node uses the received TC messages to build and maintain a topology table which records topological information, i.e., the MPRs of other nodes. In particular, an entry of the topology table is a pair in the form [last hop, destination] wherein the last hop corresponds to the originator of a received TC message, while the destination is the node advertised as a neighbor in this message. The routes are built by tracking the connected pairs in a descending order. Indeed, when a node S wants to find the path towards another node D, it first searches for a pair $[X,D]$. If this pair exists, it increases the distance by 1 and searches for another pair $[Y,X]$, and so forth until it finds that Y is one of its 1-hop neighbors. Thus, the required route is calculated and Y is considered as the next hop towards the destination D. Note that if there are two (or more) 1-hop neighbors that are can be selected as a next hop towards a destination, then the one with the highest willingness and the highest amount of MPR selectors is preferred.

APPENDIX B

DISTRIBUTION TABLES

z-critical value. Figure B.1 provides the z critical value. This value is used to estimate the confidence interval when the number of gathered evidences ≥ 30. Recall that this value is related to the used confidence level and represents the portion outside the confidence interval in the standard normal distribution. To understand how the z critical value is taken from this table let have the following example. If the used confidence level is 95% then the upper tail outside the confidence interval constitutes $(100 - 95)/2 = 0.025$ of the area of the standard normal distribution. While the left area constitutes 0.975. In Figure B.1, we find the latter value (i.e., 0.975) at the intersection of the line 1.9 with the column 0.06. Thus, the z criteria value that corresponds to the confidence level 95% is 1.96.

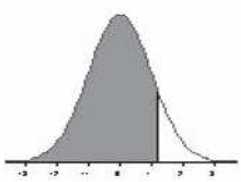
t-critical value. Figure B.2 provides the t critical value. This value is used to estimate the confidence interval when the number of gathered evidences < 30. Recall that this value is related to the used confidence level and represents the portion outside the confidence interval in the student t-distribution. We select the t critical value at the intersection of the column of the required confidence level and the line of the used degree of freedom (df). During the estimation of the confidence interval, the used df is the number of gathered evidences - 1.



z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7852
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389
1.0	0.8413	0.8438	0.8461	0.8485	0.8508	0.8531	0.8554	0.8577	0.8599	0.8621
1.1	0.8643	0.8665	0.8686	0.8708	0.8729	0.8749	0.8770	0.8790	0.8810	0.8830
1.2	0.8849	0.8869	0.8888	0.8907	0.8925	0.8944	0.8962	0.8980	0.8997	0.9015
1.3	0.9032	0.9049	0.9066	0.9082	0.9099	0.9115	0.9131	0.9147	0.9162	0.9177
1.4	0.9192	0.9207	0.9222	0.9236	0.9251	0.9265	0.9279	0.9292	0.9306	0.9319
1.5	0.9332	0.9345	0.9357	0.9370	0.9382	0.9394	0.9406	0.9418	0.9429	0.9441
1.6	0.9452	0.9463	0.9474	0.9484	0.9495	0.9505	0.9515	0.9525	0.9535	0.9545
1.7	0.9554	0.9564	0.9573	0.9582	0.9591	0.9599	0.9608	0.9616	0.9625	0.9633
1.8	0.9641	0.9649	0.9656	0.9664	0.9671	0.9678	0.9686	0.9693	0.9699	0.9706
1.9	0.9713	0.9719	0.9726	0.9732	0.9738	0.9744	0.9750	0.9756	0.9761	0.9767
2.0	0.9772	0.9778	0.9783	0.9788	0.9793	0.9798	0.9803	0.9808	0.9812	0.9817
2.1	0.9821	0.9826	0.9830	0.9834	0.9838	0.9842	0.9846	0.9850	0.9854	0.9857
2.2	0.9861	0.9864	0.9868	0.9871	0.9875	0.9878	0.9881	0.9884	0.9887	0.9890
2.3	0.9893	0.9896	0.9898	0.9901	0.9904	0.9906	0.9909	0.9911	0.9913	0.9916
2.4	0.9918	0.9920	0.9922	0.9925	0.9927	0.9929	0.9931	0.9932	0.9934	0.9936
2.5	0.9938	0.9940	0.9941	0.9943	0.9945	0.9946	0.9948	0.9949	0.9951	0.9952
2.6	0.9953	0.9955	0.9956	0.9957	0.9959	0.9960	0.9961	0.9962	0.9963	0.9964
2.7	0.9965	0.9966	0.9967	0.9968	0.9969	0.9970	0.9971	0.9972	0.9973	0.9974
2.8	0.9974	0.9975	0.9976	0.9977	0.9977	0.9978	0.9979	0.9979	0.9980	0.9981
2.9	0.9981	0.9982	0.9982	0.9983	0.9984	0.9984	0.9985	0.9985	0.9986	0.9986
3.0	0.9987	0.9987	0.9987	0.9988	0.9988	0.9989	0.9989	0.9989	0.9990	0.9990
3.1	0.9990	0.9991	0.9991	0.9991	0.9992	0.9992	0.9992	0.9992	0.9993	0.9993
3.2	0.9993	0.9993	0.9994	0.9994	0.9994	0.9994	0.9994	0.9995	0.9995	0.9995
3.3	0.9995	0.9995	0.9995	0.9996	0.9996	0.9996	0.9996	0.9996	0.9996	0.9997
3.4	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9998

Figure B.1: Standard normal distribution table [a].

