A Novel Cooperative Intrusion Detection System for Mobile Ad Hoc Networks

Adam Solomon
Nova Southeastern University, adam_j_solomon@yahoo.com

A Novel Cooperative Intrusion Detection System for Mobile Ad Hoc Networks

by

Adam Solomon

A dissertation submitted in partial fulfillment of the requirements

for the degree of Doctor of Philosophy

In

Information Systems

College of Engineering and Computing

Nova Southeastern University

We hereby certify that this dissertation, submitted by Adam Solomon, conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

[illegible]

James D. Cannady, Ph.D.
Chairperson of Dissertation Committee

12/11/2018
Date

[illegible]

Glyn T. Gowing, Ph.D.
Dissertation Committee Member

11 DEC 2018
Date

[illegible]

Peixiang Liu, Ph.D.
Dissertation Committee Member

12/11/2018
Date

Approved:

Meline Kevorkian

Meline Kevorkian, Ed.D.
Interim Dean, College of Engineering and Computing

12/11/18
Date

College of Engineering and Computing
Nova Southeastern University

2018

An Abstract of a Dissertation Submitted to Nova Southeastern University
in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

A Novel Cooperative Intrusion Detection System for Mobile Ad Hoc Networks

by
Adam Solomon
2018

Mobile ad hoc networks (MANETs) have experienced rapid growth in their use for various military, medical, and commercial scenarios. This is due to their dynamic nature that enables the deployment of such networks, in any target environment, without the need for a pre-existing infrastructure. On the other hand, the unique characteristics of MANETs, such as the lack of central networking points, limited wireless range, and constrained resources, have made the quest for securing such networks a challenging task. A large number of studies have focused on intrusion detection systems (IDSs) as a solid line of defense against various attacks targeting the vulnerable nature of MANETs. Since cooperation between nodes is mandatory to detect complex attacks in real time, various solutions have been proposed to provide cooperative IDSs (CIDSs) in efforts to improve detection efficiency. However, all of these solutions suffer from high rates of false alarms, and they violate the constrained-bandwidth nature of MANETs. To overcome these two problems, this research presented a novel CIDS utilizing the concept of social communities and the Dempster-Shafer theory (DST) of evidence. The concept of social communities was intended to establish reliable cooperative detection reporting while consuming minimal bandwidth. On the other hand, DST targeted decreasing false accusations through honoring partial/lack of evidence obtained solely from reliable sources. Experimental evaluation of the proposed CIDS resulted in consistently high detection rates, low false alarms rates, and low bandwidth consumption. The results of this research demonstrated the viability of applying the social communities concept combined with DST in achieving high detection accuracy and minimized bandwidth consumption throughout the detection process.

Acknowledgments

First and foremost, I would like to thank my creator for gifting me the passion to pursue further knowledge and education. Throughout my dissertation process, I have had great support from my dissertation adviser and role model: Dr. James Cannady. His invaluable guidance during my research process made me indeed a better researcher. I'm forever grateful to him for instilling in me solid principles of scientific research. I would like to thank Dr. Gowing and Dr. Liu for their valuable feedback and time throughout this process.

I would also like to thank my family for their continuous support and patience during my research studies. I'm very appreciative of your encouragement and love that have pushed me towards persevering till the finish line of my dissertation work.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Background

The rapid proliferation of wireless network devices, alongside the growth of related technologies, has increased the adoption of mobile ad hoc networks (MANETs) across different military and commercial fields. MANETs are multi-hop wireless networks consisting of a number of wireless devices (nodes) that communicate with each other without the need for preexisting configurations (Singh, Woo, & Raghavendra, 1998). This gives such networks the ability to be deployed in various situations where no infrastructure exists, such as disaster relief sites, emergency conferences, and battlefields (Johansson, 1999). MANETs are distinguished from other types of networks, by their possession of certain unique characteristics. These include infrastructure-less routing, limited resources, constrained bandwidth, dynamic topologies, and limited wireless range (Corson, Macker, & Cirincione, 1999).

The dynamic nature of MANETs has been a major challenge for those providing security solutions for these networks. The same characteristics that give these networks survivability in sites where no infrastructure exists, render them susceptible to unique security challenges (Dorri, Kamel, & Kheirkhah, 2015). The current body of knowledge contains an extensive amount of research proposing various security solutions for fixed networks. These solutions typically rely on central traffic points to monitor and collect audit data. As such, these solutions cannot be applied for MANETs due to the infrastructure-less nature of these networks (Anantvalee & Wu, 2007). On the other hand, the resource-constrained nature of MANETs poses another challenge in the way of network availability. Nodes in MANETs utilize a shared wireless medium to relay their messages back and forth. However, such a medium is limited in capacity as it conforms to the inherently bandwidth-constrained nature of MANETs. As more nodes begin to utilize the same channel, the chances of interference and link errors rise in proportion to the increasing number of nodes. These errors, in turn, may result in communication interruptions as well as information loss that can have devastating consequences for mission-critical networks (Yang et al., 2004).

The ever-challenging task of designing security solutions for MANETs has always been hindered by the limited energy and processing power possessed by nodes in these networks. These limitations have pushed a large number of researchers towards the task of establishing balance between energy conservation and minimization of processing overhead (Kim & Jang, 2006; Mikki, 2009; Maleki, Dantu, & Pedram, 2002). This balance is necessary because any packet transmission, such as sending and receiving, or even standby hardware operations consumes a nontrivial amount of power that can deplete a node's battery. Extra processing overhead imposed by security solutions may result in draining nodes' battery power, leaving them incapable of participating in normal network operations. As such, resource conservation is critical when designing security implementations for MANETs to sustain a longer network lifetime (Kim & Jang, 2006).

Various preventive security solutions for MANETs exist in the current body of knowledge. These solutions serve as a first line of defense against malicious attempts to compromise the network. However, history has shown that preventive security solutions cannot survive on their own, and this issue continues to be problematic. As networks evolve and become more complex, security is still an afterthought in many designs while the exploitability of preventative solutions increases along with the network complexity. Therefore, the need for an intrusion detection system (IDS) solution as a second line of defense is regarded as a necessity for maintaining the survivability of MANETs (Zhang, Lee, & Huang, 2003).

A plethora of research studies have proposed IDSs as solid lines of defense that can provide an intrusion-free environment for MANETs. Such systems target continuous monitoring of network traffic in order to detect and respond to hostile activities that might compromise network security. However, a significant number of the proposed IDS solutions in the current literature fall short in fulfilling their security goals. This can mostly be blamed on the dynamic nature of MANETs, which has made the quest of intrusion detection an exceptionally complicated task (Cannady, 2010).

The lack of central traffic points in MANETs introduces a mandatory need for real-time cooperative detection techniques to achieve an effective IDS (Mahmood, Amin, Amir, & Khan, 2009). Besides, a broader view of the network is deemed significant to defend against insider attacks, which can only be accomplished through nodes' cooperation in the detection process (Morais & Cavalli, 2012). The existing body of research contains various solutions for cooperative IDS (CIDS) in MANETs. However, all of the solutions suffer from two major problems: high communication overhead caused by constant information exchange and reliance on intrusion reports originating from anonymous nodes. These, in turn, result in high false alarms rates that cause degradation in the detection accuracy as well as increased bandwidth consumption that might result in disrupting the normal routing operations in such bandwidth-constrained networks.

To this effect, researchers emphasized the continuous increase of security threats, and the overwhelming need for an efficient and reliable IDS for MANETs (Banerjee, Nandi, Dey, & Saha, 2015; Keyshap, 2015). By proposing a novel IDS for MANETs, this research attempted to address the high bandwidth consumptions and false alarms issues found in the current solutions. The proposed system is cooperative in nature, protocol-independent, and aimed to decrease bandwidth consumption as well as improving detection accuracy through reducing false positives.

Problem Statement

Currently, there is no efficient CIDS for MANETs. All of the proposed solutions suffer from high communication overhead and false alarm rates. The reliance of such solutions on constant information dissemination throughout the network violates the bandwidth-constrained nature of MANETs. This, in turn, causes disruption of normal network operations as well as loss of detection-related messages, which affects detection accuracy. On the other hand, relying on reports from anonymous nodes in intrusion decision-making renders the current solutions susceptible to increased rates of false alarms and incapable of efficient detection of attacks when large numbers of malicious nodes exist in the network.

Dissertation Goal

The goal of this research was to develop an effective CIDS for MANETs that is capable of detecting malicious activities with high accuracy and achieving low bandwidth consumption to preserve the normal routing operations in the network. We presented a novel approach for intrusion detection that used a combination of mechanisms to achieve these goals. High detection accuracy is accomplished through the application of a two-layered detection model (local and cooperative) that honors partial and lack of evidence against a suspicious activity in the detection decision through the application of Dempster-Shafer theory (DST). However, the uniqueness of such application as compared to similar approaches is the avoidance of weighting nodes’ trustworthiness in evidence consideration, which can result in inaccurate results when high numbers of malicious nodes exist in the network. Due to the highly dynamic nature of MANETs, evidence collection throughout the process of intrusion detection might suffer from cases where partial or no evidence is collected. Such cases, in the proposed approach, were not taken as negative evidence against the suspicious node as opposed to similar techniques found in the literature, such as the Bayesian theorem (Gordon & Shortliffe, 1984). Instead, through the application of DST, each node calculates and sends out its own degree of belief in the collected evidence against the suspicious node, alongside the evidence itself. The degree of belief in this context represents the level of certainty towards the collected evidence from a certain node. Based on the received degrees of belief, each piece of evidence or lack thereof is given a certain weight. Those weights are then honored in the intrusion detection process to calculate the final decision against the suspicious node. This, in turn, targeted reducing false alarms rates in cases of accidental behaviors, such as high node mobility or nodes leaving the network.

All of the current solutions rely on collaborative calculations that require extensive back and forth communications to establish the reliability of information exchange between nodes. However, they tend to fall short when there is a high number of malicious nodes in the network. Besides, their high bandwidth consumption for exchanging detection-related information causes disruption of normal network operations and potential loss of such information. Therefore, for the reduction of this large volume of information dissemination found in current solutions, this research targeted this issue through employing a self-sufficient technique for each node to determine its own social circle. This enables nodes to limit information exchange, such as cooperative detections or false alarms generations, with only certain sets of nodes. This, in turn, has two benefits. First, detection-related information is only sent to strongly-tied nodes, which eliminates the possibility that malicious nodes receive such information and temporarily adjusts their behavior to avoid exclusion from the network. The second benefit relies upon having intrusion cooperation limited to a node’s social circle, instead of the entire neighborhood. Such restriction ensures reliable detection information through excluding anonymous nodes from cooperative investigation operations. At the same time, it decreases information exchange, even in such short-lived operations.

The proposed approach employed the concept of social communities, which has never been applied to MANETs security before, to improve detection accuracy. This was done by building strongly-tied communities that enable the exchange of reliable detection information solely among nodes' social circles. This type of application addresses a major limitation that exists in current approaches: their reliance on intrusion reports from anonymous nodes, which can result in high false alarms rates. As such, in this research, intrusion-related reports are only considered if they come from a node belonging to a strongly-tied community.

The combination of the mechanisms, mentioned above, successfully attained a highly efficient CIDS for MANETs through achieving high detection accuracy and low bandwidth consumption, as compared to the existing solutions.

Relevance and Significance

MANETs have experienced a rapid growth in their applications with alignment to the proliferation of mobile devices and technologies. The ability of MANETs to function without the need for Internet connectivity or fixed infrastructure have led to increased deployments of such networks towards various scenarios, such as disaster relief operations and military tactics

(Johansson, 1999). The unique characteristics of these networks, such as the lack of infrastructure, limited bandwidth, constrained resources, and the essential need for cooperation, renders these networks susceptible to various types of attacks. These attacks may result in devastating consequences, such as loss of confidential information, service interruptions, or even eradication of the entire network (Vij & Sharma, 2016). Such disastrous potential of these attacks has made the application of security for MANETs a challenging mission (Hubaux, Buttyán, & Capkun, 2001).

Different preventive techniques have been proposed to secure such networks. However, the mechanisms utilized in these propositions cannot guarantee an intrusion-free network. Thus, a second line of defense is needed to achieve this goal, represented by the application of IDSs. There exists a plethora of research in the area of MANETs security through the development and applications of various IDSs. Three primary architectures exist in the current body of literature for the implementation of IDSs in MANETs: stand-alone, hierarchical, and cooperative. In the stand-alone architecture, each node analyzes and detects attacks on its own through a locally installed IDS. However, the reliance on locally collected data affects detection accuracy and limits the range of malicious attacks that can be uncovered (Şen & Clark, 2009). In a hierarchical setup, a MANET is divided into clusters, which each have a dedicated monitoring node to perform detection functions. The main drawback of this architecture is the high risk of leaving the entire network in jeopardy when such nodes get compromised or drop out of the network (Cannady, 2010). In a cooperative configuration, each node has its own IDS and cooperates with others in attack detection. The use of a cooperative architecture over the other types can be reasoned to the lack of central traffic points in MANETs, which mandate the need for real-time cooperative detection techniques for an effective IDS (Mahmood, Amin, Amir, &

Khan, 2009). Additionally, a broader view of the network is deemed significant to defend a network against insider attacks, which can only be achieved through nodes' cooperation in the detection process (Morais & Cavalli, 2012).

A large body of knowledge exists in the area of CIDS implementations, pertaining different methodologies and mechanisms. However, all of these implementations rely on extensive dissemination of detection information throughout the network. This reliance has two major consequences: disruption of normal routing operations and potential loss of detection information, leading to degradation of detection accuracy. The other downside found in the existing systems is the high rate of false alarms resulting from their reliance on unreliable detection reports obtained from anonymous nodes throughout the decision-making process. Making intrusion decisions based on anonymous reports can lead to false accusations towards innocent nodes in case of fake information generated by malicious nodes (Razak, Samian, & Maarof, 2008). These vulnerabilities found in existing approaches hinder the efforts towards a widespread adoption of MANET applications against more areas since security forms the main obstacle towards such adoption (Nadeem & Howarth, 2013). Therefore, there is a need for an efficient CIDS that can detect intrusions with high accuracy while maintaining an efficient bandwidth consumption.

Barriers and Issues

The unique characteristics of MANETs represented in their continuously changing network topologies, infrastructure-less nature, and lack of central networking points make the mission of securing such networks an inherently difficult one. Despite the extensive research efforts put forth in the area of intrusion detection, the development of an efficient IDS that is able to detect complex attacks remains an obstacle in the way of security researchers (Cannady, 2013). This is largely due to a number of contributing factors, such as the continuous advancements in technologies, attackers' creativity, and the dynamic nature of threats. Cannady (2013) outlined the criticality of achieving timely and accurate attack detection when implementing an IDS for MANETs. This aligns with the primary goals of this research, as the proposed system was intended to achieve high detection accuracy through decreasing false positives while avoiding overwhelming bandwidth consumption to accomplish timely completion of the detection process, towards meeting Cannady's criterion for an efficient IDS. However, the primary lesson learned from previous research studies in this area is that attempting to achieve such timely accuracy requires a high degree of complexity and extensive efforts, which were anticipated to accompany this research study.

The acquisition of representative attack data was another barrier for this research. This type of data is critical for validating the effectiveness and viability of this study's proposed approach in real-world attack scenarios and implementations. However, since the proposed approach was intended to work with real-time network traffic, it was not sufficient to obtain and operate on static data to simulate malicious activities. Previous research studies that have attempted the implementation of various attack scenarios were examined for this purpose. Some edits and manipulation for these implementations were necessary to accommodate the required scenarios in this research. Nonetheless, the validity of such implementation as efficient representations of real-world attack scenarios is still an outstanding area of critical concern. The Defense Advanced Research Projects Agency (DARPA) provides a number of sample datasets, consisting of network traffic and audit logs, for the evaluation of intrusion detection techniques. However, these datasets do not include attack data that are specific to MANETs. As such, common methodologies implemented by previous studies were followed for attack representations in this research. However, in the face of increasingly creative attackers that continuously vary existing attack patterns, such implementation might not necessarily represent all scenarios for real-life attacks.

Assumptions, Limitations, and Delimitations

To manage the scope of the study, the proposed research was conducted in a simulated environment using ns-2. Network prototypes, validation of the efficiency and applicability of the proposed system, and data analysis were performed in that environment. The simulated network prototypes produced different attack scenarios that are common to MANETs. Prominent researchers have used common methodologies to simulate such attacks. Their work was validated and piloted in the attack simulation process for the proposed system. However, one limitation here lied in the adaptability of attackers. As networks continue to advance, attackers evolve and become better at devising innovative ways to launch security attacks against MANETs. Thus, the simulated environment cannot represent every threat level that MANETs might be exposed to in real-life scenarios. However, the system was implemented to be applicable to every MANET.

This research was not intended to result in an IDS that can detect all types of attacks. Instead, the proposed system demonstrated how the application of social communities accompanied by DST can enhance accuracy while minimizing communication overhead throughout the intrusion detection process. A possible extension of this study is to implement detection mechanisms against a more comprehensive set of attacks against MANETs. This was kept in mind during the design of the proposed system to potentially simplify future extensions.

Summary

The dynamic nature of MANETs represented in features such as the lack of fixed infrastructure, constant mobility, and dynamic topology have made designing effective security mechanisms a difficult challenge. Researchers have put forth extensive efforts towards MANET’s security through the implementation of IDSs. Since cooperation between nodes in a MANET is mandatory to achieve efficient detection (Mahmood et al., 2009), a large number of studies have proposed a variety of CIDS implementations. However, all of the proposed CIDS in the current literature suffer from high bandwidth consumption and false alarm rates. Due to the resource-constrained nature of MANETs, high bandwidth consumptions can have adverse effects on normal network operations. On the other hand, the reliance current solutions have on intrusion detection reports from anonymous nodes may lead to false accusations towards innocent nodes, which in turn, decreases detection accuracy.

The goal of this research was to apply the concept of social communities and DST towards the implementation of a CIDS that is capable of achieving high detection accuracy while minimizing bandwidth consumption. Implementation of the proposed research was able to produce an efficient CIDS that is capable of accurately detecting security attacks in a timely manner.

This Dissertation Proposal is structured as follows: Chapter 2 presents a comprehensive review of literature relevant to intrusion detection in MANETs. Chapter 3 details the research methodology that is going to be followed in the implementation of this research.

Chapter 2

Review of Literature

Introduction

This chapter presents a summary of relevant research studies in the current body of knowledge. As will be shown, these studies support the research presented in this Dissertation Proposal pertaining to MANETs, their security issues, and current security solutions. The review starts with a survey of what makes MANETs such unique networks. This survey will include reviewing their characteristics, security issues, security requirements, and possible attacks. The review then progresses towards the specific domain of this research: discussing intrusion detection for MANETs in relation to the various mechanism, techniques, and architectures followed in IDS implementations.

Overview of MANETs

Characteristics

Early work on ad hoc networking traces back to 1972 when DARPA introduced their Packet Radio Network (PRNet) project. The work focused on the application of packet-switching techniques, such as store-and-forward routing and bandwidth sharing for mobile wireless networking (Jubin & Tornow, 1987). DARPA continued their development in this field, resulting in the introduction of Survivable Radio Networks (SURAN) in 1983. This new technology addressed scalability, energy consumption, and security issues found in PRNet (Beyer, 1990). In 1987, DARPA designed the Low-cost Packet Radio (LPR) technology, which enhanced the scalability and network management aspects of packet radio networks (Jubin & Tornow, 1987).

Advancements continued to be made, and in 1994, DARPA started the Global Mobile (GloMo) program targeting the support of multimedia connectivity in wireless devices anytime and anywhere (Leiner, Ruther, & Sastry, 1996). With all these advancements in ad hoc networking, the US Army's Tactical Internet (TI) that was developed in 1997 is considered the largest-scale implementation of a mobile wireless multi-hop packet radio network. Numerous amounts of work have followed to build on top of this implementation, exposing the potential of ad hoc networking to the research community, outside the military domain (Chlamtac, Conti, & Liu, 2003).

Early work on MANETs was sharply focused on military applications. The proliferation of wireless networking and mobile devices, as well as the ongoing advancements in these technologies, have increased the widespread application of MANETs implementations across medical, military, financial, and various other fields. In military applications, MANETs provide a decentralized configuration that is necessary and advantageous for these types of operations. However, the same configuration can be employed to non-military scenarios where no infrastructure exists, such as construction sites, disaster relief areas, conferences, and many others (Singh, Woo, & Raghavendra, 1998; Johansson, 1999). A MANET is a multi-hop wireless network consisting of a number of wireless devices that have the ability to perform routing on their own. As such, these nodes cooperatively maintain connectivity inside a MANET (Singh, Woo, & Raghavendra, 1998). The movements of such nodes are arbitrary because they are free to move in any direction needed. A node can be an integrated unit of network devices, such as laptops and handheld devices, or it may consist of physically separated network devices. In order for a node to connect to others in a MANET, wireless connectivity in the form of a dynamic, multi-hop ad hoc network must exist. This is done through a variety of wireless transmitters and receivers, which are usually implemented with different kinds of antennas that nodes must be equipped with (Corson, Macker, & Cirincione, 1999).