[a] *www.math.bgu.ac.il/ ∼ ngur/Teaching/probability/normal.pdf*



	p										
df	0.75	0.80	0.85	0.90	0.95	0.975	0.980	0.990	0.995	0.9975	0.9990
1	1.0000	1.3764	1.9626	3.0777	6.3137	12.706	15.895	31.821	63.656	127.32	318.29
2	0.8165	1.0607	1.3862	1.8856	2.9200	4.3027	4.8487	6.9645	9.9250	14.089	22.329
3	0.7649	0.9785	1.2498	1.6377	2.3534	3.1824	3.4819	4.5407	5.8408	7.4532	10.214
4	0.7407	0.9410	1.1896	1.5332	2.1318	2.7765	2.9985	3.7469	4.6041	5.5975	7.1729
5	0.7267	0.9195	1.1558	1.4759	2.0150	2.5706	2.7565	3.3649	4.0321	4.7733	5.8935
6	0.7176	0.9057	1.1342	1.4398	1.9432	2.4469	2.6122	3.1427	3.7074	4.3168	5.2075
7	0.7111	0.8960	1.1192	1.4149	1.8946	2.3646	2.5168	2.9979	3.4995	4.0294	4.7853
8	0.7064	0.8889	1.1081	1.3968	1.8595	2.3060	2.4490	2.8965	3.3554	3.8325	4.5008
9	0.7027	0.8834	1.0997	1.3830	1.8331	2.2622	2.3984	2.8214	3.2498	3.6896	4.2969
10	0.6998	0.8791	1.0931	1.3722	1.8125	2.2281	2.3593	2.7638	3.1693	3.5814	4.1437
11	0.6974	0.8755	1.0877	1.3634	1.7959	2.2010	2.3281	2.7181	3.1058	3.4966	4.0248
12	0.6955	0.8726	1.0832	1.3562	1.7823	2.1788	2.3027	2.6810	3.0545	3.4284	3.9296
13	0.6938	0.8702	1.0795	1.3502	1.7709	2.1604	2.2816	2.6503	3.0123	3.3725	3.8520
14	0.6924	0.8681	1.0763	1.3450	1.7613	2.1448	2.2638	2.6245	2.9768	3.3257	3.7874
15	0.6912	0.8662	1.0735	1.3406	1.7531	2.1315	2.2485	2.6025	2.9467	3.2860	3.7329
16	0.6901	0.8647	1.0711	1.3368	1.7459	2.1199	2.2354	2.5835	2.9208	3.2520	3.6861
17	0.6892	0.8633	1.0690	1.3334	1.7396	2.1098	2.2238	2.5669	2.8982	3.2224	3.6458
18	0.6884	0.8620	1.0672	1.3304	1.7341	2.1009	2.2137	2.5524	2.8784	3.1966	3.6105
19	0.6876	0.8610	1.0655	1.3277	1.7291	2.0930	2.2047	2.5395	2.8609	3.1737	3.5793
20	0.6870	0.8600	1.0640	1.3253	1.7247	2.0860	2.1967	2.5280	2.8453	3.1534	3.5518
21	0.6864	0.8591	1.0627	1.3232	1.7207	2.0796	2.1894	2.5176	2.8314	3.1352	3.5271
22	0.6858	0.8583	1.0614	1.3212	1.7171	2.0739	2.1829	2.5083	2.8188	3.1188	3.5050
23	0.6853	0.8575	1.0603	1.3195	1.7139	2.0687	2.1770	2.4999	2.8073	3.1040	3.4850
24	0.6848	0.8569	1.0593	1.3178	1.7109	2.0639	2.1715	2.4922	2.7970	3.0905	3.4668
25	0.6844	0.8562	1.0584	1.3163	1.7081	2.0595	2.1666	2.4851	2.7874	3.0782	3.4502
26	0.6840	0.8557	1.0575	1.3150	1.7056	2.0555	2.1620	2.4786	2.7787	3.0669	3.4350
27	0.6837	0.8551	1.0567	1.3137	1.7033	2.0518	2.1578	2.4727	2.7707	3.0565	3.4210
28	0.6834	0.8546	1.0560	1.3125	1.7011	2.0484	2.1539	2.4671	2.7633	3.0470	3.4082
29	0.6830	0.8542	1.0553	1.3114	1.6991	2.0452	2.1503	2.4620	2.7564	3.0380	3.3963
30	0.6828	0.8538	1.0547	1.3104	1.6973	2.0423	2.1470	2.4573	2.7500	3.0298	3.3852
31	0.6825	0.8534	1.0541	1.3095	1.6955	2.0395	2.1438	2.4528	2.7440	3.0221	3.3749
32	0.6822	0.8530	1.0535	1.3086	1.6939	2.0369	2.1409	2.4487	2.7385	3.0149	3.3653
33	0.6820	0.8526	1.0530	1.3077	1.6924	2.0345	2.1382	2.4448	2.7333	3.0082	3.3563
34	0.6818	0.8523	1.0525	1.3070	1.6909	2.0322	2.1356	2.4411	2.7284	3.0020	3.3480
35	0.6816	0.8520	1.0520	1.3062	1.6896	2.0301	2.1332	2.4377	2.7238	2.9961	3.3400
36	0.6814	0.8517	1.0516	1.3055	1.6883	2.0281	2.1309	2.4345	2.7195	2.9905	3.3326
37	0.6812	0.8514	1.0512	1.3049	1.6871	2.0262	2.1287	2.4314	2.7154	2.9853	3.3256
38	0.6810	0.8512	1.0508	1.3042	1.6860	2.0244	2.1267	2.4286	2.7116	2.9803	3.3190
39	0.6808	0.8509	1.0504	1.3036	1.6849	2.0227	2.1247	2.4258	2.7079	2.9756	3.3127
40	0.6807	0.8507	1.0500	1.3031	1.6839	2.0211	2.1229	2.4233	2.7045	2.9712	3.3069
50	0.6794	0.8489	1.0473	1.2987	1.6759	2.0086	2.1087	2.4033	2.6778	2.9370	3.2614
60	0.6786	0.8477	1.0455	1.2958	1.6706	2.0003	2.0994	2.3901	2.6603	2.9146	3.2317
75	0.6778	0.8464	1.0436	1.2929	1.6654	1.9921	2.0901	2.3771	2.6430	2.8924	3.2024
100	0.6770	0.8452	1.0418	1.2901	1.6602	1.9840	2.0809	2.3642	2.6259	2.8707	3.1738
∞	0.6745	0.8416	1.0364	1.2816	1.6449	1.9600	2.0537	2.3263	2.5758	2.8070	3.0902

Figure B.2: t (or Strudent) distribution table[a].

[a] *www.rosaweb.org/descargas/tstudent.pdf*

APPENDIX C

LIDR IMPLEMENTATION

LIDR is implemented in *Perl*[63], which facilitates the text processing and hence the processing of the OLSR logs. Although this version of our IDS considers the link spoofing attack, other attacks may be considered by specifying the evidences to be searched in the logs and the information to be exchanged during the diagnostic. In practice, our code is organized into three basic units: the logs parsing (Section C.1), the communication manager (Section C.2), and the diagnostic (Section C.3).

C.1 Logs Parsing

Listing C.1 contains the core function associated with the reading and parsing of the OLSR logs: information related to the 1-hop neighbors, the 2-hop neighbors, and the MPRs is extracted and further maintained in a database. This database is realized in *Mysql* [64]. It is worth to mention that the format of the OLSR logs depends of the version/implementation of the protocol.

Listing C.1: Parsing OLSR logs

```
sub handle_logs{
# Getting the node's IP address and the parameters of DB connection.
  my($nodeIP,$nodeNumber)=@_;
  my $dbh1=&connectDB_threads($nodeNumber);

# Reading the new lines in the log file
  $file=File::Tail->new(name=>"/var/log/olsr/OLSR$nodeNumber.log", maxinterval=>0.1 , interval=>0.1,
       adjustafter=>1) or die "I could not open OLSR logs" ;
  while (defined($line=$file->read))
  {
# Extracting information about the 1-hop neighbors
    if($line =~ /(\d{0,}\.\d{0,})s\s(\d{0,})(.*)\sNeighborTuple\(neighborMainAddr=
    (\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})\,\sstatus=([a-zA-Z]+)(.*)/)
    {
      $current_time= $1 ;
      $n1= $4;
      $sth= $dbh1->prepare("insert into OLSR_1hop_Neighbors(node_ip, n1_ip, time) values (?,?,?)");
      $sth->execute($nodeIP,$n1,$current_time);
     }
# Extracting information about the 2-hop neighbors
   elsif($line =~ /(\d{0,}\.\d{0,})s\s(d{0,})(.*)\sTwoHopNeighborTuple\(neighborMainAddr=(\d{1,3}\.\d
       {1,3}\.\d{1,3}\.\d{1,3})\,\stwoHopNeighborAddr=(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})\,\
       sexpirationTime=(\d{0,}\.{0,1}\d{0,})(.*)/)
```