In contrast with other types of networks that rely heavily on Internet connectivity to operate, MANETs are autonomous systems of nodes that can fully operate in isolation without the need to be connected to the Internet. In traditional mobile networks, although a user might change location due to continuous movements, a variety of fixed networking infrastructures are typically used to support such routing with other devices. However, in MANETs, the entire routing infrastructure moves with users' devices, leading to frequent and unpredictable changes in the network topology. As such, a fixed routing infrastructure is no longer useful in such networks (Corson et al., 1999).

The main characteristics of MANETs that differentiate them from other types of wireless networking can be summarized by the following (Corson et al., 1999; Chlamtac et al, 2003; Johansson, 1999):

- Dynamic Topologies: since nodes in MANETs are free to move arbitrarily and route changes, frequent network partitioning and packet loss are natural outcomes of the constant change in topology in such networks. Nodes in MANETs vary in regard to their transmission range and software/hardware configurations. This results in different processing powers, causing a dynamic, heterogeneous networking environment.
- Infrastructure-less Networking: nodes in MANETs do not rely on fixed infrastructure to communicate with others residing within their wireless range. Should communication be needed with an out-of-range node, other nodes can be used as intermediates, relaying the

message hop by hop until it reaches its destination. As such, it can be seen that each node in a MANET acts as a router, enabling message forwarding and information sharing inside the network.

- Resource constrained: applications and services provided by mobile nodes in MANETs are restricted due to the limited energy, storage, and processing power. Each node in a MANET acts as both an end- and intermediate-system. Resource limitations become a big problem in MANETs as a network moves towards scaling to a larger number of nodes and requires more resources to perform the different routing operations.

Nodes in MANETs have the unique ability to instantaneously form routes among themselves as they roam freely in the network. The infrastructure-less nature of MANETs allows this dynamic routing to occur without interruptions to normal network operations. This is achieved through nodes exchanging their routing information with each other as they move around the network to establish routes dynamically without disrupting communications. Such a capability allows MANETs to be a suitable choice for special deployment scenarios. These include challenging deployment environments, such as disaster relief locations, where addressing terrestrial and geographical limitations requires a high distribution of network devices without any fixed base stations or central networking units. (Deng, Li. & Agrawal, 2002).

Although the dynamic configuration of MANETs allows network operations to survive through high-mobility scenarios, it does impose certain requirements and obstacles for MANET routing protocols. Depending on the network size, which is defined by the number of nodes in a MANET, communications can suffer from data loss and continuous interruptions (Wang & Li, 2002). This can happen, for instance, with small MANETs in which mobile nodes move in opposite directions and cause network partitioning where a network is divided into multiple disconnected sections. Routing disruptions, packet delays and loss can occur in such situations, causing the network to suffer from high overhead in an attempt to rediscover routes to the disconnected partitions (Tan & Seah, 2005).

Network partitioning caused by the continuously changing topological formations in MANETs has occupied many researchers in efforts to find solutions that might lessen the impact of communication disruptions. These efforts focused mainly on predicting nodes' mobility patterns to avoid network partitioning (Wang & Li, 2002). Such efforts were put forth due to the criticality of some MANET applications, such as battlefield deployment where certain levels of network availability and quality of services must be guaranteed. To that extent, routing protocols along with the applications deployed on nodes must be designed to accommodate the high rates of topological changes in the network (McDonald & Znati, 1999).

The resource-constrained nature MANETs also makes them prone to network availability challenges. When nodes communicate with each other, they utilize a shared wireless medium to relay their messages back and forth. This medium is limited in capacity as it conforms to the inherently bandwidth-constrained MANETs. The more nodes utilize the same channel, the higher the chances of interference and link errors to occur. These, in turn, can result in interruptions of communications as well as information loss, which can have devastating consequences in mission-critical networks (Yang et al., 2004).

The primary issue with limited-capacity wireless channels in MANETs lies in the inevitable unpredictability of communication quality between nodes. This unpredictability is caused by the highly variable environmental conditions surrounding the deployment of MANETs, which are difficult to anticipate in advance (Xiao, Seah, Lo, & Chua, 2000). Such unpredictability continues to contribute to a lower quality of service in MANETs when compared to their fixed-wireless or even wired counterparts. On the other hand, nodes in MANETs usually come equipped with different transmission rates subject to their wireless antenna design. However, nodes with high transmission power cannot guarantee a full utilization of such a power. This is due to the unpredictable effects of the shared wireless channel, which might cause interference, noise, and continuous collisions. All of these factors have adverse effects on communication and service quality in MANETs, causing such networks to be constantly vulnerable to availability problems (Garg & Mahapatra, 2009).

Despite the increasing adoption of MANETs in areas other than military and disaster recovery operations, the applications that can be deployed on these networks are somewhat constrained by nodes' limited energy and processing power. This has pushed researchers to devise various methods to enhance energy conservation while minimizing processing overhead incurred by routing protocols (Kim & Jang, 2006; Mikki, 2009; Maleki, Dantu, & Pedram, 2002). By directing their efforts towards allowing more room for application to utilize such energy and processing, researchers hoped to increase the adoption of MANETs in a wider range of civilian deployments. In addition, resource conservation is deemed critical for MANETs to sustain a longer network lifetime (Kim & Jang, 2006).

Along with a detrimental impact on the feasibility of installing certain applications on nodes, the inherent energy constraints in MANET nodes also affect hardware and signal processing operations. This happens because any packet operation such as sending and receiving, or even standby hardware operations, consumes a nontrivial amount of a node's battery power (Goldsmith & Wicker, 2002). This inevitable drainage of battery power not only imposes high restrictions for designing communication protocols, but it also prompts necessary considerations for conserving such energy, even in sleep modes where nodes are idle. Such considerations are usually applied in the hardware and operating system design for mobile devices (Goldsmith & Wicker, 2002).

Although the unique nature of MANETs sometimes poses as a limitation towards the deployment of these networks in certain scenarios, it also serves as an attractive factor for a large number of applications. Data, home, and sensor networks, for instance, have experienced a wide and successful adoption of MANETs. This is all thanks to a dynamic nature that enables sufficient flexibility for these applications to operate in versatile environments. Data network applications for MANETs provide wireless connectivity between laptops, handheld devices, and other information devices in commercial settings. Home network applications, on the other hand, can support automatically adjusting light and room temperature or act as security alarms alerting home owners when odd movements are detected outside of their houses (Huhns, 1999).

Another promising application of MANETs with high potentiality of expansion can be seen in sensor networks. These applications can be of a great benefit for both military and commercial deployments. Examples of military-targeted applications of sensor networks include optical, chemical, and biological sensors that can serve as first lines of attack detection in war zones. On the other hand, commercial usage of these sensors includes gas, water, and electricity meters that can help consumers regulate their usage for high-energy consumption devices, such as water heaters and air conditioners (Goldsmith & Wicker, 2002).

Nonetheless, the inherent energy constraints, processing limitations, and infrastructure-less nature of MANETs pose the requirements for different networking strategies to be implemented than those for fixed wireless networks. As such, a tremendous amount of research has been directed to design a wide range of network management models to provide efficiently functional communications with regard to limitations in MANETs (Abolhasan, Wysocki, & Dutkiewicz, 2004).

Routing

Due to the limited transmission range of nodes in MANETs, each node has to act as both a host and router, forwarding packets to out-of-range nodes as multiple hops might be needed for packet exchange. Route discovery and maintenance also fall under nodes’ responsibility (Royer, 1999). Since MANETs are infrastructure-less by nature, this means the lack of a fixed routing infrastructure forces nodes to dynamically establish routes among themselves to communicate with each other across the network. Such routes are formed instantaneously, providing an extremely flexible communication environment for the continuously changing network topology (Deng, Li, & Agrawal, 2002).

In traditional wired networking, there are two primary routing algorithms: link-state routing and distance-vector routing. In link-state routing, each node in the network maintains an updated routing table through periodically broadcasting the link-state costs of its neighboring nodes. Tables are then updated through the application of the shortest-path algorithm to determine the next hop nodes for destinations. In distance-vector routing, each node maintains a table of distances to each neighboring node, allowing nodes to calculate the shortest path to each destination (Abolhasan et al., 2004). Although these two protocols are widely used in wired networks, they are inefficient for the resource-constrained MANETs due to their high bandwidth consumption when performing their frequent route updates. To overcome such inefficiency, a number of routing protocols have been specifically developed for MANETs. These protocols can be categorized into three groups: proactive, reactive, and hybrid (Abolhasan et al., 2004).

In proactive routing, each node maintains information about all other nodes in the network. Routes to all destinations are determined at the network initialization phase. Different tables are used to maintain route information and prevent them from getting out of date due to continuous route updates (Royer, 1999). These updates are of two types: periodic and triggered. In periodic updates, the entire routing table of each node is broadcasted to all other nodes. On the other hand, triggered updates occur only when a node detects changes in its neighboring nodes. Thus, only these changes are broadcasted. This type of routing is also called table-driven routing (Srivastava et al., 2014). There are a number of routing protocols following this strategy, such as Destination-Sequenced Distance Vector (DSDV), Wireless Routing Protocol (WRP), and several others that operate in a similar fashion (Royer, 1999; Shenbagapriya & Kumar, 2014). However, the main differences among the different protocols in proactive routing are the number of tables used to maintain information, types of information stored in each table, and the mechanisms used to circulate such information.

Destination-Sequenced Distance Vector Routing (DSDV) provides an example to illustrate how proactive routing works. DSDV improves on the classical Bellman-Ford routing mechanism through the elimination of loops in routing tables. Nodes record every possible destination in the network and the number of hops it takes to reach that destination in their routing tables along with a sequence number. This number allows nodes to distinguish stale routes, which in turn, eliminates routing loops. That is, a higher sequence number indicates a more recent route to a destination. When a node wants to transmit packets to a destination, it only considers routes with the most recent sequence numbers. However, if a node receives two updates with the same sequence number, it selects the one with the smaller hop count to the destination (Royer & Toh, 1999). Routing updates can be in a form of a “full dump” in which all available routing information is broadcasted to the network. Smaller incremental updates that only send updates from events after the last “full dump” are also employed in this protocol. These updates contain the address of a destination, a sequence number received for that destination, and the number of hops needed to reach it (Abolhasan et al., 2004).

In an attempt to reduce network congestion from frequent route updates, mobile nodes in DSDV may employ a certain broadcasting delay before sending subsequent updates. This delay is determined through a mechanism that keeps track of settling times for routes, which are defined as the times when routes to a destination fluctuate before the route with the best metric is received (Royer & Toh, 1999). However, DSDV does not thrive in large networks as the excessive amounts of routing overhead would occupy a large portion of the network and result in potential communication disruptions (Abolhasan et al., 2004).

The main advantage of proactive routing is the immediate availability of routes when needed, which decreases delays in packet delivery (Sudarsan & Jisha, 2012). However, the main disadvantage of these protocols is their excessive amounts of bandwidth usage for route update propagations, which might affect the scalability of the network (Abolhasan et al., 2004). Another disadvantage is the constant increase of the routing tables’ sizes as more nodes join the network (Srivastava et al., 2014).

Reactive routing protocols attempt to reduce bandwidth usage and overheads that exist in proactive protocols by maintaining information on active routes only. Along with this, routes are determined and maintained on demand. As such, no periodic flooding of information is needed when a topology change occurs. Therefore, this type of routing is also called on-demand or source-initiated routing (Royer, 1999). When a node needs to send a packet to a destination with an unknown route, a route discovery process is initiated. During this process, the network is flooded with route request packets. When the route is determined through an intermediate or destination node, route reply packets are sent back through link reversal if bidirectional links were used or by flooding the network with such packets (Royer, 1999). Maintenance of the route ends when the destination node becomes inaccessible or no longer desired (Srivastava et al., 2014). Fewer route updates and maintenance are the main themes of reactive routing. However, increased delay in packet delivery may occur due to the time consumed in the route discovery phase (Sudarsan & Jisha, 2012). A large number of protocols exist in reactive routing, such as ad hoc on-demand distance vector (AODV) and dynamic source routing (DSR).

To illustrate how this type of routing works in a MANET, we will briefly examine the AODV protocol. AODV builds on the DSDV algorithm. However, the improvement this protocol offers over DSDV is the elimination of expensive route information exchange through the creation of routes on an on-demand basis (Abolhasan et al., 2004). Nodes that are not in a selected path neither maintain routing information nor participate in routing tables' exchanges. Hence, the protocol is referred to as a "pure on-demand route acquisition". When a node wants to transmit packets to a destination with an unknown route, it starts a route discovery process by sending a route request (RREQ) packet to its neighbors who then forward the request to their neighbors. The process continues until the destination or an intermediate node with a recent route to the destination is found (Royer & Toh, 1999).

Sequence numbers are used in this protocol to eliminate loops and ensure recency of path information. A sequence number is a monotonically increasing number that is maintained by a node originating a route request. This number is then used by other nodes to determine the freshness of information received from the originating nodes. Therefore, the higher the sequence number, the fresher the received route to a destination. Each node checks if it receives the same sequence number in a routing message. That way, if a node detects a duplicate sequence number, it drops the routing message to avoid potential loops. Additionally, this sequence number is incremented each time a node initiates a route discovery, which along with the IP address identifies the RREQ. Intermediate nodes can reply with the route information only if they have a route to the destination with a greater sequence number than that contained in the RREQ (Patel, Patel, Kothadiya, Jethwa, & Jhaveri, 2014).

During the process of RREQ, a reverse path is established as each intermediate node records the address of the neighbor from which it received the first copy of the RREQ. This path is then used when the destination is found to unicast a route reply (RREP) packet to the neighbor that a node received the first copy from, until it reaches the original requesting node. A forward path is established during the process of RREP, in which each node records the information of the neighbor that sent the RREP. If a node along an established route moves and needs to maintain an updated record of routing information during the route discovery phase, a link propagation failure message (RERR) is sent from the node's upstream neighbor to all its neighbors. These neighbors continue to forward the failure to their upstream neighbors until it reaches the source node. To keep a list of active neighbors, nodes listen to packet retransmissions of their neighbors to ensure the availability of their next hop. If such retransmissions are not heard from a certain node, nodes send a "Hello" message to check if the next hop is still alive. Such messages are used by AODV as periodical local broadcasts to keep nodes informed about others in their neighborhood (Royer & Toh, 1999).

All of these protocols possess comparable mechanisms in regard to route discovery and maintenance operations (Abolhasan et al., 2004). They also share similar transmission routing costs when considering packet transmission between two nodes with no prior communications (Patel et al., 2014).

Hybrid routing protocols for MANETs were introduced to overcome network scalability and latency issues found in both proactive and reactive protocols. This is achieved by allowing close-proximity nodes to have full knowledge of the routes between them, which reduces overheads resulting from route discovery (Saeed, Abbod, & Al-Raweshidy, 2012). To accomplish such a task, proactive routing is used for neighboring nodes to maintain routes between them. When these nodes need to transmit data outside of their circle, reactive routing is used to determine the required routes through route discoveries (Sudarsan & Jisha, 2012; Patel et al., 2014). Hybrid routing protocols have the ability to quickly switch between proactive and reactive mechanisms (Srivastava et al., 2014). Different hybrid routing protocols were developed, such as the zone routing protocol (ZRP) and distributed dynamic routing (DDR).

ZRP provides an example to illustrate the functionality of hybrid protocols. This protocol is based on dividing the network into multiple routing zones. ZRP defines a zone for each node with a radius of (p) that includes all the neighboring nodes who are (p) hops away. Inside each zone, ZRP utilizes locally proactive routing defined as the Intra-Zone Routing Protocol (IARP). This protocol uses table-driven routing that relies on nodes to continuously update routing information regarding nodes in their local zone. To communicate with distant zones, ZRP uses its reactive routing component called Inter-Zone Routing Protocol (IERP). This protocol is reactive in nature. Thus, it issues route queries on demand whenever a route to an out-of-zone node is requested (Schaumann, 2002).

When a node wants to transmit packets to a destination, it checks whether that destination is within the local zone by querying the information provided by IARP. If so, the node uses proactive routing through IARP to transmit the packet. However, if the destination is outside the local zone, the node must use reactive routing through IERP (Schaumann, 2002). This is done by having the requesting node initiate route request packets to other nodes in its local zone. If a receiver node has knowledge of the destination, it responds with a route reply to the requesting node. If not, the request packet continues to bordercast until it reaches the destination or an intermediate node that has a recent path to the destination. Bordercasting is a process of requesting route information from nodes located at zones' borders. Bordercasting is specific to the IARP that is provided by the Bordercast Resolution Protocol (BRP). This protocol creates a bordercast tree map containing nodes that are located on the local zone's border by querying topology information provided by IARP. Such queries are only initiated when reactive routing is needed to a distant destination. ZRP uses this protocol to direct route requests to distant zones through the process of bordercasting (Beijar, 2002).

For the reply packet to be able to make it back to the requesting node across the zones, each node that forwards the packet appends its own address. This way, when the destination receives the packet, it copies these saved addresses to the route reply packet so that it can be routed back to the source. On the other hand, nodes along the path to the destination save the next-hop address to their routing tables for future use (Beijar, 2002).

A critical consideration when it comes to designing a MANET with ZRP as the main routing protocol is identifying zones' radiuses. The radius of the routing zones is a determining factor in ZRP to decide what type of routing should be used and when. That is, the smaller the radius, the more reactive routing is used, causing a significant increase of routing traffic in the network. On the other hand, a bigger zone radius means more proactive routing and more inter-zone traffic to maintain the view of each zone (Abolhasan et al., 2004).

The main similarity between these protocols is their zone-based nature, meaning they tend to partition a network into a number of zones/clusters. However, each protocol employs different mechanisms to form these zones (Abolhasan et al., 2004).

There have been tremendous efforts put forth towards the design and development of the above-mentioned routing protocols. However, there is no one-solution-fits-all approach when it comes to MANET applications. A thorough understanding of a MANET's condition and requirements is needed for selecting the suitable protocol for each implementation. Such a selection should balance efficiency and performance, and it should account for security, which plays a significant role in the survivability of the network (Saeed et al., 2012; Deng et al., 2002).

Security Issues

Security has become a primary concern in the path to providing protected communications between nodes in MANETs. Since each node acts as both a host and router, both legitimate and malicious nodes can access wireless channels. A fundamental question when examining security for MANETs is how to protect the basic connectivity of each node to enable safe data transmission throughout the network (Yang et al., 2004)? MANETs possess several unique characteristics that make the quest for security a nontrivial challenge, including the following:

- No Predefined Boundary: the nomadic environment of a MANET does not impose physical boundaries. This allows nodes to freely join or leave the network as they desire. Malicious nodes can immediately start communicating with nodes in a network once they are in the same wireless range (Sheik, Chande, & Mishra, 2010). This lack of physical protection raises the chances for a network to get compromised (Zhou & Haas, 1999).

- Lack of Central Authority: MANETs don't have a central node for network control and management (Raj, Bharti, & Thakur, 2015). This means that traffic cannot be monitored from a centralized station. Instead, it is distributed among different nodes. This makes the detection of attacks a challenging mission because the lack of security association affects trust between nodes (Sheikh et al., 2010). At the same time, having a single, central authority can cause major vulnerabilities; if such a node gets compromised, the entire network is sabotaged (Zhou & Haas, 1999). Thus, distribution of traffic monitoring and attack detection cooperatively among nodes in MANETs can be viewed as mandatory to overcome the security issues related to a single or lack of central authority in these networks (Mahmood et al., 2009). This is because such distribution provides a broader outlook on the network, which is considered crucial to achieve effective attack detection (Morais & Cavalli, 2012).
- Dynamic Topology: nodes in MANETs roam independently and in any desired direction, causing nodes to drop in and out of the network as they wish (Deng et al., 2002). This constant mobility of nodes inside the network causes changes to the underlying topology, which affect trust establishment because such changes might disrupt trust relationships among nodes. Attention to the different challenges in this dynamic nature is a cornerstone in designing security solutions for MANETs (Goyal, Parmar, & Rishi, 2011).
- Cooperativeness: due to the distributed nature of MANETs, cooperation between nodes is needed for such networks to function properly. This cooperation compensates for the lack of fixed infrastructure. However, the lack of a centralized authority to enforce such cooperation might lead some nodes to behave selfishly or maliciously (Conti, Gregori, & Maselli, 2004). Besides, routing protocols assume that all nodes in a MANET are non-

malicious and cooperative in nature. This poses a significant vulnerability when a node gets compromised and starts disrupting network communications (Goyal et al., 2011).

- Resource Availability: nodes in MANETs each operate on a limited power supply through their built-in batteries. This presents inevitable processing and operational restrictions, which cause limitations to the types of security mechanisms that can be implemented to secure such networks (Raj et al., 2015). Such limitations give attackers ample opportunities to direct expensive computational tasks towards victim nodes, which might result in draining their battery power (Mishra, Nadkarni, & Patcha, 2004).
- Scalability: as nodes in MANETs are free to join or drop out, scalability of these networks changes frequently. This makes the prediction of the number of nodes a difficult task when routing and designing various services for these network (Raj et al., 2015). Scalability issues are more challenging in large networks with high mobility, causing excessive network routing and transmission overhead (Hong, Xu, & Gerla, 2002).
- Shared Wireless Medium: opposite to wired networks, which utilize a dedicated channel to connect two hosts, nodes in MANETs share the same access medium to communicate. Such a medium is susceptible to signal interference and external noise, affecting the reliability of packet transmission in the network (Sheikh et al., 2010). Since communications in MANETs are broadcast in nature, data transmission can be interrupted by existing malicious nodes (Raj et al., 2015). Various implementation scenarios for MANETs occur in hostile environments, such as military operations. This leaves a high possibility and expectations for attacks against such a vulnerable wireless medium (El Defrawy & Tsudik, 2008).

- Adversaries inside the network: nodes inside the network can get compromised and start behaving maliciously. Such insider threats are harder to detect than external ones (Sheik et al., 2010). Insider threats can cause the inability of other nodes to detect incorrect or malicious behaviors. These insider adversaries can cause different kinds of transmission problems, such as false advertisements of routes, message dropping, and incorrect link-state information (Mishra, 2004).

Maintaining connectivity for a high mobility medium in a MANET is mandatory for the survivability of the network. However, the different vulnerabilities in such networks make them susceptible to various types of attacks from malicious sources targeting different services and functionalities. Such attacks might result in service interruptions, data loss, stealth of confidential information, or in some cases, eradication of the entire network (Vij & Sharma, 2016).

Security Requirements

Security is a significant consideration in MANETs, especially for those applications involving highly secretive information, such as military operations (Zhou & Haas, 1999). The path to an effective security implementation in a MANET should consider the following requirements:

- Authentication: this ensures the originality of communications between two nodes, eliminating the possibility of a malicious node masquerading as a trusted one (Djenouri, Khelladi, & Badache, 2005). That is, a communicating node can trust that the other end of the connection is truly the designated destination (Abdelaziz, Nafaa, & Salim, 2013).

In general, authentication is considered an assurance that all participants in a data communication are verified entities and not impersonators (Goyal, 2011).

- Availability: this sustains networking functionalities and ensures resources are available at all times without any interruptions (Abdelaziz et al., 2013). This means that such services should be available for consumption by nodes in a MANET, even in the presence of an attack. Interruption of services could potentially put the availability of resources in a dangerous position, impacting the entire network operations of a MANET (Raj et al., 2015).
- Integrity: when two nodes communicate in a MANET, they expect to receive the exact information that was sent from one to another. Integrity ensures that the exchanged messages are authentic, uncorrupted, and untampered with by a malicious node (Djenouri et al., 2005). Malicious tampering might include modification of contents, deleting and recreating the message, or changing certain bits of information (Goyal et al., 2011). However, some messages might get accidentally modified during transmission due to a network failure and are not necessarily results of malicious acts (Raj et al., 2015). In all cases, integrity should protect from malicious and accidental modifications of messages between two communicating parties.
- Confidentiality: MANETs have a wide variety of applications in various military and commercial fields. Sensitive information might flow during communication in such operations (Raj et al., 2015). Confidentiality ensures that such sensitive information remains undisclosed to anybody other than the authorized parties, and it ensures the information cannot be understood by unauthorized entities (Abdelaziz et al., 2013; Djenouri et al., 2005).

- Nonrepudiation: repudiation refers to the denial of a node that was involved in a data communication of its full or partial participation in such communication. Consequences of repudiation can lead to damaging results, such as a node denying the participation in military communications leading to compromising the confidentiality of secret information. Nonrepudiation guarantees that a node cannot deny sending/receiving a certain message (Djenouri et al., 2005).

Any implementation of security solutions for MANETs should pay close attention to the above-mentioned security requirements. Additionally, a number of significant metrics must be incorporated in the design of such solutions. These metrics are referred to as "security parameters." Neglecting these parameters can increase the potentiality of serious security vulnerabilities when implementing security solutions for MANETs (Dorri et al., 2015). These security parameters can be summarized as follows:

- Communications Overhead: security solutions often introduce special control and intercommunication packets that aim to pass messages around between the different components of these solutions for management and reaction purposes (Yang et al., 2004). However, failing to limit the number and size of such messages might cause normal network operations to suffer. This, in turn, can result in network congestions, packet loss, and disruption of packet routing among nodes in a MANET. Packet retransmission is another potential consequence that results from packet loss due to the imposed overhead from security solutions. This would eventually affect nodes' battery power levels because of continuous retransmission and would ultimately cause nodes to potentially halt during communication to conserve energy (Dorri et al., 2015).

- Energy Consumption: nodes in MANETs are energy-constrained due to their reliance on battery power to operate in the network. Security solutions tend to add extra processing overhead to apply various defense mechanisms against malicious activities, resulting in an increased energy consumption on nodes. This is usually viewed as an inevitable consequence that makes optimization of battery consumption a critical consideration in the design of security solutions. Nonetheless, such optimization is highly challenging and extensive research has been solely dedicated to this purpose (Biswas, Nag, & Neogy, 2014; Kerache et al, 2017; Lupia & Marano, 2016). Keeping energy efficiency in mind when developing security solutions for MANETs would result in positive effects on the network. This would be beneficial because decreasing nodes' battery consumption may lead to a longer lifetime of the network in general (Kerache et al, 2017).
- Processing Time: for security solutions to respond to a suspicious activity in the network, certain processing on the received/sent packets must occur. Based on such processing, a suspicious node might be marked as malicious or, in some cases, isolated from the network (Nadeem & Howarth, 2013). While these actions are mandatory to impose security in the network, so is the need for minimizing processing times accompanied with these actions. During such processing times in highly dynamic environments like MANETs, this is necessary because attackers might move to a different location or even drop and rejoin the network with a

different identity. This can potentially render any actions taken by these security solutions obsolete (Dorri et al., 2015).

- Scalability: the plethora of MANET security implementations found in the current literature includes appealing solutions that largely target issues of energy consumption and communications overhead. Addressing these targets might be all that is needed for small networks. However, larger networks might require other actions. As the number of nodes in a MANET increase, so does the challenge for security solutions to preserve their promised performance measures. This is due to the unpredictability of the network size and the ever-changing topological conditions in MANETs. Thus, accounting for network scalability is a critical consideration for those attempting to design a viable security implementation (Sheikh, Chande & Mishra, 2010).

The criticality of the above-mentioned security parameters enforces the need to find a balance between these parameters in the design and implementation of security solutions for MANETs (Yang et al., 2004). Thus, security researchers are left with an important task: finding a common ground between optimal security enforcement and the unique system requirements needed to support security in MANETs. Failing to regard such a balance may leave these solutions inefficient and could potentially cause more harm than good (Dorri et al., 2015).

A MANET is vulnerable to different types of attacks that could affect its various resources, or even more, to attacks that block the entire network operations. Having the above-mentioned goals in place is important to protect network resources from such attacks and misbehaviors originating from both internal and external entities (Djenouri et al., 2005).

Attacks on MANETs

Security in MANETs is essential for sustaining the basic functionality of the network. The unique characteristics of MANETs, such as the lack of fixed infrastructure, open wireless medium, cooperative algorithms, dynamic routing, and lack of a clear line of defense, have made these networks vulnerable to a variety of attacks (Rajakumar, Prasanna, & Pitchaikkannu, 2014). Attacks on MANET can be classified into passive and active.

In passive attacks, an attacker listens and attempts to obtain information from the communication traffic between nodes without disrupting the network operation. Attackers can gather information from the data exchanged among nodes. Stealing or tampering with sensitive data in this way might degrade the confidentiality of the entire network. Such confidential information may include the network topology, locations of nodes, or identities of critical nodes (Nadeem & Howarth, 2013).

In active attacks, an attacker tries to disrupt the network functionality through modifying, corrupting, or fabricating the exchanged information inside the network (Wu, Chen, Wu, & Cardei, 2007; Wang, Hu, Zhi, 2008). In active attacks, attackers aim to disturb the network operations by launching malicious activities, such as modifying, forging, or dropping data/control packets. These attacks can be as extreme as bringing the entire network down.

These types of attacks can be launched by either a single or multiple colluding attackers (Nadeem & Howarth, 2013).

Attacks on MANETs can be further divided into external and internal attacks. External attacks are performed by entities that don't belong to the network while internal ones are carried out through insider nodes in a MANET. Insider attacks are more damaging when compared to external ones. This is because insider nodes possess privileged access rights and have knowledge of valuable and sensitive information exchanged inside the network (Wu et al., 2007).

In this research, we classify attacks on MANETs based on the different layers of the Open Systems Interconnection (OSI) model they operate on, as shown in Table 1.

Layer	Attack(s)
Physical	Eavesdropping, jamming
Data Link	Traffic analysis and monitoring, attacks on MAC protocols, attacks on WEP
Network	Black hole, gray hole, rushing, blackmail, worm hole, routing table poisoning, routing table overflow, byzantine, flooding, sinkhole, Sybil
Transport	SYN flooding, session hijacking
Application	Malicious software programs, repudiation
Multi-Layer	DoS attacks, man-in-the-middle, impersonation

Table 1- Attacks on OSI Layers

Physical Layer Attacks

- Eavesdropping: communications in wireless networks can be intercepted if an eavesdropper tunes to the proper frequency (Rajakumar et al., 2014). Therefore, due to the open wireless medium characteristic of MANETs, a packet that is transmitted between two nodes can be overheard by other nodes within the same radio range. An eavesdropping attack is hard to uncover since communicating nodes have no idea of its occurrence (Nadeem & Howarth, 2013).
- Jamming Attacks: signals can be corrupted by a malicious node with a stronger transmitter that can disrupt communications through overwhelming the target signals. This type of malicious activity is referred to as a jamming attack (Rai et al., 2010).

Data Link Layer Attacks

Connectivity between neighboring nodes is maintained by the data link layer protocols. Attackers may launch traffic analysis and monitoring, disruption of reservation-based wireless medium access control (MAC) protocols, or attacks against weaknesses in the wired equivalent privacy (WEP) protocol (Kannammal & Roy, 2016). The following summarizes the different attack types against this layer:

- Traffic analysis and monitoring: attackers intercept and examine exchanged messages within the network as an attempt to uncover information through analyzing communication patterns, amounts of data exchanged, and transmission attributes (Rajakumar et al., 2014; Nadeem & Howarth, 2013). This type of attack can uncover critical information, such as the location of commanding nodes in military operations. Even if encryption is enforced throughout the network, these attacks can still extract useful information through analyzing communication patterns (Nadeem & Howarth, 2013).

- Attacks on MAC protocols: MAC protocols have the responsibility of coordinating transmissions in a shared wireless medium. These protocols assume cooperation from all nodes in a MANET. Therefore, an attacker can ignore these protocols in an attempt to prevent others from sharing wireless channel access and disrupt communication between nodes in the network (Wu et al., 2007).
- Attacks on WEP: radio signals are encrypted at the data link layer level through the WEP protocol. This protocol is known for its weaknesses, such as lack of key management and other found vulnerabilities in the cryptographic algorithms used. Malicious nodes can exploit these weaknesses to gain access to information communicated by neighbors (Wu et al., 2007).

Network Layer Attacks

Routing is the most significant operation carried out in the network layer. Both reactive and proactive routing protocols for MANETs assume nodes' cooperation in the discovery of optimal routes. This assumption can be exploited as a vulnerability by malicious nodes to launch different kinds of attacks (Nadeem & Howarth, 2013), including the following types:

- Black hole attack: a malicious node advertises itself as having optimal routes to one or more destinations, attracting all routes to these nodes (Tamilselvan & Sankaranarayanan, 2007). However, when such malicious nodes receive packets that are targeted to such destinations, it drops and never forwards them. This causes the creation of a "black hole." The severity of the attack increases as the attacker becomes part of more routes in the network (Nadeem & Howarth, 2013).

- Gray hole attack: this attack is a variation of the black hole method. The attacker node initiates this attack by advertising itself as having optimal routes to a certain destination. After that, it starts to selectively drop or forward packets (Sen, Chandra, Harihara, Reddy, & Balamuralidhar, 2007). This can sometimes depend on the source/destination of the packet. In some cases, the attacker might drop packets for a certain period and then resume forwarding (Abdelaziz et al., 2013). This type of attack is more difficult to detect than a black hole because of its selective nature (Sen et al., 2007).
- Rushing attack: on-demand routing protocols in MANETs limit the overhead of route discovery packets by having each node forward the first route request received for each discovery (Hu, Perrig, & Johnson, 2003). Rushing attacks exploit such mechanisms as the malicious node rushes to flood the network with route requests. This increases the probability of including the attacker in more routes during future routes discoveries. It also causes the suppression of other legitimate routes in the network (Hu et al., 2003).
- Blackmail attack: in some routing protocols, nodes in a MANET maintain a blacklist of other malicious nodes. This type of attack targets such protocols by fabricating malicious-activities-reporting-messages from the attacker in an attempt to corrupt the reputation of legitimate nodes. Such messages may result in isolating the reported nodes from the network (Wang et al., 2008).

- Worm hole attack: in this type of attack, the attacker records packets at one location in the network and tunnels them to another attacker residing in a different location. This tunnel consists of a shared private communication link between the two attackers, referred to as a "wormhole" (Wang et al., 2008). The tunnel's link usually has a faster data transmission rate than the rate between legitimate nodes, causing the attackers to be included in more routes (Abdelaziz et al., 2013). This attack may result in distorting the network topology by preventing the discovery of alternative routes, other than the worm hole (Wu et al., 2007).
- Routing Table Poisoning attack: routing protocols in MANETs maintain one or more table(s) to store routing information. Malicious nodes can fabricate and send fake packets or modify legitimate ones to create false entries in the routing tables of other nodes. This type of attack is referred to as routing table poisoning (Wang et al., 2008).
- Routing Table Overflow Attack: a malicious node may attempt to overflow the routing tables of others by forging packets to non-existent destinations. After that, the attacker floods the network with these forged packets through excessive route advertisements, resulting in flooding the routing tables of other nodes. This disrupts network routing by leaving victim nodes incapable of creating new entries in their routing tables (Abdelaziz et al., 2013).
- Byzantine attack: in this type of attack, an attacker creates/modifies routing control packets with false information in an attempt to disrupt routing operations in the network. This type of attack can result in routing loops, non-optimal routes, and packet dropping (Abdelaziz et al., 2013).

- Flooding attack: this type of attack occurs when a malicious node attempts to drain the battery resources of others by requesting excessive route discoveries or by forwarding irrelevant packets to other nodes (Wu et al., 2007). Since nodes in MANETs usually have limited battery power, such attacks result in having the energy-drained nodes incapable of participating in the routing process. This renders such nodes unreachable by others in the network (Sarkar & Roy, 2011).
- Sinkhole attack: a malicious node attempts to attract all traffic to itself by advertising false routing information to its neighbors. This allows that node to be included in more routes to destination nodes. When the sinkhole node receives the data packets, it drops or modifies them silently (Gandhewar & Patel, 2012). This can boost nodes' energy consumption by increasing network overhead. As a result, this type of attacks can decrease network life and eventually eradicate the entire network (Gandhewar & Patel, 2012).

- Sybil attack: due to the lack of centralized identity management in MANETs, an attacker can create one or more fake identities for itself. Since every node must to have an IP address as its identity, an attacker can send control packets with different IP addresses (Nadeem & Howarth, 2013). This allows a malicious node to gain more access, information, and resources than that allocated for a single node in a network (Rajakumar et al., 2014).

Transport Layer Attacks

SYN flooding and Session hijacking can be launched against a MANET at the transport layer level.

- SYN Flooding: similar to wired networks, nodes in MANETs rely on the transmission control protocol (TCP) to perform their communications. TCP uses a handshake mechanism between nodes before the start of any communications (Wu et al., 2007). This mechanism can be exploited by SYN flooding attacks. In these attacks, a malicious node overwhelms the victim by creating a large number of half-open TCP connections. It is considered a type of denial of service (DoS) attack, which leaves the victim incapable of accepting new connections with other nodes.
- Session Hijacking: through exploiting the session establishment mechanism in TCP, an attacker can spoof a victim's IP address, continue the session with the target, and launch a DoS attack (Kannammal & Roy, 2016).

Application Layer Attacks

Malicious software programs and repudiations are the main types of application layer attacks.

- Attacks by malicious software programs: the application layer contains user data and supports a wide variety of protocols. Malicious programs operating on this layer, such as worms and viruses, can be launched against a MANET. Such programs can traverse the network, find, and infect nodes through probing and exploiting existing vulnerabilities, resulting in data corruption (Wu et al., 2007).
- Repudiation: this is another type of attack at this layer. In this scenario, a malicious node denies its full or partial participation in a communication operation with other nodes. Malicious nodes may launch such attacks to send false routing information, resulting in isolating legitimate nodes from the network (Yi, Naldurg, & Kravets, 2001).

Multi-Layer attacks

Attackers may launch different types of attacks from multiple layers, including the following types:

- Denial of Service (DoS) attacks: the main objective of this attack is to drop legitimate, authorized nodes out of the network. These attacks usually target draining nodes' resources and/or create a contention in the network that disrupts communications (Jawandhiya et al., 2010). DoS attacks can be launched from any layer in the network. At the physical layer, malicious nodes may employ jamming against the wireless communication channels in a MANET. At the data link layer, an attacker can occupy communication channels, preventing others from access. Packet modifications, dropping, and routing table overflow can be launched from the network layer. At the transport layer, attackers can employ SYN flooding. Malicious programs can perform DoS attacks from the application layer as well (Kannammal & Roy, 2016).
- Man-In-The-Middle attacks: a malicious node can position itself between two communicating nodes to sniff the traffic flowing between them. The attacker may also participate in the communication by impersonating the sender or the receiver. This type of malicious activity is known as a man-in-the-middle attack (Wu et al., 2007).
- Impersonation attacks: the lack of central network management in MANETs can lead to impersonation attacks. In these attacks, a malicious node impersonates another legitimate one by stealing its identity and using it to communicate with other legitimate nodes in the network (Rai et al., 2010).

The above-mentioned attacks target different operational and communicational aspects of MANETs, rendering the network incapable of performing its basic functions. Providing security for such networks is considered the major roadblock in the way of a wider adoption of MANET applications. As such, throughout the years, researchers have made the mission of providing security solutions that target different types of potential attacks their main objective in the area of MANET security (Nadeem & Howarth, 2013).

Security Strategies in MANETs

The implementation of security strategies is considered one of the highest priority design elements for any type of network architecture. Applying these strategies as an afterthought requires expensive efforts and can lead to breaches from malicious attacks before a solution is put in place. The unique characteristics of MANETs make achieving security a complicated process (Hubaux, Buttyán, & Capkun, 2001). The distributed nature of these networks does not allow the assumption that networked devices are always controlled by their legitimate owners. As such, any application of a security scheme for MANETs requires a device/node-level implementation (Papadimitratos & Haas, 2004). Throughout the years, researchers have created an extensive body of literature on the development of security strategies for MANETs and their applications. These strategies can be classified into two categories: prevention mechanisms and detection and reaction mechanisms.

Prevention Mechanisms

Preventive solutions for MANET security target the primary quest of deterring malicious attackers by hardening the system, which makes it notably difficult to penetrate (Yang et al., 2004). These solutions act as first lines of defense to minimize potential attacks (Zhang & Lee, 2005). Different prevention mechanisms have been proposed for MANETs in the current body of research, including device imprinting, key management schemes, secure routing, and cooperation enforcement techniques.

MANETs applications span different fields with a wide variety of implementations, such as military and disaster relief operations. Considering the sensitive data carried out through these operations and the vulnerable physical locations of nodes, devices in MANETs are prone to be physically captured by malicious entities. As such, these devices can be equipped with security checks that are capable of tracking their legitimate owners and deciding whom to trust (Hubaux et al., 2001). Sensitive data contained in these devices should also be protected by enforcing a variety of security models, such as biometrics and smart cards (Wu et al., 2007).

For a group of nodes in a MANET to securely communicate, an efficient group membership management scheme should exist. Such a scheme would be responsible for protecting the transmitted data against potential attacks. For this technique to be established, a secret key must be shared by all group members and used to encrypt/decrypt communications within the network. The possession of this key is considered a proof of trustworthiness (Hegland et al., 2006). Examples of key management services are symmetric cryptography, asymmetric cryptography, and group key management. The primary responsibility of these services is to ensure the secret key, targeted for legitimate users, is not shared with unauthorized entities. The main issues with these mechanisms are their susceptibility to the various resource and bandwidth constraints possessed by MANETs (Aziz & Nourdine, 2008).

Routing operations in MANETs are vulnerable to both external and internal attacks. An external attacker can disrupt operations through retransmission and inefficient routing, which might lead to network partitioning and extensive traffic load. Different strategies have been proposed to protect against external attacks on routing, such as the use of digital signatures during message transmission (Zhou & Haas, 1999). Internal attacks that are generated from compromised nodes, on the other hand, cannot be prevented through the usage of authentication and encryption solely. Preventing such attacks requires securing the underlying routing protocols by preventing malicious nodes from disrupting the routing process (Zhang & Lee, 2005).