[63] http://www.perl.org

[64] http://www.mysql.com

```
    {
      $current_time= $1;
      $n1= $4;
      $n2=$5;
      $exp_time= $6;
      if ($exp_time > $current_time)#&& !($nodeIP eq $n2))
         {
           $sth= $dbh1->prepare("insert into OLSR_2hop_Neighbors(node_ip, n1_ip, n2_ip, time) values (?,?,?,?)");
           $sth->execute($nodeIP,$n1,$n2,$current_time);
         }
    }
# Getting the node's point of view about the network topology
    elsif ($line =~ /(\d{0,}\.\d{0,})s\s(d{0,})(.*)BEGIN dump TopologySet for OLSR Node(.*)/)
    {
        $current_time= $1;
        if($x_time_topology != $current_time)
        {
                $ok_topology= 1;
        }
        else {
                $ok_topology= 0;
        }
    }
    elsif ($line =~ /(\d{0,}\.\d{0,})s\s(d{0,})(.*)\sTopologyTuple\(destAddr=(\d{1,3}\.\d{1,3}\.\d
        {1,3}\.\d{1,3})\,\slastAddr=(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})(.*)/)
    {
      $current_time= $1;
      $des_ip= $3;
      $last_ip=$4;

      if($ok_topology == 1)
      {
        $x_time_topology = $current_time;
        $sth= $dbh1->prepare("insert into Topology_entry(node_ip, dest_ip, last_ip, time) values (?,?,?,?)");
        $sth->execute($nodeIP,$des_ip,$last_ip,$current_time);
      }

    }
# Extracting information about the MPRs
   elsif ($line =~ /(\d{0,}\.\d{0,})s\s(d{0,})(.*)\sComputed MPR set for node\s(\d{1,3}\.\d{1,3}\.\d
       {1,3}\.\d{1,3}):\s\[(.*)\]/)
    {
     $current_time= $1 ;
     $MPR= $5;
     if ($MPR ne '')
      {
          while ($MPR =~ /(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})/g)
          {
             $sth = $dbh1->prepare("insert into OLSR_MPR_Info (node_ip, mpr_ip, time)values(?,?,?)");
             $sth->execute($nodeIP,$1,$current_time)|| die $DBI::errstr;
          }
# Extracting the suspiciou MPR(s)
          &Get_supicious($nodeIP,$current_time,$nodeNumber,$dbh1);
      }

    }
  }
 $sth->finish();
 close(FILE);
}
```

The *Get_suspicious* function (Listing C.2) is called at the end of the parsing function. It analyses the neighborhood and the MPR-related information in order to identify the suspicious MPRs. In practice, this function searches for a node that is recently selected as a MPR and whose neighbor set contains the neighbor set of an previous-MPR. Here, the new MPR could be an intruder that falsifies its 1-hop neighbors set in order to be selected as a MPR and further exclude a legitimate MPR. Such case represents an evidence with a sufficient level of gravity that triggers an in-depth diagnostic as we will see in Section C.3.

Listing C.2: Identifying the suspicious MPRs

```
sub Get_suspicious{
    my($node_ip, $curr_time,$nodeNumber,$dbh_thread)= @_;
    my $sth=$dbh_thread->prepare("select time from OLSR_MPR_Info where node_ip=".$dbh_thread->quote($node_ip)
        ." and time <".$curr_time."order by time desc limit 1");
    $sth->execute();
    my @ex_time= $sth->fetchrow();
```

```
    if(defined($ex_time[0])){
       my $sql= "select Distinct n2_ip from OLSR_2hop_Neighbors where node_ip=".$dbh_thread->quote($node_ip)."
            and time=".$curr_time." and n1_ip in (select mpr_ip from OLSR_MPR_Info where node_ip=".$dbh_thread->
            quote($node_ip)." and time=".$ex_time[0].") and n2_ip in (select n2_ip from OLSR_2hop_Neighbors where
             node_ip=".$dbh_thread->quote($node_ip)." and time=".$ex_time[0].") and n1_ip not in (select mpr_ip
             from OLSR_MPR_Info where node_ip=".$dbh_thread->quote($node_ip)."and time= ".$curr_time.")";
       $sth= $dbh_thread->prepare($sql);
       $sth->execute();
       my $unsure_2hop="";
       while (my @row = $sth->fetchrow_array){
             if($row[0] ne $node_ip){
               $unsure_2hop="'".$row[0]."',".$unsure_2hop;
             }
       }
# Finding the MPRs that cause the execlusion of ex-MPRs and tag them as suspicious
      if($unsure_2hop ne ""){
         $unsure_2hop = substr($unsure_2hop,0,length($unsure_2hop) -1);
         $sql= "select Distinct n1_ip,n2_ip from OLSR_2hop_Neighbors where node_ip=".$dbh_thread->quote($node_ip
              )." and time= ".$curr_time." and n2_ip in (".$unsure_2hop.")and n1_ip in (select mpr_ip from
              OLSR_MPR_Info where node_ip=".$dbh_thread->quote($node_ip)." and time=".$curr_time.")";
         $sth= $dbh_thread->prepare($sql);
         $sth->execute();
         $verified_2hop=""; $x_n1="";
         while(@row = $sth->fetchrow()){
              $n1=$row[0];
              $n2=$row[1];
              $sql="select node_ip from OLSR_2hop_Neighbors where node_ip=".$dbh_thread->quote($node_ip)." and
                   time=".$ex_time[0]." and n1_ip=".$dbh_thread->quote($n1)." and n2_ip=".$dbh_thread->quote
                   ($n2);
              $sth1=$dbh_thread->prepare($sql);
              $sth1->execute();
              if($sth1->rows == 0){
                if($x_n1 eq ""){
                  $verified_2hop= $n2;
                }
                elsif($x_n1 ne $n1){
                     eval{
# Identifying the 1-hop relationships to be verified
                          $sth1=$dbh_thread->prepare ("insert into MIM_alarm (node_ip,unsure_mpr_ip,
                               h2_checked_ip,time,result) values(?,?,?,?,?)");
                          $sth1->execute($node_ip,$n1,$verified_2hop,$curr_time,0)|| die $DBI::errstr;
                        }or do {print "error insertion suspicious $@ ..\n";};
                        $verified_2hop=$n2;
                }else
                { $verified_2hop=$verified_2hop.",".$n2;}
                $x_n1 = $n1;
              }
         }
         if($x_n1 ne ""){
           eval{
             $sth1=$dbh_thread->prepare("insert into MIM_alarm (node_ip,unsure_mpr_ip,h2_checked_ip,time,result)
                  values(?,?,?,?,?)");
             $sth1->execute($node_ip,$n1,$verified_2hop,$curr_time,'0')|| die $DBI::errstr;
           } or do {print "error insertion suspicious2 $@...\n"; };
         }
      }
   }
}
```