An extensive body of knowledge has been dedicated to designing and evaluating different preventive security mechanisms for routing protocols for MANETs. The primary target of these protocols is to prevent malicious nodes from disrupting normal routing in the network. Such disruptions can be in the form of fabricating false routing messages, modifying original routing information, or impersonating other nodes (Chlamtac et al., 2003). Secure routing protocols typically build on existing protocols by adding security extensions, such as message authentication through cryptographic methods. This way, nodes can differentiate legitimate traffic against unauthenticated transmission packets originated from malicious entities (Yang et al., 2004).

Unlike the operations in fixed-infrastructure networks, routing, data forwarding, and network management are carried out by all available nodes in a MANET. This makes the enforcement of cooperation among nodes an essential requirement for a MANET to remain operational (Chlamtac et al., 2003). Every communication between two nodes that are more than one-hop away is carried out by intermediate nodes. This introduces concerns regarding malicious and selfish nodes. One or more of these intermediate nodes may not cooperate in forwarding the data and can potentially disrupt network operations. On the other hand, some nodes act selfishly by not forwarding packets to conserve their resources (Chlamtac et al., 2003). Such behaviors have introduced the need to design systems that encourage collaboration among nodes to keep routing and data forwarding tasks alive. The current body of knowledge in this area contains a plethora of methodologies for cooperation enforcement. These can mainly be divided into reputation-based and credit-based systems (Marias, Georgiadis, Flitzanis, & Mandalas, 2006).

Reputation-based models utilize different techniques to calculate a reputation score for each node. This score increases every time a node forwards a packet without alteration. Based on the calculated score, nodes with higher reputations are used to forward data packets (Mandalas, Flitzanis, Marias, & Georgiadis, 2005). On the other hand, in credit-based systems, data forwarding among nodes is considered a service that is valuated or charged. Since data forwarding incurs resource consumption on nodes, it’s treated as an incentive to persuade non-cooperative nodes to participate in the routing process. Credit-based systems employ a virtual currency mechanism that is usually implemented to reward nodes that forward packets and punish those which act selfishly and drop packets (Hu & Bemester, 2009).

Preventive security techniques can help defend against certain attacks on MANETs. However, such techniques are incapable of mitigating new attacks because they are usually designed for known ones (Wu et al., 2007). Besides, the application of these techniques is usually accompanied by a high processing overhead and energy consumption, which might prove inefficient considering the resource constrained nature of MANETs (Zhang & Lee, 2005). These limitations in preventive techniques introduce the need for a new line of defense that is capable of uncovering unknown threats as well as balancing security and resource consumption. This is achieved by employing detection and response mechanisms, namely intrusion detection systems (IDSs).

Detection and Reaction Mechanisms

Intrusions are malicious activities aiming to compromise the confidentiality, integrity, and availability of a network. Prevention mechanisms can be effective in reducing potential attacks. However, if a node is compromised, all secrets associated with it are prone to attacks. This, in turn, makes such mechanisms inefficient against malicious insiders that cause much greater damage than external attackers (Sun, Osborne, Xiao, & Guizani, 2007). Besides, the history of preventive mechanisms shows the impracticality of such systems to survive on their own in providing a secure, intrusion-free system (Yang et al., 2004). Therefore, the presence of detection and reaction techniques, namely IDSs, that are capable of uncovering intrusions and avoiding their adverse effects through reactions is imperative for the survivability of a network (Yang et al., 2004). IDSs consist of a set of automated components that are capable of detecting suspicious activities and reacting to them in an attempt to prevent the security of the network from getting compromised. The detection part of an IDS involves constant monitoring of network activities. On the other hand, the reaction part involves raising alarms as well as taking preventive measures, such as isolating the suspect from connecting to the network (Mishra, Nadkarni, & Patcha, 2004).

Origins of IDS

In 1972, while working with the United States Air Force (USAF), Anderson (1972) noticed that the increased reliance on computer systems to perform sensitive operations led to a similar increase of computer security problems. Ad hoc security rules cannot be relied on because their inherent design flaws make them an easy target for malicious attacks. Thus, securing a computer system calls for the implementation of authorization mechanisms, access control mechanisms, and security marking of electronic and physical resources. Anderson (1972) emphasized the need for security controls due to the lack of operating-system-level defense mechanisms against malicious attempts to gain unauthorized access.

Anderson (1980) introduced the idea of identifying abnormal user behaviors as threats to computer security and generating alerts through examining the information contained in audit files. Anderson defines a threat as the likelihood of a targeted unauthorized attempt to access/manipulate information, leaving a system unreliable. Anderson (1980) classified threats as coming from the following sources:

- External penetrators: a person with physical access to a computer system’s location but no authorization to use it.
- Internal penetrators: a person who is authorized to use a computer system but unauthorized to use certain resources. Internal penetrators are further sub-classified in the following categories:
    - Masqueraders who gain access to a system under different credentials,
    - Legitimate users that misuse their authorized access, and
    - Clandestine users who are able to evade access controls and audit trail recordings

Host-Based Intrusion Detection

Anderson's (1980) system operates based on statistical abnormality, which means that user activities are considered “abnormal” if they cross certain predefined limits. Abnormality in this system is purely parametric. This means any usage outside of the identified parameters is considered abnormal. These parameters are defined based on applied statistics on audit files to define what would be "abnormal." If such limits are crossed, the system auto-generates security alerts in the form of exception reports. The system addresses the identification of intrusion attacks from both internal and external malicious users (Anderson, 1980).

Building on Anderson's (1980) work, research began towards real-time IDSs. The susceptibility of computer systems to intrusions due to their existing hard-to-replace flaws, the difficulty of developing flaw-free systems, and users' continuous misuse of their privileges, motivated Denning (1987) to develop the first real-time IDS. It was the very first general-purpose framework for IDSs and has become a fundamental core of most intrusion detection methodologies in this area. Denning (1987) followed the first system-independent approach to identify intrusions when the vulnerability behind intrusions is unknown. Her system records normal user activities as signatures to compare against in the detection of abnormal behaviors. Normal usage of the system is monitored, deviations from normal usage are identified, and an alert is sent to the designated security officer when deviations are observed (Denning, 1987).

Following Denning's (1987) work, Smaha (1988) introduced Haystack, which is an IDS for multi-user Air Force computer systems. The system relied on processing audit trail files to produce short summaries of normal/abnormal behaviors and security incidents. An intrusion in this system is defined as any violation of the already-established security and administrative policies of a computer system. It utilizes a combination of both a multi-user model, which establishes normal behaviors of a group of users, and a single-user model, which is based on a user's past behavior. These two models are then used in this system to establish statistically-based models. One of the novelties of Haystack is the self-modifying behavior in which user models evolve to match the changes in users' work requirements.

Lunt and Jagannathan (1988) believed that implementing an IDS on a machine separate from the target machine would enhance security and performance. Their belief has led them to propose their own methodology for designing an IDS. Regarding the classifications of anomalous behaviors and defining such behaviors as deviations from the expected user behavior, their system follows the same approach as Denning's (1987). However, their new system extends Denning's (1987); in their system, an IDS sits on a separate machine within the network, receiving user activities through audit data over the network. This way, the IDS cannot be tampered with from would-be intruders, and any existing flaws on the target system cannot impact the security of the detection system.

Network-based Intrusion Detection

Because Anderson's (1980), Denning's (1987), and Smaha's (1988) IDSs operated on a single machine for detecting anomalies, audit files must exist on the same computer system as the IDS. Shortly after, due to the increased usage of networking devices that were suffering from scant or nonexistent security measures, Heberlein et al. (1990) came up with the first network-based IDS (NIDS) operating on Ethernet-based local area networks (LAN). The system was called: Network Security Monitor (NSM). Although encryption techniques presented an equally appealing option to prevent network attacks, Heberlein et al. (1990) believed they couldn't protect against privilege misuse by legitimate users. Incorporating ideas on stand-alone IDSs from Denning (1987), Whitehurst (1987), and Lunt et al. (1988), the new model builds a normal activity profile of the entire network and monitors activities against that profile in real time.

Computer systems come with inherent security vulnerabilities that make the mission of having an intrusion-free system, or network of systems, an extremely difficult task. Even if a system is thought to be the most secure one due to the applied harnesses, it is still susceptible to insider threats through privilege misuse (Lunt, 1993). All of the above-listed research on IDSs agree on one common conception: attacks against computer security primarily come from the inside. Traditional access controls can help provide certain levels of defense against threats. However, in most cases, they are incapable of guarding a system against insider threats (Lunt, 1993). As network-based security attacks became more widespread and sophisticated, the focus of IDSs has shifted towards the exploration of mechanisms against such attacks, which influenced the direction of the research that followed (Vigna & Kemmerer, 1998).

Intrusion Detection in MANETs

A plethora of IDS solutions populate the current body of research targeting traditional fixed networks. Such networks typically possess central traffic points where all network packets must pass through. Therefore, installing an IDS on one of these central points gives it a wide view on the network activities. This is not the case for MANETs as no central traffic points exist. Besides, the constant topological changes in these networks due to nodes' mobility makes intrusion detection process even harder (Yang et al., 2004). This is mainly due to potential false routing information that is sourced from stale routing tables caused by nodes' volatile mobility. Additionally, communications among nodes in a MANET are carried over an open-access wireless medium that both legitimate and malicious nodes have access to. This, in turn, leaves no explicit differentiation between legal and illegal activities for IDS solutions (Anantvalee & Wu, 2007).

Since IDS implementations in MANETs do not have the ability of collecting audit data from fixed traffic points, audit data sources pose as a limitation for these systems. In fact, the only available audit trace for IDSs is communication activities within the radio range of the IDS node. The main concept behind intrusion detection is the examination of audit data to determine whether the system is under attack. This leaves researchers who design intrusion detection algorithms with no choice but to capitalize and devise new methodologies to make use of the limited localized information (Zhang, Lee, & Huang, 2003).

For an intrusion detection implementation to work, researchers must make certain necessary assumptions. The first assumption is that nodes' activities in the network are observable. Another critical assumption is that normal and malicious activities possess clear separable behavior. However, such assumptions are typically faced with the inherent limitations in MANETs, making the task of implementing an efficient IDS a difficult challenge. This, in turn, leaves researchers working on intrusion detection implementations with an increasingly difficult task of distinguishing between false alarms from real attacks against the network (Zhang et al., 2003).

IDS solutions in general require continuous monitoring, collection, and processing of audit data to detect possible intrusions. This level of effort required to operate an IDS is sometimes deemed too costly to implement due to the bandwidth and resource-constrained nature of nodes in MANETs. Such constraints demand careful design considerations from researchers to establish a balance between resource consumption and the level of provided security. This has also made collaborative cooperation between nodes in the IDS activities, a critical requirement to distribute the load among nodes while providing a wide view on the network (Karygiannis, Antonakakis & Apostolopoulos, 2006).

For researchers who attempt an IDS implementation, certain design requirements must be considered for the proposed system to be viable for MANETs. First, researchers should determine the suitability of their architecture for MANET applications and decide whether a proposed IDS is the right fit for the unique features in these networks. Secondly, researchers must define the appropriate sources for collecting audit data, and they must determine the right methodology to detect intrusions when only partial audit sources are available. Lastly, those researching IDS implementations should define an activity model for the target MANET that can be used by the system to distinguish intrusions from normal behavior when the network is undergoing an attack (Zhang et al, 2003).

Along with these considerations, Butun et al. (2014) have proposed that researchers should integrate the following requirements when designing an IDS for MANETs:

- An IDS implementation for MANETs must not introduce new weaknesses to the nodes' operating system.
- An IDS design should be tailored to enable the system to operate in the resource-constrained nodes in MANETs without degrading their performance or exhausting available resources.
- An IDS should be able to run continuously and transparently to the users.
- An IDS should be cooperative and reliable to minimize false negatives and false positives in the detection process.

Studies targeting the design and implementation of IDS solutions for MANET must be able to accommodate the unique requirements imposed by these networks. Besides, traditional solutions for fixed networks cannot be directly applied to MANETs due to their fundamental differences. The current body of research contains various IDS implementations for MANETs employing different techniques to accommodate the above-mentioned design requirements.

Abiding by these requirements means that any proposed IDS implementation should detect a substantial percentage of malicious behaviors, keep a low rate of false alarms, and keep resource consumption minimal (Mishra et al., 2004).

IDS Mechanisms in MANETs

There exists a large body of research pertaining to different solutions for intrusion detection in wired networks. However, the lack of a fixed infrastructure, disconnected patterns of communications, and lack of central traffic concentration points make the application of wired-network IDS solutions infeasible for MANETs (Zhang & Lee, 2000). Researchers have proposed different detection mechanisms to adapt to the unique characteristics of MANETs. These mechanisms can be classified into three categories based on their detection techniques: anomaly-based, knowledge-based, and specification-based intrusion detection (Mishra et al., 2004).

Anomaly-based Intrusion Detection

Since Denning's work back in 1987, a plethora of research has been dedicated to developing various implementations of anomaly-based intrusion detection (ABID). Due to the perceived power of such systems in detecting known as well as unknown attacks in the network, previous research studies have favored this type of IDS over the others. This is primarily because attacker creativity continues to present a major challenge for intrusion detection implementations (Cannady, 2009). Therefore, the ability of ABIDs to address novel attacks continues to make this type of IDS a popular area of academic research (Tavallaee, Stakhanova, & Ghorbani, 2010).

Despite the presence of an extensive body of research on various methodologies for ABIDs, a clear definition of what constitutes an anomaly is still lacking from many studies (Tavallaee et al., 2010). However, the common understanding of anomaly in the research community leans towards defining an anomaly as an "abnormal behavior" in the network. For instance, a survey conducted by Tavallaee et al. (2010) has shown that the majority of research studies in the area of ABIDs tend to describe anomalous behavior as a behavior deviating from normal. Additionally, the seminal work of Denning's (1987) on ABIDs has treated anomalies as activities that deviate significantly from the established normal network profile.

Previous studies tend to establish normal network profiles, in this type of intrusion detection, through statistical behavioral modeling. This means that normal network operations of nodes in a MANET are outlined so that any deviation of these profiles is considered as an anomaly. For ABIDs to maintain high accuracy throughout their operations in the network, they typically are bound to perform periodic updates to their established normal profiles. This is due to the dynamic topological nature of MANETs, which causes normal network behavior to change rapidly. Since nodes in MANETs are resource-constrained in nature, such frequent updates might impose extra overhead that could cause disruptions in the normal network operations (Butun, Morgera, & Sankar, 2014).

In an ABID, also known as a behavior-based IDS, intrusions are detected through a comparison between expected normal behavior and the current behavior of the network. This type of IDS has two phases: training and testing. In training, baseline profiles are created to model the normal behavior of the network. In the testing phase, the constructed profiles are used to detect any significant deviations from the current behavior, which are considered intrusions.

Different methodologies are typically applied for calculating these deviations (Nadeem & Howarth, 2013).

For ABIDs to gather the information required for establishing a network's normalcy profile, extensive data-mining operations are deemed critical during the training phase. The implementations of ABIDs in the current body of knowledge present different techniques for collecting training data (Mitchell & Chen, 2014). Some of these implementations train their systems with predefined sets of data, also known as "truth data". Others opt for training their systems against live network traffic. Investing time in this training phase before starting the actual detection operations of ABIDs can be extremely beneficial when it comes to discovering unknown attacks. Additionally, training enables this kind of intrusion detection to eliminate the extra storage space needed to save all attack vectors. This is because ABIDs do not look for something specific when it comes to attack detection. Instead, deviations from normalcy trigger these systems to determine intrusions in the network (Mitchell & Chen, 2014).

All of the training techniques devised in the current literature share one common goal: the process of feature extraction. Features in ABIDs are defined as security-related measures that are needed to construct an effective intrusion detection algorithm. For these features to be effective, they need to reflect the target subject (e.g., a MANET node) activities. Typical examples of features include number of sent/received packets, number of routing-related messages, and mobility patterns (Cai, Ci, Guizani & Al-Fuqaha, 2006).

According to the processing model of the target system's behavior, anomaly-based detection approaches can be categorized into three groups: statistical-based, knowledge-based, and machine learning techniques (Kheyri & Karami, 2012). In statistical-based techniques, an ABID monitors network traffic activities to create the normal network profile. A normalcy threshold is set during the training phase, and when the threshold is exceeded, an activity is considered an anomaly. When the system transitions from training to detection mode, the ABID compares the current observed activities against the normal profile to check if the deviation exceeds the normalcy threshold. If so, the system marks the node(s) originating the detected activity as malicious. The accuracy of detection for statistical-based anomaly detection tends to increase when longer durations are dedicated to the training phase. On the other hand, attacks might pass unnoticed in these systems due to their susceptibility of high false positives. This is usually correlated to the creativity of attackers who vary their approaches for launching different attacks and fool the system into recognizing these attacks as normal traffic (Kheyri & Karami, 2012).

Alternatively, knowledge-based techniques for ABIDs rely on a set of specific rules, usually defined by a human expert. These rules are then applied to determine the legitimate behaviors of the system. During the training phase, specific sets of features are extracted from the gathered data. Based on these features, the system classifies the gathered data according to the application of the predefined rules. This technique offers significant flexibility and robustness in detecting illegal activities in the network. Nonetheless, developing a comprehensive set of rules for these systems is viewed as a difficult and often time-consuming task (Butun, Morgera, & Sankar, 2014).

Machine learning techniques in ABIDs employ an implicit or explicit model to characterize activities overserved in a MANET. This approach usually requires labeling the data gathered in in the detection process for the system to learn the behavior model of nodes in the network. The process of data labeling is considered a high-resource operation, demanding extra energy and bandwidth from the resource-constrained nodes in MANETs. Despite the resource-demanding nature of these systems, they have shown undeniable efficiency in detecting both known and unknown attack vectors in previous studies. Common techniques used in this category are Markov chains, fuzzy logic, neural networks, genetic algorithms, and Bayesian networks (Butun, Morgera, & Sankar, 2014).

In general, ABIDs have the downside of demanding extensive training for behavior modeling. This is problematic for MANETs and other resource-constrained mobile computing environments because each profile must be continuously updated, which increases the processing overhead imposed by recalculations of deviations (Mishra et al., 2004). The significance of ABIDs comes from their ability to uncover unforeseen attacks. However, the constant changes in the normal network behavior of MANETs due to their dynamic topology, render these systems prone to high rates of false positives (Nadeem & Howarth, 2013).

Knowledge-based Intrusion Detection

Knowledge-based intrusion detection (KBID), also known as misuse-, signature-, or rule-based intrusion detection, targets events that match predefined patterns of malicious behaviors. KBIDs encode and store known attack signatures and system vulnerabilities in a designated database, and they constantly monitor activities in the network and compare them against stored attack signatures. An intrusion is detected, and an alarm is raised if a match is found between current activities in the network against one or more signatures (Anjum, Subhadrabandhu, & Sarkar, 2003).

KBIDs typically implement detection techniques that are completely reliant on accumulated knowledge of known attack patterns. The system analyzes network traffic against such patterns to determine the occurrence of intrusions. KBIDs usually have lower false alarms rates when compared to ABIDs. However, human experts are required to formulate attack signatures for the IDS to use in the detection process. This, in turn, requires the system to maintain attack signatures through regular updates, which is usually a time-consuming task that is difficult to design properly (Debar, Dacier, & Wespi, 1999).

There are three primary approaches that researchers follow when implementing a KBID: expert systems, model-based reasoning, and state transition analysis. Jackson (1986) defined expert systems as computing systems that are capable of applying reasoning about a knowledge-rich domain towards problem solving and advice formulating. In expert systems, conditions required for an attack detection are coded into the system in “if-then” implication rules. When all conditions in the “if” part are satisfied, a KBID triggers the predefined actions in the “else” part. These typically include response actions for the detected intrusion and sometimes might trigger extra rules to confirm that the detected activity is indeed an intrusion. The main advantage of expert systems is their separation between problem solving and control reasoning. Nonetheless, these systems often require extensive development and maintenance to create and manage the attack signatures (Kumar & Spafford, 1994).

Model-based systems implement a database of known attacks scenarios. An attack scenario is defined as a sequence of events that collectively make up the attack. In model-based KBIDs, the system continues to monitor the network behavior and assumes that it might be experiencing attack scenarios at any moment. Thus, as the system collects audit data, it continuously analyzes it to verify or disprove the occurrence of an attack. An evidential reasoning calculus is typically built into these systems to update the likelihood of specific attack scenarios as audit data is analyzed. Model-based systems can potentially decrease the extensive data processing incurred on nodes by actively monitoring successfully identified attacks while passively monitoring attacks that are yet to be encountered (Garvey & Lunt, 1991).

State transition analysis KBIDs use a sequence of state transitions for the monitored system to represent an attack occurrence. An attack is defined here as a sequence of events that might transition the system to a compromised state. The system identifies the events that occur between the system transitioning from normal to compromised as “signature actions”. That is, if these actions are eliminated from an attack scenario, the attack would not be able to complete successfully. The system detects intrusions by extracting state transitions from the collected data and comparing them against those of known attack scenarios stored in the knowledge database. If a match is found, the system marks the activity as an intrusion and triggers any predefined response steps (Ilgun, Kemmerer, & Porras, 1995).

In KBIDs, a carefully written attack signature can detect even major variations of the same attack. This is a major advantage for KBIDs versus ABIDs because attackers’ creativity and careful monitoring towards the behavior of an IDS in classifying attacks might enable malicious activities to pass unnoticed by an ABID. However, the reliance on attack signatures renders KBIDs incapable of uncovering new types of attacks (Sun et al., 2007). KBIDs resemble anti-virus programs because they effectively detect most of the known attack patterns but are inefficient at uncovering new ones (Butun et al., 2014).

Specification-based Intrusion Detection

Specification-based intrusion detection systems (SBIDs) combine the advantages of both ABIDs and KBIDs through the use of manually constructed constraints to model the normal system behavior. However, they are more similar to ABIDs because ABIDs and SBIDs both define intrusions as deviations from the characterized normal behavior. SBIDs define a set of constraints as specifications that establish the correct operation of a program or protocol. The execution of these programs/protocols is continuously monitored with regard to the defined specifications. Deviations from the specifications are considered intrusions (Butun et al., 2014).

The reliance of SBIDs on security specifications for the correct system behavior allows them to achieve tremendously higher accuracy over both KBIDs and ABIDs in detecting both known and unknown attacks scenarios. Since all specifications are based on legitimate behaviors, the system does not generate false alarms when encountering unusual, though legitimate, behaviors from the target system (Uppuluri & Sekar, 2001). The original concept of specification-based detection was first introduced by Ko, Ruschitzka and Levitt (1997) when they described the desirable behavior of the system through its functionalities along with the application of security policies. Accordingly, Ko et al. (1997) stated that any sequence of actions executed outside of the system's specifications should be considered as a violation.

For SBIDs to detect malicious behaviors, they typically employ execution analyzers. These, in turn, monitor the execution of an activity/program with respect to the predefined specifications. Each specification has a dedicated analyzer that only checks for violations against that specification. The system raises an alarm whenever it detects missing/unknown events in the monitored execution of the target program (Ko et al., 1997). Based on such a design, SBIDs in general don't detect the actual intrusions. Instead, they detect the impact of intrusions on program executions causing a violation in the specified behavior (Tseng et al, 2003).

Considering that such detection systems can achieve higher accuracy than KBIDs and ABIDs, the adoption of SBIDs is still not as wide as it is for the others. The main hindrance against the adoption of these systems is the inherent limitations involved in defining specifications for every operation in the target system. Thus, in MANETs, the focus of SBIDs solutions in the current literature so far has been put towards routing protocols (Tseng et al, 2003; Tseng et al., 2005). One limitation of current implementations of SBIDS against routing protocols for MANETs is their tendency to constrain the messages exchanged among nodes in the network, such as restricting receipt acknowledgements and message contents. Another potential obstacle hindering SBIDs adoption is the difficulty of verifying the correctness of the developed specifications and that the specifications comprehensively cover the threat model (Berthier & Sanders, 2011).

SBIDs can be loaded into mobile nodes prior to deploying a network. This is possible because SBIDs do not require periodical updates for attack discoveries. This mechanism is known for having low rates of false positives along with the ability to uncover unknown attacks. However, extensive manual work is usually required to define the needed constraints (Brutch & Ko, 2003).

IDS Architectures in MANETs

The concept of a “one network configuration works for all” does not apply to MANETs. Based on the deployment field and operational requirements, MANETs can be configured in a flat or multi-layered infrastructure. In a flat configuration, all nodes are considered equal and free to take part in the routing process. On the other hand, in a multi-layered configuration, all nodes are not equal. In a multi-layered setting, nodes within each other's transmission range are divided into clusters. Each cluster will have one or more nodes that are designated as gateways providing connectivity with other clusters in the network (Brutch & Ko, 2003). Taking into consideration these different configurations, an effective IDS implementation should accommodate the target network infrastructure. Extensive research has been done in this area, resulting in proposing different IDS architectures in efforts to satisfy the needs of the various network configurations in MANETs. Examining previous research in this area, IDS architectures are primarily categorized into stand-alone, hierarchical, and cooperative architecture (Anantvalee & Wu, 2007).

Stand-alone

This architecture targets flat network configurations in MANETs. In this type of IDS, each node runs its own intrusion detection engine independently from others. Intrusion determination decisions are made solely based on the information gathered by that individual node. Nodes do not cooperate with each other regarding information sharing. As such, if a node detects an intrusion, others would not know about it. This architecture is more suitable for situations where not all nodes are capable of installing/running an IDS (Brutch & Ko, 2003).

Different techniques have been proposed in the current body of research as part of the efforts to provide efficient self-contained IDS solutions for MANETs. Bensal and Baker (2003) proposed their observation-based cooperation enforcement in ad hoc networks (OCEAN) targeting the DSR protocol. Their system relies solely on nodes' own observations to detect intrusions. The system classifies nodes as selfish and misleading. If a node observes its neighbor as not participating in the route discovery process, it marks that neighbor as "selfish". On the other hand, if a node observes a neighbor participating in the route discovery process but fails to forward packets, that neighbor is marked as "misleading". To detect misleading nodes, a watchdog-like mechanism is implemented on each node to monitor whether its own neighbor forwards or drops packets. This is done by having each node calculate and save a checksum for each packet it forwards to its neighbors. The node then listens to the forwarded packet and compares it against the saved checksum. If a match is found, the monitoring node increments the positive rating for that neighbor. Otherwise, the packet is considered dropped, and the monitoring node increments a negative rating for that neighbor. The system predefines a threshold for a neighbor’s rating. If a neighbor’s rating falls below that threshold, it is avoided in future routes.

Nadkarni and Mishra (2004) presented a threshold-based stand-alone IDS. Their system establishes threshold values for well-known attack patterns. During the network initialization phase, attack thresholds are set based on comparing the normal network profile against the average frequency of events for each known attack type. Intrusions are then detected individually by nodes when a predefined threshold is crossed for a specific attack. Jacoby and Davis (2007) introduced their system that relied on battery-power consumption to detect intrusions. The proposed system constantly monitors the battery consumption of every node in the network. Intrusions are detected by comparing a node’s battery consumption against a set of battery-consumption patterns of well-known attacks. Their system relies fundamentally on Smart Battery technology in which nodes’ batteries are equipped with an internal circuit that enables communication of battery states to a node’s operation system.

In an attempt to solve receiver collision problems and limited transmission power issues among communicating nodes in MANETs, Liu, Deng, Varshney, and Balakrishnan (2007) proposed their two-hop acknowledgment (TWOACK) system for detecting packet dropping attacks. When a source node sends a packet to a destination, the system requires an acknowledgement for every packet transmitted over every three consecutive nodes along the route. That is, when an intermediate node receives a packet, it must send an acknowledgement to every node that is two hops away from it down the route. This acknowledgement packet contains the reverse route from the intermediate node to the source. A node is declared as malicious when it fails to send back that acknowledgement.

To reduce the overhead incurred in TWOACK through unnecessary acknowledgements from intermediate nodes, Sheltami, Al-Roubaiey, Shakshuki, and Mahmoud (2009) presented their adaptive acknowledgment (AACK) IDS. In this system, when two nodes are communicating, the destination node must send an acknowledgment of packet receipt to the sending node within a predefined period. Thus, replacing required acknowledgements from intermediate nodes with acknowledgements from destination nodes in a communication. This system reduces routing overhead in comparison to TWOACK when the route to a destination consists of more than two hops.

Lauf, Peters, and Robinson (2010) came up with a stand-alone IDS that has two intrusion detection engines. The first engine detects observed anomalies and then passes them down to the second engine to check against the predefined threshold. The system sets such a threshold by establishing a network normalcy profile through maintaining a history of interactions occurred at the application layer. The threshold is then defined as the average application layer behavior of all nodes in the network. As such, intrusions are detected when a node behavior exceeds the identified threshold.

Joseph, Lee, Das and Seet (2011) came up with their stand-alone cross-layer IDS targeting malicious sinking behavior in MANETs. Joseph et al. (2011) define sinking as a malicious behavior exhibited from certain nodes that don’t cooperate in the routing process and, instead, tend to drop data and routing packets. The system collects audit data from physical, MAC, and network layers in efforts to maximize detection accuracy. Their implementation includes a mixture of anomaly-based and signature-based detection. The system relies on machine learning through the application of a support vector machine (SVM) for classification of audit data. However, SVM algorithms are known to impose resource-exhaustive operations on nodes. Thus, their application of SVMs is targeted towards the training phase. Each node in the system monitors its own neighbors through activating promiscuous listening against neighbors' communications. The captured data is then fed into the detection engine to detect any deviations based on a comparison with attack thresholds calculated in the training phase.

Both ACK and TWOACK systems suffer from one major vulnerability: the acknowledgement receipts can be forged by malicious nodes rendering the system incapable of detecting packet dropping attacks. Thus, Shakshuki, Kang and Sheltami (2013) presented their approach to address this vulnerability through the implementation of a new IDS. This approach is called Enhanced Adaptive Acknowledgement (EAACK). The idea behind their system is the usage of acknowledgement packets for communications among nodes in a MANET. Whenever a node sends a packet to a destination, it waits for a receipt acknowledgement packet from that destination. If the originating node does not receive the acknowledgement packet within a predefined period, it sends an acknowledgment-request packet to intermediate nodes in the route to the destination. Misbehaving nodes in the routes are detected when one of them does not return an acknowledgement packet stating that it has received the acknowledgment request. All acknowledgements are digitally signed to eliminate packet forging.

Digital signatures represent the most valuable innovation behind the EAACK system since it is proposed to overcome packet forging issues found in both ACK and TWOACK. Thus, all acknowledgement packets must be signed, so they can be verified by the receiver. In their evaluation of packet signing, Shakshuki et al. (2013) proposed that the application of the Digital Signature Algorithm (DSA) produced desirable results in terms of signing/verification speed.

However, utilization of digital signatures imposes extra computational overhead, resulting in high consumption of battery power on the resource-constrained nodes in a MANET.

Motivated to overcome the difficulty of detecting multiple attackers launching different attacks in a network, Mapanga et al. (2017) proposed their stand-alone IDS. Their system relies on machine learning through the application of multilayer perception neural network (MLP-NN) for classification of packet dropping attacks. The system undergoes a training phase that is based on locally collected data from each node, focusing on nodes participating in the route discovery process of the AODV protocol. This data is then passed to the MLP-NN engine to extract routing parameters from control messages exchanged among the nodes to establish the normalcy profile. Their approach assumes that malicious nodes have built-in attack mechanisms to drop all data packets while responding successfully to routing messages.

Despite the various improvements proposed for stand-alone intrusion detection, it still possesses limited efficiency when compared to other types of architectures (Brutch & Ko, 2003). This limitation exists because of the reliance of stand-alone architecture solely on locally collected data, which impacts detection accuracy and limits the range of malicious attacks that can be uncovered (Şen & Clark, 2009). Additionally, this type of architecture does not provide visibility for the security of other nodes in the network, which indicates its inability to detect coordinated attacks (Farhan, Zulkhairi, & Hatim, 2008).

Hierarchical

A hierarchical IDS is more suitable for multi-layered configurations in MANETs. In this architecture, each node has its own IDS to detect intrusions locally. However, a MANET is divided into groups, called clusters. Each cluster has a "cluster-head" that acts as a gateway connecting a cluster with others. The collection of cluster-heads in the network provides the connectivity backbone needed for all clusters to be able to talk to each other (Anantvalee & Wu, 2007). Cluster-heads are also responsible for monitoring nodes' activities in their own clusters, and they are responsible for participating in global intrusion detection and response activities (Butun et al., 2014).

Researchers have employed different techniques for implementing hierarchical IDSs. Cluster-head nodes in this architecture are supposed to carry on the extensive intrusion detection responsibilities in the network. This, in turn, presents two major issues: resource consumption and trustworthiness. As cluster-heads carry out both local and global intrusion detection, they need to have more resources to be able to efficiently perform such activities (Butun et al., 2014). At the same time, these nodes act as gateways in their network. This mandates the need for cluster-heads to be trustworthy; otherwise, network operations would be disrupted if a malicious node gets elected as a cluster-head (Huang & Lee, 2003).

Previous studies attempted to tackle the resource consumption problem of cluster-heads by introducing various techniques. Cabera, Gutierrez, and Mehra (2005) presented their distributed IDS through network clustering. Each cluster would designate a cluster-head through electing the node which first responds to the election broadcast messages in the network initialization phase. Cluster-heads perform IDS operations through the installed anomaly-based detection engine. The engine employs cross-feature analysis (CFA) to calculate the normalcy profile for each node. The network must go through an extensive training phase to extract anomaly features before the IDS engines can be installed on cluster-heads. The system triggers the cluster-head election process periodically to eliminate draining resources of cluster-head nodes.

In an attempt to increase network lifetime with their approach, Kim, Kim, and Kim (2006) proposed their lifetime-enhancing selection (LES) scheme for selecting cluster-heads. In each cluster, the system chooses the node with the highest battery power as the cluster-head. This is done by having nodes periodically exchange their battery power through control messages. During these periods, nodes in the cluster vote to elect the node with the highest battery power as the cluster-head. When the batter power of the elected cluster-head falls below a predefined threshold, it initiates a re-election process in the cluster. To further enhance energy consumption, the system specifies a limited size queue of packets to be inspected at once by the cluster-head. Once the queue is full, no more packets are inspected until the current queue has been fully inspected. This, in turn, is proposed to prevent overwhelming cluster-head nodes with processing overhead by limiting the number of packets inspected at a time. The system is intended to enhance network lifetime by balancing the energy consumed by IDS activities among nodes in a cluster.

Otrok, Mohammed, Wang, Debbabi, and Bhattacharya (2007) followed a similar approach for cluster-head election by having each node report its battery power to its neighbors. The system implements a reputation scheme to motivate selfish nodes to cooperate. Based on voting, the node with the highest battery power would be elected as the cluster-head. As for the trustworthiness issue for cluster-head nodes, the system introduces the concept of "checker nodes". The system randomly selects nodes in each cluster to serve as checkers. The responsibility of checker nodes is to monitor the behavior of the cluster-heads and initiate a re-election process if the currently elected cluster-head starts misbehaving. The IDS defines a timer that expires when a new cluster-head election is triggered.

Pahlevanzadeh and Samsudin (2007) presented their agent-based hierarchical IDS for MANETs. Their system divides the network into clusters and defines cluster-heads as the nodes that have the highest battery power, processing power, bandwidth, and number of connections to other nodes in the neighborhood. At every cluster node, a data collection agent is installed to collect audit data about nodes in the neighborhood. This information is then passed to the cluster-head who has the intrusion detection agent (IDA) installed. The IDA uses an anomaly-based detection engine to make decisions about the collected data. However, Pahlevanzadeh and Samsudin (2007) did not describe the technique used for establishing normal network profiles. The IDA sends alarms to nodes in the cluster as well as to other cluster-heads to keep the network informed of the identified attacker.

Ma and Fang (2009) followed a similar idea to that of Kim et al. (2006) in their hierarchical IDS proposal. During the network initialization phase, each node reports its battery power to its neighbors. After that, nodes vote to elect the one with the highest battery power as the cluster-head. However, the election process doesn’t happen periodically as in Kim et al. (2006). Instead, a re-election is triggered whenever a new node joins the network, the elected cluster-head leaves the network, or the battery power of the cluster-head falls below the predefined threshold. The proposed system installs a local detection engine on cluster nodes and a global one on cluster-heads. However, it is unclear what kind of detection techniques the system employs as Ma and Fang (2009) did not explain it.

Roy, Chaki, and Chaki (2009) came up with the idea of “guard nodes” in their hierarchical IDS. The system was specifically designed to detect worm hole attacks, and it divides the network into clusters with a cluster-head responsible for the intrusion detection decisions of each cluster. However, the novelty of their approach is in the addition of guard nodes in each cluster. These nodes have the ability to monitor any node in the cluster and report back to the cluster-head if it detects any suspicious activities. This ability comes from positioning guard nodes in a cluster where they can sniff all neighborhood traffic by activating their promiscuous mode.

Abdel-Fattah, Dahalin, and Jusoh (2010) proposed their region-based hierarchical IDS. In their system, the network is divided into non-overlapping regions. Each region would have member nodes and gateway nodes. A node is considered a “gateway” if it has a connection to the neighboring region. Otherwise, it is deemed a member node. All nodes have an IDS installed on them. However, if member nodes detect intrusion, they must send their decision to gateway nodes to start global intrusion detection. The IDS in this architecture employs a combination of signature-based and anomaly-based engines. The anomaly-based engine utilizes the conformal prediction for k-nearest neighbor (CP-KNN) and distance-based outlier detection (DODO) algorithms to calculate the certainty of an attack existence in the collected data. If an anomaly is detected with a high certainty, the extracted intrusion pattern is passed down to the signature-based engine to extract and store the attack signature in its database.

Believing that a hierarchical architecture supports network scalability and fault tolerance, Shao, Lin, and Lee (2010) presented their hierarchical IDS for MANETs. However, their cluster-head election does not rely on battery power. Instead, the node with the highest number of connections to other nodes is elected as the cluster-head in every cluster. Nodes in each cluster declare the number of their one-hop neighbors. The system then employs voting among such nodes to choose the cluster-head. The presented IDS implements an anomaly-based engine to detect intrusions. This engine implements a back-propagation network (BPN), which is a supervised learning neural algorithm to train against and establish the normal profile of the network.

Zeng, Chen, Qiao, and Xu (2011) believed that since MANETs are energy-constrained by nature, nodes might be inclined to lie about their battery power during cluster-head elections. Zeng et al. (2011) stated that previous cluster-head selection approaches suffer from this issue, which might affect network lifetime negatively. Their approach focused solely on cluster-head election without detailing what kind of intrusion detection engine is used. Thus, they proposed their system in which selfish nodes are encouraged to reveal their battery power through incentives called reputation points. These points are important since the system uses them to distribute services provided by cluster-heads. Thus, nodes that don't reveal their battery energy wouldn't have enough reputation points to receive such services.

Due to the significance of the role played by cluster-heads in hierarchical IDS implementations, Katal, Wazid, Sachan, Singh, and Goudar (2013) believed that battery power alone is not sufficient for cluster-head election. Katal et al. (2013) attempted to address the challenge of cluster-head trustworthiness by adding more parameters to the election of cluster-heads. These include communication range, hop count, battery power, and mobility. Katal et al. (2013) also introduced the idea of "super cluster-head" formations, which are randomly-selected nodes with the responsibility of monitoring the elected cluster-head for malicious behavior. Excluding the cluster-head, super cluster-head nodes typically possess the highest battery power in their cluster. The proposed approach is intended to increase network lifetime as it keeps energy dissipation at low rates.

Amouri, Jaimes, Manthena, Morgera, and Vergara-Laurens (2015) proposed a hierarchical IDS that specifically targets blackhole attacks. In their system, the network is divided into a predefined number of squares. Each square has an imaginary circle representing the area of IDS activation. Whenever a node enters that circle, it becomes a pseudo cluster head (PCH). PCH nodes activate their promiscuous mode to sniff packets transmitted within their radio range. These PHCs then use the collected data to calculate anomalies in each cluster. This information, in turn, is passed to a manager node that is assigned at network initialization to act as the command and control unit in the network. The manager node makes the final intrusion decision at the network level. PCHs change as nodes in MANETs are in continuous motion. However, a node that is assigned the manager role remains in that position for the lifetime of the network. The manager node then uses the data collected from all PCHs to calculate network-level intrusions. The approach utilizes a simple C4.5 decision tree to learn typical network behavioral responses to black hole attacks during the training phase.

Specifically targeting the military applications of MANETs, Theresa and Sakthivel (2017) presented their fuzzy-based IDS for cluster-based battlefield networks. Their approach consists of three stages: cluster-head selection, fuzzy logic technique, and intrusion detection. The system periodically chooses the node with the maximum battery power to serve as a cluster-head in each cluster. This is done by having nodes share their residual energy levels at predefined election times. Nodes then vote to elect the node with the highest energy level as the cluster-head. The second stage of the system uses fuzzy logic to generate detection rules in the form of “if-then” clauses. The proposed system focuses on rule generation for data communications between regular nodes and control stations on the battlefield. When a node tries to communicate with a control station, the data packet passes through the cluster-head first as it serves as the central traffic point in its cluster. The intrusion detection (third stage) is performed by cluster-heads. These then apply the generated rules from the fuzzy-based engine against each packet passing through. If a match is found, the cluster-head considers it as a malicious packet and drops it. Otherwise, the packet is passed along until it reaches the destination.