C.2 Communication Manager

Listing C.3 performs the cooperations with others. The cooperation attempts to verify the neighborhood relationships. In practice, the node may receive interrogations about its relation with a specific node at a given time. In such case, the node checks its database in order to either confirm or deny being a 1-hop neighbor of the node under investigation. Note that TCP-based bidirectional communication is realized between the requester and responder nodes during the cooperation. In addition, the communication manager runs in a separate thread so that other functions are not blocked.

Listing C.3: Handling cooperation requests

```
sub handle_cooperation{
```

```
   my($nodeIP,$port)=@_;
   my $socket_Child = new IO::Socket::INET(
                      LocalHost => $nodeIP,
                      LocalPort => $port,
                      Proto => "tcp",
                      Listen => 5,
                      Reuse => 1
                     )or die "ERROR in Chield Socket Creation $nodeIP: $!\n";
  while(1){
          $client_socket = "";
# Acceptin a connection request from another node
          $client_socket = $socket_Child->accept();
          $result="";
          $peer_address = $client_socket->peerhost();
          $peer_port = $client_socket->peerport();
          $client_socket->recv($recieved_data,2024);
          @msg= split(/ /, $recieved_data);
# Handling a request message
          if($msg[0] eq "R"){
               $unsur_mpr=$msg[1];
               $curr_time=$msg[2];
               @list_verification= split(/,/, $msg[3]);
               $dbh_child= &connectDB_threads($nodeNumber);
               my $sth_child=$dbh_child->prepare("select distinct time from OLSR_2hop_Neighbors where node_ip="
                    .$dbh_child->quote($nodeIP)."order by time desc limit 1");
               $sth_child->execute();
               my @ex_time= $sth_child->fetchrow();
               if(defined($ex_time[0])){
                  $counter=0;
                  while(($ex_time[0] < $curr_time)&&($counter < 5 )){
                        sleep(10); $counter++;
                        $sth_child=$dbh_child->prepare("select distinct time from OLSR_2hop_Neighbors where
                             node_ip=".$dbh_child->quote($nodeIP)."order by time desc limit 1");
                        $sth_child->execute();
                        @ex_time= $sth_child->fetchrow();
                  }

               }
               $x_time = $curr_time - 2;
               if ($ex_time[0] <  $x_time){
                   print"logs are slowly traited\n";
                   $result="0,";
               }else
                   {
#check if there is a link between this node and the unsur MPR
                     for($i=0; $i < scalar(@list_verification); $i++){
                        if($list_verification[$i] eq $nodeIP){
                          $sql_child="select count(n1_ip) from OLSR_2hop_Neighbors where node_ip=".$dbh_child->
                               quote($nodeIP)." and n1_ip=".$dbh_child->quote($unsur_mpr)." and (time
                               between ".$x_time." and ".$curr_time.")";
                          @get_count = $dbh_child->selectrow_array($sql_child);
                          if ($get_count[0] == 0){
                              $result= $result."-1,";
                          }
                          else{
                                $result= $result."1,";
                          }
                        }else
                             {
                              $sql_child="select count(n1_ip) from OLSR_2hop_Neighbors where node_ip=".
                                   $dbh_child->quote($nodeIP)." and n1_ip=".$dbh_child->quote(
                                   $list_verification[$i])." and (time between ".$x_time." and ".$curr_time
                                   .")";
                              @get_count = $dbh_child->selectrow_array($sql_child);
                              if ($get_count[0] != 0){
                                  $sql_child="select count(n1_ip) from OLSR_2hop_Neighbors where node_ip=".
                                       $dbh_child->quote($nodeIP)." and n1_ip=".$dbh_child->quote(
                                       $list_verification[$i])." and n2_ip=".$dbh_child->quote($unsur_mpr
                                       )." and (time between ".$x_time." and ".$curr_time.")";