Justin, Marathe, and Dongre (2017) presented a hybrid hierarchical IDS combining the accuracy of signature-based engines and the adaptability of anomaly-based intrusion detection. The system chooses the node with the highest energy as the cluster-head. However, Justin et al. (2017) did not explain how and when such selection occurs. The anomaly-based detection engine uses support vector machines for learning and establishing the normalcy profile during the training phase. Every node activates its signature-based engine to detect malicious activities by matching the collected data against the attacks-signature database. If no match is found, nodes activate their anomaly-based engine for further detection. If an anomaly is found, an alarm is sent to the other cluster-heads. When other cluster-heads receive this alarm, they send their vote to the originating cluster-head stating their findings from local detection against the suspicious node. If the majority of the nodes' votes support the suspicion, the originating cluster-head sends an alert to the entire network. In addition to that, the anomaly-based engine passes the detected attack to the signature-based engine to create a new predefined rule that can be used in future detections.

Despite the extensive research done in the area of hierarchical IDSs, this architecture imposes additional processing and communication overhead on other nodes in the network for electing and maintaining cluster-heads. Moreover, having a single node as a central point of detection raises the potentiality of such nodes to become the main target for malicious attacks. The success of such attacks can give malicious nodes the ability to take over the entire cluster and, eventually, the network. This, in turn, may cause false detection information reporting as well as fake accusations against legitimate nodes (Panos, Xenakis, & Stavrakakis, 2010).

Relying on dedicated nodes, such as cluster-heads, to perform heavy duty intrusion detection also entails a high processing overhead on the designated nodes. Considering the resource-constrained nature of MANETs, energy drainage of cluster-heads is a common issue in this type of architecture. Additionally, since each cluster-head represents a single point of failure in its cluster, this might lead to having groups of nodes disconnected and unable to communicate with each other, resulting in network segmentation (Butun et al., 2014).

Cooperative

In a cooperative architecture, each node in the network participates in the intrusion detection and response activities. This is done through having an IDS engine installed on each node, enabling it to collect audit data, detect intrusions, and alert the entire network when an intrusion is detected. Each IDS node analyzes the collected audit data and searches for evidence for intrusions. Such evidence is defined as signs of intrusions found in local or neighboring nodes' activities. Each node then assigns a certainty value to the collected evidence, which is called a confidence level. If a node finds a high confidence in the collected evidence, it marks the identified activity as an intrusion and alerts the entire network. At the same time, nodes can launch a cooperative intrusion detection when inconclusive evidence is found in the locally collected information against a suspicious activity. Inconclusive evidence situations occur when the level of certainty in the collected evidence is deemed too low for a node to make an intrusion decision on its own. In this case, nodes share their opinions, based on individual observations, to reach a final collaborative decision against the suspect node and initiate a global response (Soni, Ahirwar, & Agrawal, 2015).

The seminal work on cooperative IDSs (CIDSs) for MANETs was introduced by Zhang and Lee (2000). Their research proposed a new cooperative architecture for intrusion detection. Their system is an anomaly-based one that relies on both local and global detection mechanisms. Each node has its own local IDS agent that can detect and respond to intrusions. However, a global detection IDS is utilized when the confidence level of the detected intrusion is low. Thus, requesting neighboring nodes IDS agents to participate in the global IDS actions cooperatively. The system gives immediate neighboring nodes the highest values in evaluating an intrusion state. Zhang and Lee (2000) believe that such reports are accurate since compromised nodes do not have the incentive to send intrusion reports, fearing of their expulsion from the network.

Marti, Giuli, Lai, and Baker (2000) presented their watchdog and path-rater mechanisms on top of the DSR protocol towards the detection of misbehaving nodes. In their methodology, each node can overhear traffic passed by its direct neighbors. Each node maintains a buffer of the recently sent packets, which is then used to compare against the overheard ones from neighbors. The comparison entails checking for packet dropping and modification attacks. Dropped packets are detected if no match is found between the overheard and the recently sent packets. On the other hand, the system detects packet modifications through comparing the contents of the header/payload of the overheard packets against the recently sent ones. A node is marked as misbehaving if any packet drop/modification is found through this comparison, and the packet sender is notified of the misbehaving node. This is referred to as the “watchdog” mechanism. Along with this, Marti et al. (2000) proposed their “path-rater” mechanism that combines the knowledge of misbehaving nodes from the watchdog along with link reliability to provide optimal routing paths free of misbehaving nodes.

One weakness found in the watchdog mechanism proposed by Marti et al. (2000) is its inability to detect nodes that falsely report others as malicious. This weakness stems from their system's incapability to establish trust among nodes in the network. Thus, reports are considered trustworthy from any watchdog node. As such, compromised nodes may send false reports about legitimate nodes in the network causing them to be marked as malicious by others. This shortcoming in Marti et al. (2000) watchdog mechanism has pushed Patcha and Mishra (2003) to propose their collaborative system to detect black hole attacks on top of the AODV protocol. In their system, nodes in a MANET are classified into ordinary, trusted, and watchdog nodes. Every node must prove its trustworthiness to be included in the trusted group. Periodically, a node from the trusted group is chosen to be the watchdog based on energy, computation power, and storage capacity. This, as stated by Patcha and Mishra (2003), eliminates the problem of false reports encountered by the Mari et al. (2000) watchdog mechanism. Another weakness in the original watchdog mechanism (Marti et al., 2000) is its inability to detect packet dropping in the presence of colluding nodes. Patcha and Mishra (2000) attempted to overcome this problem by having each node send a message to the watchdog before forwarding any packet. Thus, a node is declared malicious if it doesn't inform the watchdog of a message forwarding event after exceeding a certain predefined time limit set by the system during initialization.

Based on a neighborhood-watch concept, Manikopoulos and Ling (2003) designed their mobile ad-hoc network security (MANS) system. Each node in MANS runs its own IDS that is responsible for collecting local data from the node itself as well as data from neighbors. All nodes exchange information periodically or when a suspicious event occurs. Majority voting from each neighborhood is employed when a node reports another as malicious. Results from voting are then propagated through the entire network. The system implements its own security policy, forcing nodes to (a) send the status of the IDS agent to the entire neighborhood at a given interval, (b) isolate a node that declares itself as compromised, and (c) opt to majority voting when a suspicious activity is reported. This policy is enforced by a controller module installed on each node, which is logically placed between a node and its network interface.

Stemming from their belief that intrusion detection should be carried out in a distributed manner, Huang and Lee (2003) proposed their anomaly-based CIDS. Their IDS relies on cross-feature correlation against anomaly detection models for routing protocols. During the training phase, the system establishes the normal network profile as well as a set of identification rules for well-known attacks. Their system constantly inspects and compares network traffic against such identification rules and triggers an intrusion alarm when a match is found. Cooperation is implemented in this system by dividing the network into cliques of nodes, with each having a periodically changing cluster-head. Cooperation in intrusion detection is carried out among all nodes in a clique where member nodes compute routing and location-related features while cluster-heads perform computations for traffic-related features. The system requires trained IDS models to be pre-installed on every node before deployment.

Sterne et al. (2005) proposed their CIDS based on a dynamic tree hierarchy in which intrusion detection data flows from the leaves towards the root of the tree. The system employs clustering for maintaining the proposed hierarchy with each cluster having its own cluster-head. The tree hierarchy is presented as having cluster nodes as the leaves. Those leaves then report to their cluster-heads representing a higher level in the hierarchy. At the root of the tree are security nodes to whom cluster-heads report their data. Security nodes are responsible for managing the intrusion detection capabilities for all clusters. These responsibilities include sending certain information to all other nodes in the network, such as intrusion detection directives and attack signatures. As the data flows from the leaf nodes to the root, it incrementally gets aggregated, filtered, and analyzed. An IDS is installed on each node for local intrusion detection. Each node is also responsible for reporting data about other nodes in the network. Such responsibility includes monitoring, logging, and analyzing data across protocol layers. If a node is able to detect an attack on its own, then it would only send an alert message to its cluster-head. However, if it is unable to make a decision, it transfers its local detection data upwards through the hierarchy to further aggregate it with other collected data. Despite the anticipated efficiency of the model, proficiency of the proposed IDS is not determined as the model was not simulated or applied towards real-life scenarios.

Stamouli, Patroklos, Argyroudis, and Terwari (2005) attempted to address the issue of high false alarm rates found in IDSs for MANETs by proposing their real-time intrusion detection for ad hoc networks (RIDAN). Their system is a knowledge-based one that utilizes time finite state machine (TFSM) to detect real-time routing attacks against the AODV protocol. All nodes in the network must have RIDAN installed on them. When a node detects an attack, it attempts to avoid the malicious node from its future routing until it goes back to behaving normally. However, the detecting node does not share its findings with its neighbors. Building on Stamouli et al. (2005) approach, Ding and Xu (2006) proposed their enhancement to RIDAN by introducing cooperation in their real-time cooperation intrusion detection system for MANETs (RCIDMANet). The system implements cooperation modules that allow building cliques and choosing one monitoring node for each clique. Nodes in a clique cooperate with each other through information sharing and by conveying such information to the monitoring node. When a node detects a suspicious activity, it consults with other nodes to determine whether the identified behavior is an attack. Through the application of cooperation, RCIDMANet showed 6%-10% improvements in detection accuracy over RAIDAN.

Coming from their faith in the beneficial characteristics of CIDS in MANETs, Deng et al. (2006) presented their agent-based cooperative anomaly detection model. In their system, the network is divided into clusters, each with a periodically-changing cluster-head. The intrusion detection feature information is collected from member nodes and sent to the cluster-head to perform anomaly detection. This is done through an anomaly detection engine comprised of one-class support vector machines (1-SVMs) algorithms. Cluster-heads have the responsibility of instructing their cluster nodes on how feature extraction should be performed. When an attack is detected at the cluster-level, an alarm is sent to the entire network. The model relies on training data acquired from modeling normal network behavior.

Trang, Kong, and Lee (2006) developed their CIDS based on the work by Zhang and Lee (2000), targeting the AODV protocol. Their system assumes that malicious nodes do not have an IDS installed on them. Each legitimate node in the network is equipped with its own IDS, which detects anomalies based on inspecting AODV's packet headers during the route discovery phase. Specifically, the system checks for the sequence number in the received route request (RREQ) packets. Nodes locally store each RREQ packet received from others during route discovery operations. When a rebroadcast RREQ is received by a certain node, it compares the received request to the locally stored ones. If a match is found and the sequence number of the rebroadcast RREQ is different than the stored one, an intrusion is detected and the node floods the network with an "ALARM" packet. Detecting nodes cannot send "ALARM" packets to themselves or to any malicious node. A node is considered malicious if it doesn't reply to the "ALARM" message. The system demonstrated a number of false alarms during simulation as a result of losing replies to the “ALARM” message during transmission. The model was tested against two types of attacks: flooding and sequence number modification.

Suggesting that a Bayesian approach can improve intrusion detection accuracy, Karim, Rajatheva, and Ahmed (2006) presented their CIDS based on such an approach. Each node in the system has a local IDS collecting packet information from both data and network layers. A Naïve Bayes classifier is then used to compare the collected information against a set of predefined anomaly detection rules. If such a comparison results in the detection of a suspicious activity, a node would initiate a global alert and contact its neighbors to start the cooperative detection. A decision is made regarding the reported node after combining the locally collected observations from all nodes. If the final value crosses a predefined threshold for an anomaly detection rule, the reported node is marked as malicious. The proposed system responds to intrusions by rearranging the network to exclude malicious nodes.

In efforts to address the common problem of high false positives in IDSs for MANETs, Otrok et al. (2007) developed their IDS using a cooperative game theory approach. The system targets two types of attacks: cache poisoning and flooding. Nodes cooperate with each other to detect and respond to attacks. If a node experiences high rates of packet loss due to it not receiving acknowledgments to the packet it already sent, it suspects a cache poisoning attack. However, if a node receives more packets than the expected rate, a flooding attack is assumed The system introduces the concept of security classes to reduce false positives. These classes take into consideration the reputation and contribution of attack reporting using game theory as well as the severity of attacks reported by each of the cooperating nodes in the detection process. If the result of calculating the security class exceeds a certain predefined threshold, an immediate local or global response is triggered.

Bose, Bharathimurugan, and Kannan (2007) presented their multi-layer CIDS, which operates on MAC, network, and application layers in MANETs. Bose et al. (2007) believed the effectiveness of such a system comes from the fact that if an intruder node escapes one layer of detection, it would be caught in another. The system designates three anomaly detection subsystems using Bayesian classification, Markov chain construction, and association rule mining algorithms for MAC, routing, and application layers respectively. Each node combines its local intrusion information from each of the anomaly subsystems through a local integration module. The results are then combined with those obtained from cooperation with neighboring nodes via the global integration module. A global intrusion module is then used to make a final decision and initiate a response towards the identified attack.

Based on the concept of evidence chain (EC) and trust fluctuation (TF), Wang, Huang, Zhao, and Rong (2008) proposed their intrusion detection mechanism based on trust model for MANET (IDMTM). The system regards malicious behaviors from nodes as evidence. As time passes, these pieces of evidence are aggregated into an EC. This occurs until the system has enough evidence to identity the suspicious node as malicious. On the other hand, the system employs a TF mechanism in which each node is assigned a trust value. Changes in this value are monitored over time. The greater the changes of the trust value for a node, the more likely it has been compromised or turned malicious. Trust values are cooperatively calculated through observations of neighboring nodes. The combination of collected ECs and TFs result in a judgment being made against a suspected node regarding its maliciousness.

Sen, Chaki, and Chaki (2008) argued that most IDSs lack the precision to properly identify malicious nodes and tend to permanently isolate such nodes from the network. This, in turn, results in eliminating the chances for the accused nodes to recover their trust. Having this issue as their target, Sen et al. (2008) proposed their CIDS based on the concept of an honesty rate index (h-rate). Each node is assigned an h-rate that increases when it cooperates and decreases when a node behaves maliciously during a certain predefined interval. Each node stores a table for neighbors' h-rates and maintains a record of its own h-rate. Neighbors monitor each other's performance, and the h-rate gets recalculated based on the current observations. The IDS randomly selects a node to be the monitor and changes it to another at random intervals. The system implements packet signing by using public and private keys assigned to nodes through polynomial secret key sharing.

Sen, Ukil, Bera, and Pal (2008) proposed a fully cooperative system that relies on reputation and voting mechanisms. Using a monitor module, every node in the network monitors its neighbors for packet dropping and modification attacks. If such attacks are suspected, a reputation module is activated to compare the average of neighbors' opinions against the one declared by the accused node. Majority voting is then employed to make the final decision on whether the accused node is malicious. The system encrypts all intrusion-related communications against replay attacks. Each node receives a trust certificate when cooperating in the group voting through a trust maintenance module. A table containing all reputation values of malicious nodes is stored on each node. A reputation propagator module conveys the newly calculated reputation values to the neighboring nodes at regular intervals. If an intrusion decision is made, after majority cooperative voting, the participating nodes flood the network with alarm messages, and an alarm raiser module is invoked to take a response action.

Ebinger and Bibmeyer (2009) targeted the establishment of a distributed trust system as an approach to cooperative intrusion detection in MANETs. Each node calculates a trust value for every neighboring node. This value gets updated at each predefined interval to address topology changes. Each node must monitor its one-hop neighbors, and the resulting measures are used cooperatively between all nodes in the network to make intrusion decisions. To conserve resources, IDS values are updated at fixed intervals. The system uses the AODV routing protocol to distribute reputation information piggybacking on normal routing packets.

Tangpong, Kesidis, Hsu, and Hurson (2009) presented a cooperative intrusion detection model targeting Sybil attacks, specifically. The model only considers simultaneous Sybil attacks in which attackers use their fake identities to consume a larger share of the wireless channel access. The model does not solve colluding or join-and-leave scenarios for Sybil attacks. Each packet is signed with a private key before sending. Upon receiving a packet, a node verifies the packet signature and the location of the sending node. Packet signatures along with corresponding fields are cached in the receiving node and periodically shared with other nodes in an attempt to uncover Sybil attacks. The model performs a comparison against the cached observations to look for co-occurrences of multiple similar paths coming from the same region with the same identities. If the number of such occurrences exceeds a predefined threshold, a node is declared as Sybil.

Believing in the Hierarchical Graph Neuron (HGN) algorithm’s ability to increase detection accuracy, Mahmood et al. (2009) came up with their CIDS based on a distributed HGN (DHGN). The system goes through an initiation stage, in which a normal network profile is obtained and defined. In the proposed architecture, each node monitors its local traffic pattern and calculates whether a node has left the network or is still alive in the form of a “neuron index.” This index is then passed from all nodes to a master node that is chosen to have low mobility and high battery power. The master node possesses a database of normal network profiles obtained at the initiation stage and can determine an attack occurrence by comparing such profiles to the received information from the cooperative nodes. However, if the confidence calculation performed by the master node has a lower rate, the final detection decision is made based on majority voting. Mahmood, et al. (2009) did not provide an evaluation of the applicability or efficiency of the proposed system.

Shresta, Han, Choi, and Han (2010) proposed a cross-layer cooperative anomaly IDS. This stemmed from their belief that utilizing data from different layers of the protocol stack can improve detection accuracy. Information from MAC, routing, and physical layers are extracted through the system's association module to form a rule set that is used for anomaly profiling. A local data collection module aggregates data streams, related to traffic patterns, from the association module. These streams are then passed to the local anomaly detection module, which analyzes such data against intrusions. This module relies on normal behavior data collected across layers during the training phase. If an intrusion is found with high confidence, a node can locally determine the occurrence of an attack and initiates a global alarm. However, if the detection is with low confidence due to insufficient evidence, a cooperative detection is requested from surrounding nodes through a secure channel. A final decision is made according to majority voting. An alert management module exists to collect evidence from both local and cooperative engines and initiates an alarm accordingly to inform the network of the identified attack(s).

Morias and Cavalli (2012) came up with their CIDS based on routing protocol analysis. Each node captures and analyses every packet going through its network interface. A routing protocol analyzer engine examines all the routing information of each packet to generate routing events. The system defines various constraints on the normal behavior of the routing protocol to detect intrusions. Detected inconsistencies in nodes' routing behavior are then compared against a predefined threshold. When such threshold is crossed, a routing attack is considered. The system uses a cooperative consensus mechanism to obtain a majority-based decision regarding the maliciousness of the suspected node. As a response to a detected attack, the detecting node sends an alert message to all others regarding the malicious node. Along with that, the system isolates the malicious node from the network by discarding its packets and depriving it of using the network resources.

Alattar, Sailhan, and Bourgeois (2012) introduced an Intrusion Detection and Cooperative Response (IDAR) system that is a log- and signature-based IDS targeting the OLSR protocol. The system is installed on each node to collect OLSR logs and compares them against predefined attack signatures to detect intrusions. To limit bandwidth consumption, evidence obtained from these logs is divided into different groups according to their degree of suspicion. This, in turn, provides more efficiency in determining whether a cooperative detection is needed or a node can detect the attack on its own. If a node does not have sufficient evidence against a suspected activity, it quickly starts a cooperative networked detection to either confirm or affirm its suspicions and ends the investigation promptly.

Utilizing neural-fuzzy as the core detection engine, Tabari, Pouyan, Hassanpour, and Saleki (2012) proposed a lightweight semi-distributed IDS for MANETs. The system defines three modes to classify the status of nodes in the network: normal, attack-presented, and suspicious. The IDS engine can easily detect an attack-presented status through the local IDS, which runs on every node in the network. However, a suspicious status may occur in the normal mode where a node might have rapid movements as well as the attack-presented mode where attackers perform low-level attacks or are positioned away from the victim. In these cases, the system turns to the cooperative investigation to resolve ambiguity and proceeds to carry out a final decision regarding the doubted activity. The final decision is made by asking neighboring nodes for their sate in relation to the identified activity. The model was compared to a sample stand-alone architecture to prove its detection efficiency and performance.

Malek and Khorsandi (2013) presented a CIDS that is independent of any routing protocol. The system divides the network into zones based on the total number of nodes and resources. Three types of nodes exist in the proposed system: general, header, and monitor. General nodes are divided into clusters that have a separate cluster-head for each cluster. General nodes are agnostic to the intrusion detection activities and have no knowledge of the different types of packets passing through. Header nodes must have enough resources and are considered trustworthy and fault-tolerant, so are monitor nodes. Header nodes perform continuous broadcasting of voting packets and ask general nodes to rebroadcast these packets. The system assumes that legitimate general nodes rebroadcast the voting packets without modification while malicious nodes tend to drop or tamper with the contents of such packets. Monitor nodes, on the other hand, receive and inspect the results of rebroadcasting from general nodes against packet dropping and modification and send their observations back to header nodes. Based on such observations, header nodes determine normal, suspicious, and malicious nodes. The voting process is repeated for a certain number of times according to the number of nodes in the cluster and traffic load. Both suspicious and malicious nodes are isolated from the routing process. However, the main difference between the two is the possibility of reassessment for suspicious nodes, which could allow them to be re-included in the cluster. Normal nodes are recorded in a white list, suspicious ones are added to a gray list, and malicious nodes are added to a black list. These lists are broadcasted periodically by header nodes to all general nodes in the cluster.