                                  @get_count2 = $dbh_child->selectrow_array($sql_child);
                                  if ($get_count2[0] == 0){
                                      $result= $result."-1,";
                                  }else
                                      {$result= $result."1,";}

                              }else
                                   {
                                    $x_time = $curr_time - 4.5;
                                    $sql_child="select count(dest_ip) from Topology_entry where node_ip=".
                                         $dbh_child->quote($nodeIP)." and dest_ip=".$dbh_child->quote(
                                         $list_verification[$i])." and (time between ".$x_time." and ".
                                         $curr_time.")";
                                    @get_count = $dbh_child->selectrow_array($sql_child);
                                    if ($get_count[0] != 0){
```

```
                                        $sql_child="select count(dest_ip) from Topology_entry where node_ip=".
                                            $dbh_child->quote($nodeIP)." and dest_ip=".$dbh_child->quote
                                            ($list_verification[$i])." and last_ip=".$dbh_child->quote(
                                            $unsur_mpr)."  and (time between ".$x_time." and ".$curr_time.
                                            ")";
                                        @get_count2 = $dbh_child->selectrow_array($sql_child);
                                        if ($get_count2[0] == 0){
                                            $result= $result."-1,";
                                        }else
                                            {$result= $result."1,";}
                                    }else
                                        { $result= $result."0,"; }
                                    }
                                }
                            }
                        }
                        $client_socket->send($result);
                    }
# Forwarding an answer towards another node
                elsif ($msg[0] eq "N"){
                        $unsur_mpr=$msg[1];
                        $target_node=$msg[2];
                        $curr_time=$msg[3];
                        eval{
                            $socket2 = new IO::Socket::INET (
                            PeerAddr  => $target_node,
                            PeerPort  =>  $port,
                            Proto => "tcp",
                            Reuse => 1
                            ) or print "ERROR in N-R Socket Creation $target_node: $!\n";
                            if(defined($socket2)){
                               $msg2 = "R $unsur_mpr $curr_time";
                               $socket2->send($msg2);
                               $socket2->recv($recv_data,1024);
                               close $socket2;
                            }
                        };
                        if($recv_data eq ""){
                           $recv_data="unreachable target";
                        }
                        $client_socket->send($recv_data);
                }else
                    {print "Unknown command @msg \n";}
                close $client_socket;
    }
}
```

C.3 Diagnostic Unit

The intrusion diagnostic is the heart of our IDS. It contains the functions that match the evidences with the intrusion signatures, which potentially involves cooperation. It is also in relation with the trust handler and the the diagnostic confidence interval establisher.

Link Spoofing Detection Listing C.4 represents the function that verifies the relationships of a suspicious MPR. In practice, it handles every declared 1-hop neighbor relation of the suspicious MPR in a separate thread that further interrogates other nodes in order to confirm or deny this relation. Every returned answer is weighted according to the trustworthiness of its provider. Then, all the weighted answers are combined to obtain the result of the diagnostic. In addition, the function *calc_CI()*, which calculates the confidence interval is called. Finally, the trustworthiness of the suspicious MPR and of the interrogated nodes are updated according to the obtained result and the corresponding confidence interval.

Listing C.4: In-depth diagnostic

```
sub MIM_Detection{
    $port="6001";
    my($node_ip,$unsur_mpr,$verified_2hop,$curr_time,$ex_time,$nodeNumber,$dbh_thread)= @_;
```

```
    my @list_participations_results=();
    my @list_answer_node=();
    my @list_verified_nodes=();
    my @list_verified_nodes= split(/,/,$verified_2hop);
    my $msg="R $unsur_mpr $curr_time $verified_2hop";
    my $returned_answer="";
    my $participat="";
# Selecting the interrogated nodes
    $sql="(select n1_ip from OLSR_2hop_Neighbors where node_ip=".$dbh_thread->quote($node_ip)." and n1_ip <>".
        $dbh_thread->quote($unsur_mpr)." and time=".$curr_time.") UNION DISTINCT (select n2_ip from
        OLSR_2hop_Neighbors where node_ip=".$dbh_thread->quote($node_ip)." and time=".$curr_time." and n1_ip="
        .$dbh_thread->quote($unsur_mpr)." and n2_ip<>".$dbh_thread->quote($node_ip).")";
    my $sth=$dbh_thread->prepare($sql);
    $sth->execute();
    while(@row = $sth->fetchrow()){
        $returned_answer="";
        $recv_data="";
        $target_ip=$row[0];
        eval{
            my $socket_thread = new IO::Socket::INET (
            PeerAddr  => $target_ip,
            PeerPort  => $port,
            Proto => "tcp",
            Timeout => 2,
            Reuse => 1
            )or do print "no route to $target_ip \n";
# Sending the cooperation request
        if(defined($socket_thread)){
           $socket_thread->send($msg);
           $socket_thread->recv($recv_data,1024);
           $returned_answer=$recv_data;
           my @res_list= split(/,/, $returned_answer);
# Adding the cooperated node into the list of participants
# This list serves in updating the nodes' trustworthiness
           push(@list_answer_node, &nodeIp_to_nodeNumber($target_ip));
           push(@list_participations_results,[@res_list]);
           close $socket_thread;
        }
       }
    }
#Combining the returned answer about a suspicious relation
#This combination takes into account the trustworthiness of sources
    for($i = 0; $i<scalar(@list_verified_nodes);$i++){
        $result=0;
        $sum_TV=0;
        my $counter_participates=0;
        @positive_list=(); # the nodes that confirm the suspicious link
        @negative_list=(); # the nodes that deny the suspicious link
        @list_sub_results=();
        for($j=0; $j<scalar(@list_answer_node);$j++){
           if($list_participations_results[$j][$i]!=0){
             $counter_participates +=1;
             $TV = $list_tvs[$list_answer_node[$j]-1];
             $sub_res = $list_participations_results[$j][$i] * $TV;
             $result += $sub_res;
             $sum_TV += $TV;
             push(@list_sub_results, $sub_res);

                #check if the verified nodes returned a positive or negative answers
                if($list_participations_results[$j][$i] > 0)
                {
                push(@positive_list, $list_answer_node[$j]);
                }else{
                   push(@negative_list, $list_answer_node[$j]);
                }
           }
        }
        if($sum_TV != 0){
           $result = $result/$sum_TV;
           $result = eval sprintf('%.4f', $result);
# Calculating the confidence interval
           $ci=calc_CI(\@list_sub_results,$sum_TV,$curr_time,$unsur_mpr);
           $nbr=&nodeIp_to_nodeNumber($unsur_mpr);
# Updating the trustworthiness of the suspicious MPR according to the resutl
# and the corresponding confidence interval
           $list_tvs[$nbr-1]= change_suspicion_tv($list_tvs[$nbr-1], $ci , $result, $curr_time,$nbr,
               $counter_participates);
# Updating the trustworthiness of the participated nodes
           if($result > 0.1){
             for($k=0;$k<scalar(@positive_list);$k++){
                 $list_tvs[$positive_list[$k]-1]= &change_trust_value($list_tvs[$positive_list[$k]-1],
                     $sum_TV, 1, $curr_time, $positive_list[$k]-1);
             }
             for($k=0;$k<scalar(@negative_list);$k++){
```

```
                $list_tvs[$negative_list[$k]-1]= &change_trust_value($list_tvs[$negative_list[$k]-1],
                    $sum_TV, -1, $curr_time, $negative_list[$k]-1);
            }
          } elsif($result < -0.1){
                for($k=0;$k<scalar(@positive_list);$k++){
                    $list_tvs[$positive_list[$k]-1]= &change_trust_value($list_tvs[$positive_list[$k
                        ]-1],$sum_TV, -1, $curr_time, $positive_list[$k]-1);
                }
                for($k=0;$k<scalar(@negative_list);$k++){
                    $list_tvs[$negative_list[$k]-1]= &change_trust_value($list_tvs[$negative_list[$k
                        ]-1],$sum_TV, 1, $curr_time, $negative_list[$k]-1);
                }
          }
          &print_Trust_to_File($curr_time,\@list_tvs);
        }
    }
}
```

Calculation of the Confidence Interval Listing C.5 represents the function that calculates the sampling error. This error is used for estimating the confidence interval of the diagnostic result. The T-score (Appendix B.2) is used in this function because the size of the evidences sample is less than 30 in our experiments. However, if the size of the evidences sample exceeds 30 then the T-score should be replaced by the z-score (Appendix B.1).