Mustafa and Xiong (2013) proposed a CIDS that specifically targets routing attacks in MANETs. Each node in the system monitors its one-hop neighbors for malicious routing activities. Nodes in the network share their monitoring information with each other. Each node makes a detection decision based on both local and global observations obtained from other nodes. A maximum normed residual test (aka Grubb's test) is used to prevent inconsistencies in the collected observations and to detect fake information from malicious nodes. When an intrusion is detected, the detecting node shares its decision with other well-behaving nodes and isolates the malicious node from routing services. The system grants a chance for suspicious nodes to return to their normal behavior by periodically reassessing observations. A suspicious node can reutilize routing services after adjusting its behavior back to normal.

Adhikari and Setua (2013) proposed a cooperative network intrusion detection system (CNIDS) that is tailored for the DSR protocol. Every node in the network periodically runs a context analyzer to check whether it has neighbors. If not, the context analyzer component is disabled for that period to save energy. A watchdog system is implemented on each node to ensure packet forwarding by its one-hop neighbors through promiscuous overhearing of transmissions. If a neighbor does not forward a message, the IDS of the overhearing node records an anomalous event for that neighbor in the reputation table. Each node maintains a reputation table containing information regarding other nodes in the network through direct observations of its one-hop neighbors and indirect observations received by alert messages from other nodes. When a node receives an alert message regarding another node, it activates the "ALERT message verifier" component to verify if the information contained in the received alert is true before updating the reputation table. This is done through sending and overhearing the transmission of a test packet through the suspected node for one-hop neighbors. If the suspicious node is not a direct neighbor, the receiving node checks the reputation table to check that the alert initiator node is marked as “normal” and that it’s truly a direct neighbor of the reported node. Malicious nodes are punished by having normal nodes drop their packets. The system sets a predefined threshold for the maximum number of packet drops by a node. When the threshold is crossed, a node is marked as malicious.

Prasannavenkatesan, Raja, and Ganeshkumar (2014) introduced a CIDS relying on their own packet dropping detection (PDA) algorithm. In the proposed system, a MANET is divided into clusters. Each cluster has a cluster-head that is elected based on majority voting from all member nodes. Mobility, degree, energy levels, and transmission range are the four qualities considered when electing a new cluster-head. Each node is equipped with an IDS that passively listens to one-hop neighbors to detect packet dropping attacks. If a node detects a neighbor dropping or modifying packets, it requests a cooperative detection from other nodes in the cluster. Such detection is driven by the cluster-head through forcing each node to respond with the degree of maliciousness of the suspected node. The cluster-head marks the target node as malicious if the majority of the received responses from cluster members indicate so. As a responsive action, the cluster-head then floods the network with an alarm message containing information about the detected attacker.

Sharma (2015) presented a cooperative intrusion detection approach that specifically targets selective packet dropping of gray hole attacks. Each node in the network is equipped with an IDS that monitors its direct neighbors. If a neighbor continuously drops a packet, it’s marked as a black hole attacker. Since uncertainty usually surrounds gray hole attacks due to their selective nature, the proposed IDS has each node maintain a statistical table containing information about the analyzed packets. The system defines a packet dropping threshold. When this threshold is crossed at a certain time, a node is put in a gray hole attacker list maintained by its neighbors and is removed from their routing table. If a node suspects a gray hole attacker, it sends a request packet to its neighbors to consider the suspicion. Neighbors check their own gray hole lists to confirm the suspicion. The information continues to pass to the next level of neighbors for a predefined period. After receiving all responses, the initiating node checks if two-thirds of the received responses confirm the existence of the suspected node in their gray hole attacker list. Then, it marks the suspected node as a gray hole attacker. The IDS does not broadcast an alarm for the network after discovering gray hole attackers. The alarm is not issued because of a fear that the malicious node would start behaving normally once it receives the alarm.

Merchang, Datta, and Das (2017) believed that resource consumption, in a CIDS can be minimized by introducing the concept of "security levels." Since each node in a CIDS is monitored by all its one-hop neighbors, higher energy consumption would occur. However, defining a security level can solve this problem by having only a certain number of neighbors monitor the target node. A security level defines the number of neighbors designated to monitor a certain node at a given time. A threshold is set to maintain the minimum security level based on the application scenario. However, the problem with this technique is that nodes are inclined to save their energy and avoid IDS activities, which might result in an ineffective system. This is tackled by Merchang et al. (2017) by employing a cooperative game theory approach to establish an equilibrium between detection activities and energy savings. As such, the defined game forces two goals on all nodes: participating in monitoring activities and reducing energy consumption. The system defines a certain interval for all nodes to use. At each interval, a node determines whether its IDS should be active based on the calculated probability according to the selected security level. During active times, if a malicious node is detected, the detecting node broadcasts a vote message to neighbors. The selected security level determines the number of votes needed from neighbors for the suspected node to be marked as malicious.

Cooperative intrusion detection overcomes the weaknesses in both stand-alone and hierarchical architectures. The primary advantage of this architecture against the stand-alone one is represented by the cooperation mechanisms that enables a broader view of the network, which results in the ability to detect complex attacks. On the other hand, CIDS defeats the hierarchical one by removing the reliance on a single point of failure and distributing identical detection engines on every node in the network. In general, due to the dynamic nature of MANETs, cooperation is imperative to achieve real-time detection accuracy, through providing a shared view on the security situation of other nodes in the network (Morais & Cavalli, 2012).

Summary

This chapter reviewed the current body of knowledge on MANETs, their characteristics, security issues, and vulnerabilities. In addition, a review and comparison between the various intrusion detection architectures and implementations for MANETs was presented. Due to their unique characteristics and inherent vulnerabilities, the quest for an efficient IDS continues to be a challenging problem for researchers in this field. The researcher has found significant emphasis from previous studies on the critical need for cooperative detection towards achieving such efficiency. In reviewing the current literature, the urgent need for an efficient CIDS implementation for MANETs that is able to simultaneously identify malicious attacks with high accuracy and minimal communication overhead has become the motivation and driver for this research.

Chapter 3

Methodology

Overview of Research Methodology

This chapter details the methodology that was followed to implement this research study. Experimental design was implemented through the design and development of the proposed IDS. This is a common approach followed by similar studies to create and assess new IDS implementations for MANETs (Huang & Lee, 2003; Kareem et al., 2006; Wang et al., 2008; Cannady, 2009; Tabari et al., 2012; Adhikari & Setua, 2013; Merchang et al., 2017). The system was evaluated to determine its feasibility in achieving the research goals of increasing detection accuracy while minimizing communication overhead. The application of the concept of social communities accompanied with DST targeted the achievement of these goals. The researcher followed four major steps in the development of the proposed system: design of the proposed IDS components, implementation of the components, integration of the components into a deployable IDS, and testing of the IDS implementation. Detailed implementation steps are provided along with the specific procedures that were followed for attacks generation, system development, data collection, and system evaluation. Resources needed for this implementation are also outlined in this chapter.

Approach

The goal of this research study was to implement a CIDS that is able to achieve high detection accuracy while minimizing communication overhead. To achieve these research goals, the design and implementation of the proposed system depended on combining the concepts of social communities and DST. Nodes in MANETs represent characteristics similar to social behaviors of humans in a community (Banerjee, Nandi, Dey, & Saha., 2015). As such, and following Granovetter's (1973) definitions of strong and weak ties in social communities, we treated a MANET as a web of strongly tied social communities connected with each other through weak ties. Strong ties in this context are defined as social links that are formed between nodes based on recency of communications, age of communications, reciprocity, and knowledge sharing (Gilbert & Karahalios, 2009). On the other hand, the lack of such ties between two nodes indicates the nonexistence of social links between them, which is defined in this context as weak ties. Nodes in each community are connected through strong ties. However, for nodes inside these communities to interact with other communities, they utilize the weak ties as a communication bridge. Each node in the network, along with its circle of strong ties, represents a densely-knit mass of social structures. This is called a "social community". Each node would also have a set of weak ties connecting it to distant communities. Sometimes for a node to connect to one distant friend, it needs to utilize one of the weak ties as a crucial bridge to access information beyond its own social circle (Granovetter, 1983).

The application of the social community theory in our approach aimed to address the high bandwidth consumption issue found in current solutions. This is due to our unique implementation of such concepts, which eliminated the need for high information dissemination, thus minimizing bandwidth consumption required to establish nodes' social circles. Details of this implementation are provided in the next sections. The application of social communities aimed to improve detection accuracy as well. The detection accuracy is improved through a focused reliance on reports coming solely from reliable social circles. This is because, in a social community, received information is rarely trusted unless it comes from strong ties (Granovetter, 1973). However, for this research to achieve high efficiency regarding the second goal, which is decreasing the number of false positives, Dempster-Shafer theory (DST) of evidence (Shafer, 1976) was utilized. Although this research relied on reliable reporting, false positives would still occur due to a lack of sufficient evidence against a suspicious activity. Thus, the application of DST here aimed to decrease and potentially eliminate cases where a node is falsely accused due to a lack of evidence or cases of partial evidence.

Mainly, there are two commonly used theories for combining multiple beliefs from different entities: Bayesian theorem and DST (Li & Joshi, 2009). Bayesian theorem considers the lack of knowledge regarding an incident as a negative evidence (Gordon & Shortliffe, 1984). In other words, if two nodes are asked to give their observations regarding a suspected third neighbor and one of them fails to report such an observation due to a collision or unreceived request, Bayesian theorem considers it as negative evidence towards the suspected node. Besides, such theorem usually requires training data ahead of time to achieve an efficient accuracy. However, the reliance on training data in attack detection is problematic since adversaries can change their behavior with time (Li & Joshi, 2009). As opposed to the Bayesian theorem, DST does not require training data or prior knowledge of an incident. Additionally, DST does not regard the lack of knowledge as negative evidence because it can hold either a supportive or uncertain view about an incident (Gordon & Shortliffe, 1984).

The main issue with the current applications of DST, found in the literature, is the determination of trustworthiness and untrustworthiness of nodes when weighing in nodes' votes (Chen & Venkataramanan, 2005; Li & Joshi, 2009). This requires a high bandwidth overhead imposed by trust-related information dissemination to establish accurate calculations. Otherwise, DST can combine observations from nodes disregarding their trustworthiness. However, this might yield inaccurate results in the presence of a large number of malicious nodes in the network (Chen & Venkataramanan, 2005). Very little research has been done towards the application of DST for intrusion detection in MANETs. All of which relied on the calculated trustworthiness of nodes in the combined decision-making process. This can be problematic in the presence of a large number of malicious nodes, which in turn, can result in manipulated votes against legitimate nodes (Rajakumar et al., 2014).

In our approach, we eliminated both the overhead of trust calculations as well as the inclusion of anonymous votes, which usually result in a high rate of false alarms. Instead, we used DST against observations obtained solely from strong-ties to handle cases where some of these nodes did not catch any/enough evidence against the suspicious activity. Details of the application of DST are provided in the next sections.

Research Methods Employed

For this research to achieve the outlined goals, the researcher adapted the following sequential steps for the implementation of the proposed system. This ensured systematic development and execution of all the pieces necessary to conduct the experiment and obtain the required metrics to document the research findings.

- Create an initial MANET for testing the implementation of the proposed system. The designated MANET contained a number of legitimate mobile nodes communicating with each other over the AODV routing protocol. The researcher chose the AODV protocol as it is widely used for intrusion detection experimentation for MANETs. A large number of research studies have opted to use AODV as the routing protocol of choice for such experimentations (Cannady, 2009; Huang et al, 2003; Parasannavenkatesan et al, 2014; Wang et al., 2008; Otrok et al., 2007; Shrestha et al., 2010; Karim et al., 2006).

- Develop deployable attack models for black hole, gray hole, modification, rushing and flooding attacks. The development of these attacks was based on their definitions outlined in Chapter 2. This is identified further in the “Attacks Generation” section.
- Develop testing scenarios for each of the above-mentioned attacks. Mixtures of multiple attacks in the same scenario along with a variable number of attackers was developed as well.
- Perform iterative testing on the developed scenarios to define and adjust the required thresholds for the proposed IDS operations. This is identified further in the “System Design” section.
- Develop the components of the proposed IDS outlined in Figure 1 in the upcoming “System Design” section. This entailed the application of the concept of social communities and DST towards achieving the research goals.
- Test the IDS against the previously developed testing scenarios.
- Collect datasets by continuously monitoring the testing MANET in various operational models. Details of the data collection process are provided in the next sections.
- Collect evaluation metrics to determine the system’s ability to feasibly achieve the research goals of high detection accuracy with minimal communication overhead.
- Document findings in the Dissertation Report.

Attacks Generation

To simulate the proposed system’s response and behavior during malicious attacks on the network, the researcher implemented the following attacks based on their definition outlined in Chapter 2. For all these attacks, a new attacker-node module was created to simulate each one of them. This also allowed the researcher to keep the normal node behavior intact as we implemented each attack. Making the attacker-node module optional, allowed the researcher to flexibly install it whenever necessary to simulate any of the following attacks:

- Black hole: The attacker node tries to attract as many routes through it as possible. AODV was the routing protocol followed in the simulation. Thus, implementation of route attractions from the black hole attacker was focused on the route discovery phase in this protocol. Whenever the attacker-node receives a RREQ message, it generates an RREP message to the originator stating that it has a fresh route to the destination. This is done by setting the sequence number in the RREP packet to the maximum allowed number in AODV. When the requesting node receives the reply packet, it starts communicating with the destination node through the route received from the attacker. When the attacker-node receives these packets, it immediately drops them without forwarding it to the destination.
- Gray hole: this attack is similar to black hole except it follows a selective packet dropping pattern. However, the attacker-node still implemented the same procedure outlines present in the black hole attack simulation above. This is done to attract nodes to route their packets through it. The main difference in the simulation would be the implementation of selective packet dropping. This was implemented as follows- whenever the attacker-node receives a data packet, it generates a random number (RND_GH). If this random number falls between a predefined range (GH_RANGE), the attacker-node drops the packet. Otherwise, it would forward it without dropping.

The researcher varied the RND_GH and GH_RANGE through extensive testing to establish a similar resemblance of selective packet dropping to real-life attack scenarios.

- Modification: in this attack, when an attacker-node receives a data packet, it appends a randomly-generated number to the contents and forwards the packet along the route. This is done by copying the received data packet into a new one. After that, the attacker-node modifies the payload of the copied packet by appending the random number. Finally, it removes the original packet and instead forwards the copied/modified on down the route.
- Rushing: at the network initialization phase, this attacker-node generates a large number of RREQ to ensure it will be included in as many future routes during nodes' communications. This was implemented by installing an antenna with a higher transmission power on the attacker-node than those of legitimate nodes. Thus, when the attacker receives a RREQ, it forwards it to the next hop faster than a legitimate node is able to forward it. That way, more nodes will potentially include the attacker in future routes. This would cause legitimate routes to eventually become neglected as more nodes will have the attacker in their routing tables.
- Flooding: the attacker-node creates a new data packet with size equal to the maximum allowed packet size in the AODV protocol. The attacker then starts flooding the network with these packets by sending it to all its one hop neighbors as well as to fake destinations. That way, when other nodes receive such packets, high processing would be needed to inspect each one to find the destination to forward to. Continuously receiving such packets causes legitimate nodes to start draining battery power that is being spent to route such processing-intensive packets. This would potentially cause

nodes to drop out of the network due to full drainage of their battery, rendering them incapable of participating in normal network operations.

System Design

In this research, we proposed a novel CIDS for MANETs comprised of the following components:

- Data Inspection Module responsible for gathering traffic data.
- Social Ties Builder Module for establishing social communities between nodes.
- Local Detection Module for analyzing the collected data against suspicious activities.
- Cooperative Detection Module that is responsible for launching collaborative investigations with other nodes.
- Global Response Module for broadcasting alarms when an intrusion is detected.

The IDS was implemented as an integrated module that can be optionally installed on the designated nodes. This allowed the implementation of attacker and IDS nodes in the network at the same time by specifying the required module to be installed on each node, without modifying the normal behavior of nodes.

Figure 1 below illustrates the overall structure of the proposed system.

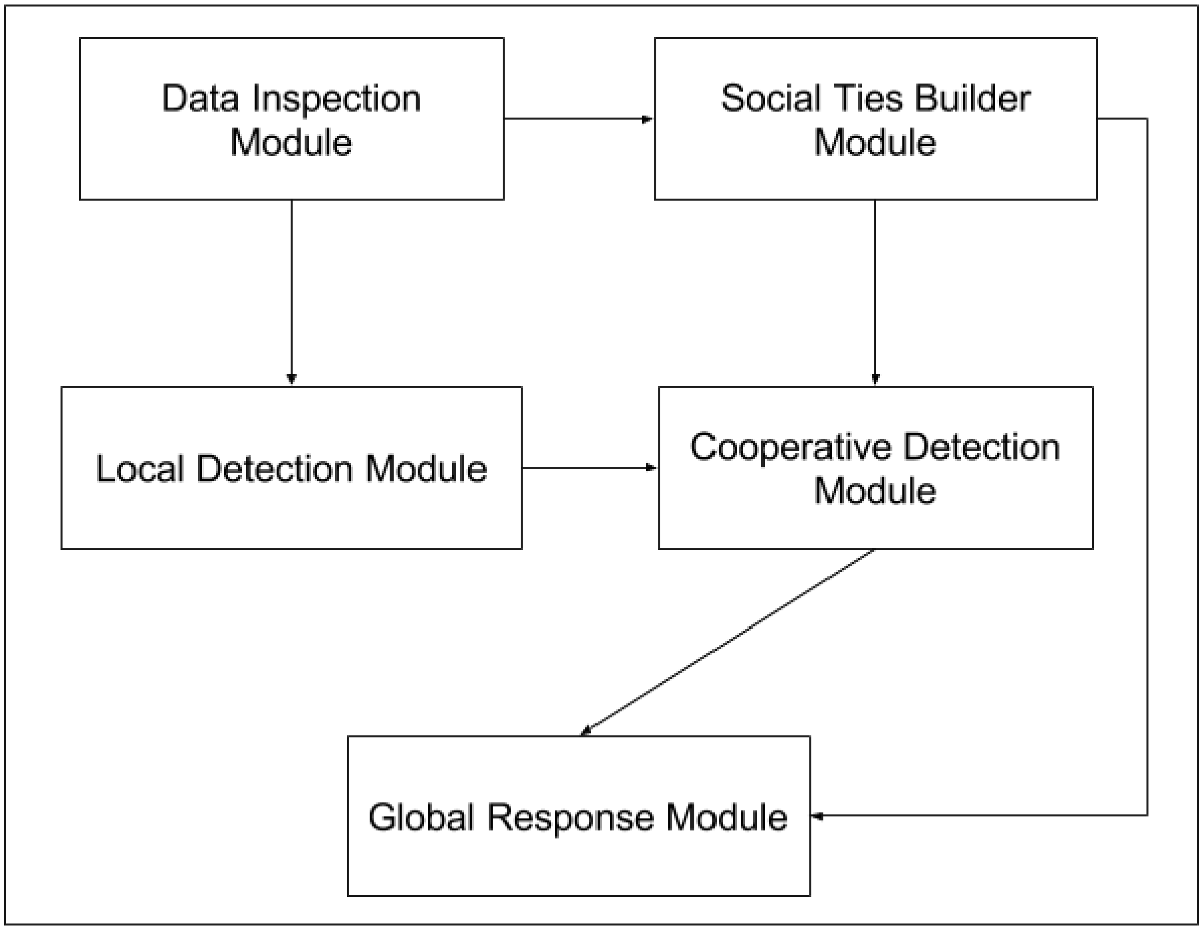

Figure 1 - System Structure

Data Inspection Module

Each node in the network activates its promiscuous mode enabling it to overhear traffic passing through its one-hop neighbors. The data inspection module (DIM) has two main responsibilities: data collection and inspection and selective packet forwarding. The DIM collects statistical data about each received/sent packet as well as neighbors' forwarded packets. This includes recording all the information in the packets' headers as well as calculating a checksum for each packet. The collected information is saved on each node so that it can be consumed by the social ties builder, local detection, and cooperative detection modules. To perform selective packet forwarding, this module checks the social ties table (STT) (defined next in the Social Ties Builder Module) before forwarding a packet to check if the source node is marked as “Malicious” then drops the packets, otherwise, the packet is forwarded.

The implementation of this module was as follows: whenever an IDS node receives a routing/data packet, it goes through the DIM. Before any further processing, this module extracts the source IP address from the received packet. The DIM then asks the STBM whether this source IP is marked as “malicious”. If so, the DIM drops the packet. Otherwise, the DIM extracts the packet type (routing or data), source/destination addresses, and packet time, and it calculates a checksum for the packet payload. The extracted data is then stored in the memory for future utilization by other modules. The packet is then forwarded to the neighbor down the route and is marked in the memory storage as “forwarded to neighbor.” The IP of the neighbor the DIM is forwarding the packet to is appended to the packet entry in the storage. This memory storage represents the table for storing DIM data. This table is called the data collection table (DCT). Since IDS nodes activate their promiscuous mode at all times, they are able to sniff packet transmissions of their one-hop neighbors. Thus, whenever a packet is forwarded by a one-hop neighbor, the LDM extracts the packet type (routing or data), source/destination addresses, packet time, and IP address of the forwarding neighbor, and it calculates a checksum for the packet payload. The packet is then marked as “forwarded by neighbor” in the DCT.