Listing C.5: Calculating the confidence interval

```
sub calc_CI{
        my($list, $sum_tv, $time, $unsur_mpr) = @_;
# T-score
        @t=(12.706, 4.303, 3.182, 2.776, 2.571, 2.447, 2.365, 2.306, 2.262,2.228, 2.201, 2.179, 2.160,
            2.145, 2.131);
        @list_res=@{$list};
        $n= @list_res;
# Calculating the mean
        for($l =0; $l < $n; $l++){
           $sum += $list_res[$l]/$sum_tv;
        }
        $res=$sum;
        $m= $sum/$n;
# Calculating the standard deviation
        $sum=0;
        for($l =0; $l < $n; $l++){
           $sum += ($m-$list_res[$l])**2;
        }
        if($n > 1){
          $sum =$sum/($n -1);
        }
        $SD= eval sprintf('%.4f',sqrt($sum));
# Calculating the sampling error
        $ci= $t[$n-1] *($SD / sqrt($n));
        $ci= eval sprintf('%.4f',$ci);
        return $ci;
}
```

Discussion on the Diagnostic Result Listing C.6 shows the function that verifies whether the result is sufficiently reliable. This function also decides whether to launch the alarm (or eliminating the suspicion), updates the trustworthiness of the suspicious Mpr and retards the conclusion of the diagnostic if necessary.

Listing C.6: Discussing the diagnostic result

```
sub change_suspicion_tv{
    my($x_tv, $ci, $res,$curr_time,$nbr,$counter_participates)=@_;
    $tv=0.4; $value=0; $div=1;
# Verifying whether the result is reliable or not
     if($res > 0){
       if($res - $ci > 0){
          $value=0.075/$div;
          print "the suspicious 1-hop relation surely exists \n";
       }
```

```
       elsif ($res - $ci =< 0){
         $value=0.05/$div;
         print "Diagnostic should be continued \n";
       }
     }elsif($res <= 0){
            if($res + $ci <= 0){
               $value=-0.2/$div;
               print "$unsur_mpr is surely intruder \n";
               alarm();
            }elsif($res + $ci > 0){
                  $value=-0.1/$div;
                  print "Diagnostic should be continued \n";
             }
      }
      $list_changes[$nbr-1]= $curr_time;
# Updating the trustworthiness of the suspicious MPR
      if(($x_tv + $value) > 1){
         $tv=1;
      } elsif(($x_tv+ $value) < 0) {
              $tv=0;
      }else
         { $tv= $x_tv + $value; }
      return $tv ;
}
```

Résumé :

Les réseaux mobiles ad hoc, communément appelés MANET (Mobile Ad hoc NETwork), sont de plus en plus présents dans notre environnement quotidien. Ils deviennent de fait une pierre angulaire des applications commerciales, militaires ou scientifiques. Cependant, ces réseaux sont amenés à opérer dans des environnements ouverts, ce qui les rend particulièrement vulnérables. Ainsi, les méthodes traditionnelles de sécurisation préventives s'appuyant par exemple sur des pare-feux et du chiffrement, ne sont plus suffisantes et doivent être agrémentées de mécanismes réactifs comme les systèmes de détection d'intrusions. Concevoir un système de détection d'intrusion pour les MANETs est difficile car il est nécessaire d'assurer à la fois une détection précise tout en limitant l'utilisation des ressources (en termes de mémoire, de batteries et la bande passante) et en rendant adaptable à la dynamicité du réseau ad hoc. De plus, le système de détection ne doit pas être la cible d'attaques ou de falsification. Nous avons proposé dans cette thèse un système de détection ciblant des exigences. Dans un premier temps, nous avons étudié les attaques qui menacent les MANETs, en se concentrant sur les attaques visant le protocole de routage OLSR (Optimized Link State Routing). Puis, nous présentons notre système qui offre un taux élevé de détection et se singularise par une utilisation efficace des ressources. Notre système analyse les traces de routage au lieu de surveiller le trafic,et cela, afin d'identifier toute évidence d'activité suspecte. Puis, il fait correspondre les évidences à un ensemble de signatures prédéfinies; une signature est perçue comme étant un ensemble partiellement ordonné d'événements caractérisant une intrusion. En outre, notre système établie le degré de suspicion des évidences de manière à limiter le nombre et la durée de ces opérations coûteuses. Nous proposons de nous baser sur une intervalle de confiance pour mesurer la fiabilité de détection. Cette mesure statistique permet de limiter la collecte et le traitement des évidences et de prendre une décision en tout état de cause. Afin d'améliorer la robustesse de notre système, nous le couplons à un modèle de confiance basé sur l'entropie. Ainsi, des crédits sont attribuées aux nœuds ce qui réduit l'effet néfaste des évidences falsifiées fournies par les nœuds. Le modèle de crédit proposé prend en compte le niveau de risque des attaques. Plus précisément, la perte de crédit dépend des conséquences de l'attaque. Nous avons expérimenté le système proposé en considérant plusieurs scénarios de mobilité et de la densité. Les résultats montrent que notre détecteur offrent un taux de détection élevé en adéquation avec les ressources disponibles.

SPIM

■ École doctorale SPIM 16 route de Gray F - 25030 Besançon cedex

■ tél. +33 (0)3 81 66 66 02 ■ ed-spim@univ-fcomte.fr ■ www.ed-spim.univ-fcomte.fr

UFC
UNIVERSITÉ
DE FRANCHE-COMTÉ