Social Ties Builder Module

This module is responsible for constructing social ties with other nodes based on the collected information from the DIM. Due to the constrained nature of nodes in a MANET, the social ties builder model (STBM) is activated after each time interval (t). That way, we can reduce energy consumption and potentially extend battery life, as opposed to having the module run indefinitely. After each interval (t), this module queries the collected data from the DIM to extract and calculate social ties features. These features are derived from Gilbert and Karahalios (2009) findings regarding the significant predictors ($p<0.001$) of strong ties in a community. These features are the following:

- Recency of communication: this is calculated by recording the time a packet is received from a certain node.
- Time since first communication: this value is saved when a node receives a packet from another for the first time.
- Reciprocity: this denotes the number of messages exchanged with other nodes.
- Shared Knowledge: this denotes the number of route discovery or link failure messages shared by other nodes.

At each interval (t), these features are extracted and stored on each node inside the STT. The novelty of this module against the current solutions is twofold: a tremendous decrease of bandwidth consumption and high reliability of intrusion detection information. Each node utilizing this module performs its calculations to establish a strongly-tied social circle. This is done without the need for reputation or trust information to flood the network. As compared to all current solutions, this demonstrated substantial enhancements in bandwidth consumption, which resulted in fewer collisions and less packet loss throughout the network. The other primary benefit from this implementation is its reliance on strongly-tied communities to deliver reliable information. Compared to the current trend in getting such information, researchers lean towards reputation-based mechanisms based on the observed node behaviors from neighbors. However, such an approach is error-prone as legitimate nodes can be marked malicious due to accidental packet drops. The other downside of reputation-based detection is its inability to perform when a high number of malicious nodes exists in the network, as compared to legitimate ones (Rajakumar et al., 2014).

A node is considered a strong-tie on strict conditions: if and only if its recency of communication is less than the predefined threshold (r_{rec}), time since first communication is greater than the predefined threshold (r_t), and the combination of reciprocity and shared knowledge is greater than the specified threshold (r_{rsh}). On the other hand, a link between two nodes is considered a weak tie if such conditions are not met. It should be noted that “recency of communication” also helps in detecting nodes that have dropped out of the network and those which turned malicious after a certain period. This stems from the assumption that such nodes tend to stop message exchange in case of leaving the network or cease to cooperate in the routing process in the event of compromised nodes, resulting in removing them from the social circle.

As far as the distribution of ties, the proposed system had high densities of strong ties among nodes inside each social community. Weak ties were more distributed as connectors between strongly tied communities. The existence of both is important to keep the flow of normal network traffic undisrupted. The system does not isolate nodes that are connected with each other through weak ties. Such nodes can still communicate with each other as well as with other nodes in the network without issues. On the other hand, nodes that were identified as malicious are excluded from the routing process among strongly tied communities.

An important question arises here: what if malicious nodes drop and rejoin the network with different identities? Since the system uses strict conditions to form strong ties, such nodes would not be considered as part of the intrusion decision making process when they rejoin. Over time and under the continuous monitoring of our system, these malicious nodes get caught again performing malicious acts and isolated from the routing process. However, if these nodes were behaving selfishly due to limited resources, such as low battery power, and their behavior changes after rejoining, they will be able to communicate with other nodes and possibly even form strong ties with others if their normal behavior is consistent.

This module was implemented as follows: the researcher defined a set interval (t) to activate the STBM operation. At every (t), this module sends a message to the DIM requesting the newly collected data, which is defined as the data collected between now and the last interval. The DIM then fetches this data from its memory storage (DCT) by selecting records with packet time between now and last interval (now – t), where (t) is the activation interval for STBM. The DIM then passes down the records to the STBM. The STBM has its own memory storage, named the social ties table (STT). At this point, the STBM checks if it has the source IP of each received record in the STT. If it doesn’t, a new record is created by extracting the above-mentioned ties features from the currently inspected packet entry. Otherwise, this module updates the record for the existing IP address. Table 2 below shows the items involved in the calculations of the social ties feature from the currently inspected packet entry:

Feature	Calculation
Recency of communication	Time packet received.

Time since first communication	Time packet received (for new records only). This value doesn't get updated for nodes that already have records in the STT.
Reciprocity	This value is incremented by one if the currently inspected packet is a data packet.
Shared Knowledge	This value is incremented by one if the currently inspected packet is a routing packet.

Table 2 - Social Ties Features Calculation

At each interval (t) when this module is done updating the records in the STT, it runs a check to see if a new strong-tie can be formed and if it can remove obsolete ties due to their inactivity. This is done by checking each record in the STT to see if r_{rec}, r_t, and r_{sh} are satisfied. If so, the entry is marked as a strong-tie. On the other hand, for existing ties that have become obsolete for reasons such as leaving the network, this module calculates if r_{rec}, r_t, and r_{sh} are still viable. In this case, r_{rec} would not be satisfied, and the "strong-tie" label would be removed from that record.

Local Detection Module

This module is responsible for detecting suspicious activities by analyzing the data collected by the DIM. All of the current approaches for CIDSs enable a node to make an intrusion decision on its own when, subjectively, enough data is present to support it. However, the uniqueness of our approach in the local detection module (LDM), as compared to the existing methodologies, is its delegation of further inspection for all suspicious activities to the cooperative detection module (CDM). Such delegation is done even if enough evidence exists for the LDM to make a decision on its own. This is due to the observed weaknesses found in the current approaches where there might be enough evidence, for example, against a node's packet dropping. However, this could be due to accidental drops related to battery power or even increased node mobility that cause a node to gradually move out of range, resulting in such drops. As a result, such legitimate nodes would be falsely marked as malicious in these systems, which degrades detection accuracy and increases false alarms.

To sustain battery life, the methodology followed by the LDM for detecting attacks is unique in an energy saving way. This is achieved by having the detection process operate on an interval basis instead of having it run the entire time, which might result in a costly overhead for limited power devices in MANETs.

This module was implemented as follows: at each time interval (t_l), the LDM sends a message to the DIM requesting the data collected between the current time and previous interval. The DIM then fetches this data from its memory storage (DCT) by selecting records with a packet time between current time and last interval (now – t_l), where (t_l) is the activation interval for LDM. Once the data is received, the LDM starts its analysis against potential attacks.

To investigate black hole attacks against the data received from the DIM, the LDM calculates the number of packets forwarded to each neighbor by counting the packets that are marked as "forwarded to neighbor" along with that neighbor's IP address. Then, the LDM calculates those who were forwarded by that neighbor by counting the packets marked as "forwarded by neighbor" and have the neighbor's IP address. The LDM then performs a comparison between the two. If the number of packets forwarded by a neighbor is less than the amount forwarded to that neighbor and the difference exceeds the predefined threshold for black hole attacks (b_t), the LDM contacts the CDM to perform a collaborative investigation against the suspicious neighbor.

For modification attacks, the LDM inspects the data received from the DIM against such attacks by comparing the checksum of packets forwarded by each neighbor against those forwarded to that neighbor. If any difference is detected and the total packets modified by that neighbor exceed the predefined threshold (m_t), the LDM contacts the CDM to start a collaborative investigation against the suspect.

For rushing attacks, the LDM inspects the data received from the DIM and calculates the total number and average receipt time of RREQ packets received from each IP address. Then, the LDM calculates the total number and average receipt time of all RREQ packets from all nodes in the current dataset retrieved from the DIM. The LDM then compares the two. If one node has the number of received RREQ with a receipt time of "smaller than the overall average" is greater than a predefined threshold (r_t), the LDM activates the CDM to launch a collaborative investigation.

For flooding attacks, the LDM counts the total number of packets received from a source node based on the data received from the DIM. The LDM then calculates the average payload size received from each node and the average payload size received from all nodes. If the total received packets from one node has an average payload size greater than that received from other nodes and is greater than the specified threshold (f_t), the LDM activates the CDM for further investigation.

As for gray hole attacks, the LDM can detect gray hole attacks in which a malicious node selectively drops packets. In this case, from the data received from the DIM, the LDM counts the total number of packet drops by each neighbor by applying the same calculations described in the black hole attack detection above. If a node has a number of packet drops but the total number of drops does not exceed the threshold for black hole attacks (b_t), the LDM contacts the DIM to retrieve data where packet time is between current time and ($n_p * t_l$) and where (n_p) represents the number of past LDM intervals. For instance, if $n_p = 2$ and $t_l = 15$ seconds, this means that the LDM would retrieve packets with a receipt time between current time and the past thirty seconds (current time – 2 * 15). Upon receiving the data from the DIM, the LDM then counts the number of packet drops belonging to the suspicious node, across the collected data to compare the total packet drops of that node over time. If the rate of packet drops is higher than the threshold (g_t), the CDM is contacted to confirm the suspicion cooperatively.

For all the above-mentioned attack detections in the LDM, activation of the CDM was implemented by sending a message to the CDM containing the IP address of the suspicious node along with the name of the suspected attack. For instance, if the LDM found that packet dropping from node (A) has crossed the predefined threshold for black hole attacks (b_t), the activation message for the CDM would contain the IP address of node (A) along with the attack type as "black hole." The same applies to the other attack types except the attack type content of the message would vary based on the detected attack (e.g., modification, rushing, flooding, etc.)

Another responsibility of the LDM is providing calculation results regarding the observed maliciousness of a certain node, when it receives a request from the CDM. The request message contains the IP address of the suspicious node and the type of suspected attack. In return, the LDM must return a reply message to the CDM containing observations regarding whether the suspect is malicious or non-malicious, along with uncertainty of the LDM regarding its observations. The LDM assigns a probability value for malicious, non-malicious, and uncertainty towards the suspect node based on its observations as explained in more details below. The LDM performs such calculations based on the received “attack type” in the CDM message.

One issue that remains open in the current research is the assignment of probabilities towards proving or refuting the designated hypotheses based on the collected evidence. This process is known as basic probability assignment (BPA) in DST. This is primarily due to the fact that Shafer (1976) did not specify a concrete methodology to assign probability values to a hypothesis based on the collected evidence. Shafer (1976) explained that assignment of probability values to a mutually exclusive set of hypotheses is subjective to the problem domain. That is, the more evidence we have supporting a hypothesis, the closer the assigned weight is moved to one. At the same time, lower evidence in a hypothesis moves the assigned weight closer to zero. If no evidence at all is found supporting a specific hypothesis, then the value is set to zero. A hypothesis is defined here as any subset of the mutually exclusive possibilities to which we can assign a value based on the collected evidence. On the other hand, evidence in this context refers to signs found in the collected data that support of refute the hypothesis. Weight is defined as the strength of evidence in supporting a hypothesis (Shafer, 1976).

Up until now, there is no clear approach on how to automatically calculate BPAs supporting/refuting a system’s hypotheses (Jiang, Zhan, Zhou, & Li, 2016). Previous studies opted to assign weight to evidence based on subjectively-assigned trust values of the sender. For instance, Boston (2000) used manual assignment of assigning probabilities through human expert opinion. Another study by (Siaterlis & Maglaris, 2004) implemented such assignment based on having a system administrator examine previously conducted experiments on the system and subjectively assign these probabilities.

In their research towards an automatic and adaptive way to calculate BPA values for their IDS for fixed wireless networks, Aparicio-Navarro, Kyriakopoulos, and Parish (2012) found that no previous research has suggested this type of BPA calculation. As such, they have come up with a novel methodology for BPAs that can automatically assign probabilities without the need for human intervention. In their approach, a maximum limit of 50% is assigned to their “attack” hypothesis to denote the presence of attacks in the network. The other 50% is assigned to a “normal” hypothesis representing an attack-free network. Lastly, uncertainty is calculated by normalizing the smaller of the two hypotheses by 50% and dividing it by the other. The system assigns probabilities by applying a predefined set of calculations for both normal and attack hypotheses throughout the detection process. Although these detection calculations were designed for fixed networks and don’t apply to MANETs, their automatic BPA approach was designed to be adaptable to all types of wireless technologies (Aparicio-Navarro et al., 2012).

Thus, in this research, we have followed the Aparicio-Navarro et al. (2012) methodology as it represents a low-cost automatic method for BPA calculations. Based on their methodology, a maximum limit of 50% was assigned to each of the malicious and non-malicious hypotheses. Additionally, uncertainty was calculated by multiplying the smaller value of malicious and non-malicious hypotheses by 50% and dividing it by the other value. However, since each attack is designated by certain characteristics, it was not feasible to use one calculation for all attacks. In our approach, each type of attack was designated a separate set of BPA calculations relative to its predefined threshold, as can be seen in Table 3 below.

To demonstrate with a simple example, let’s assume the LDM receives a message from the CDM stating that node (M) is suspected to have attempted a black hole attack. The LDM then sends a message to the DIM requesting the data collected between the current time and previous LDM interval. The DIM then fetches this data from its memory storage (DCT) by selecting entries with a packet time between current time and the last interval (now – t_l) and where (t_l) is the activation interval for LDM. Upon receiving the data, the LDM starts the calculations of the requested values as follows: Let's assume the predefined threshold for black hole attacks (b_t) is set to ten. Additionally, let's assume that through inspecting the data received from the DIM, the LDM found that the total dropped packets observed from the suspicious node (M) equals to eight out of the twenty packets forwarded to that node. On the other hand, the received data from the DIM shows that the suspicious node did not drop three packets forwarded to that node out of the twenty packets forwarded to that node. This could be tricky as these packets could have been destined to that suspect node. Based on these two values, the LDM calculates the requested BPAs according to the DST as follows:

Let LDM(M) represent the observations of the local LDM towards the suspicious node (M).
Let Ω be the frame of discernment, which represents a set of mutually exclusive possibilities (Shafer, 1976). In our case, Ω consists of two possibilities regarding node (M): $\Omega = \{P, \overline{P}\}$, where P=non-malicious and $\overline{P}$=malicious. For this Ω, we have three focal elements: hypothesis $S1=\{P\}$ stating that node (M) is non-malicious, hypothesis $S2=\{\overline{P}\}$ stating that node (M) is malicious, and hypothesis $U = \Omega$ representing uncertainty by stating that node (M) is either malicious or non-malicious. A hypothesis in this context refers to any subset of Ω for which the LDM can present evidence. Additionally, Shafer (1976) stated that the sum of all focal elements must equal to one.

The LDM calculates these values as follows:

$$LDM(M)_{S1} = \frac{total\ packets\ forwarded\ by\ suspect}{total\ packets\ forwarded\ to\ suspect} * 0.5 = \frac{3}{20} * 0.5 = 0.075$$

$$LDM(M)_{S2} = \frac{total\ packets\ dropped\ by\ suspect}{\mathrm{b_t}} * 0.5 = \frac{8}{10} * 0.5 = 0.4$$

The above calculations are multiplied by “0.5” to limit the value of malicious and non-malicious assignments to 50% as discussed previously. To calculate $(M)_{S3}$, based on the work by Aparicio-Navarro et al. (2012), the smaller value of $LDM(M)_{S1}$ and $LDM(M)_{S2}$ is normalized by 50% and divided by the other. In this case:

$$LDM(M)_{S3} = \frac{0.5 * LDM(M)_{S1}}{LDM(M)_{S2}} = \frac{0.5 * 0.075}{0.4} = 0.09375$$

According to Shafer (1976), the summation of focal elements ($LDM(M)_{S1}, LDM(M)_{S2}$, and $LDM(M)_{S3}$) should always equal to one. However, the summation of the above values dos not equal to one. As such, to avoid falling into cases where the summation is greater or less than one, we are going to apply an adjustment value (μ), as proposed by Aparicio-Navarro et al. (2012). This value is calculated as follows:

$$\mu = \frac{(LDM(M)_{S1} + LDM(M)_{S2} + LDM(M)_{S3}) - 1}{3}$$

$$= \frac{(0.075 + 0.4 + 0.09375) - 1}{3}$$

$$= -0.14375$$

To obtain the final values for $LDM(M)_{S1u}$, $LDM(M)_{S2u}$, and $LDM(M)_{S3u}$, we subtract (μ) from each of them. Thus, the final values would be (summation of all equals to one):

$$LDM(M)_{S1u} = LDM(M)_{S1} - \mu$$

$$= 0.075 - (-0.14375)$$

$$= 0.21875$$

$$LDM(M)_{S2u} = LDM(M)_{S2} - \mu$$

$$= 0.4 - (-0.14375)$$

$$= 0.54375$$

$$LDM(M)_{S3u} = LDM(M)_{S3} - \mu$$

$$= 0.09375 - (-0.14375)$$

$$= 0.2375$$

Comparing the results from this example with the gathered evidence that shows a high number of packet drops as compared to the predefined threshold (b_t) from node (M), the above calculations show that the LDM has more evidence supporting the maliciousness of node (M), but still comes with uncertainty. However, the final decision can only be made by the CDM when it combines all evidence from other nodes while regarding uncertainty to account for accidental packet drops.

After performing the above calculations, the LDM sends a message back to the CDM containing the IP address of the suspected node, the suspected attack, LDM(M)$_{S1u,}$ LDM(M)$_{S2u,}$ and LDM(M)$_{S3u}$. The CDM then uses this reply to participate in the cooperative investigation. Further DST calculations along with a demonstration of how such observations were used in the CDM for the intrusion detection process can be seen in the next section.

Based on the outlined mechanisms of the LDM to detect malicious attacks, Table 3 below lists the necessary calculations performed by the LDM to calculate BPA values for each attack. For the sake of readability, the suspect node is referred to as (M) in the below table. For all LDM calculations, based on the work by Aparicio-Navarro et al. (2012), uncertainty was calculated by multiplying the smaller value of $LDM(M)_{S1}$ and $LDM(M)_{S2}$ by 50% and dividing it by the other value, as seen in the above example. Overall average in these calculations refers to the corresponding average of all nodes.

Attack Type	$LDM(M)_{S1}$	$LDM(M)_{S2}$
Black hole	$\frac{total\ packets\ forwarded\ by\ suspect}{total\ packets\ forwarded\ to\ suspect}$	$\frac{total\ packets\ dropped\ by\ suspect}{b_t}$
Modification	$\frac{total\ packets\ unmodified\ \ by\ suspect}{total\ packets\ forwarded\ to\ suspect}$	$\frac{total\ packets\ modified\ by\ suspect}{m_t}$

Rushing	$\frac{total\ routing\ packets\ received\ from\ suspect\ with\ receipt\ time\ equal\ to\ the\ overall\ average}{total\ routing\ packets\ received\ from\ suspect}$	$\frac{total\ routing\ packets\ received\ from\ suspect\ with\ receipt\ time\ smaller\ than\ overall\ average}{r_t}$
Flooding	$\frac{total\ data\ packets\ received\ from\ suspect\ with\ payload\ size\ equal\ to\ the\ overall\ average}{total\ data\ packets\ received\ from\ suspect}$	$\frac{total\ data\ packets\ received\ from\ suspect\ with\ payload\ size\ greater\ than\ the\ overall\ average}{f_t}$
Gray hole	$\frac{total\ packets\ forwarded\ by\ suspect\ accross\ the\ requested\ intervals}{total\ packets\ forwarded\ to\ suspect\ across\ the\ requested\ intervals}$	$\frac{total\ packets\ dopped\ by\ suspect\ accross\ the\ requested\ intervals}{g_t}$

Table 3 - BPA Calculations in the LDM

Cooperative Detection Module

This module (CDM) is activated only upon receiving a request message from the LDM. The request message contains the IP address of the suspicious node as well as the type of the predicted attack. When such a request is received, this module contacts other nodes in the social circle to launch a collaborative investigation to prove/disprove the raised suspicion. Participants of the collaborative investigations are determined through querying the SST to pick only strong-ties. This collaborative investigation relies on the decisions made by each neighbor’s LDM. However, sometimes, some of these strongly tied nodes might not have collected enough data during the interval to form an opinion against the suspicious node. It’s important to note that CDM does not consider opinions from weak ties (nodes that are not strong ties) or from malicious nodes. Some studies have utilized majority-voting techniques for collaborative decision making (Manikopoulos & Ling, 2003; Sen et al., 2008; Mahmood et al., 2009). However, such approaches might lead to unreliable results in the presence of a large number of malicious nodes (Chen & Venkataramanan, 2005). As such, a methodology is needed to weigh in the received opinions based on a certain degree of belief. Thus, this module utilizes DST to do so.

The primary benefit of applying DST to CDM's operations is to eliminate inaccurate detection decisions against accidental activities, such as packet dropping, in which a legitimate node may be falsely accused to be malicious. However, the fact is that such packet dropping can happen for other non-malicious reasons, such as collisions and low battery power. Through the application of DST, the CDM is able to make a decision based on different received opinions and with uncertainty towards the observed behaviors in mind. A unique quality of our application of DST is the consideration of cases where some nodes would only have partial evidence against an incident. This, in turn, would significantly decrease false accusations against accidental non-malicious cases.

We used DST against observations obtained solely from strong ties. This was done to handle cases where some of these nodes did not catch enough evidence against the suspicious activity. To illustrate the intended calculations, let's assume that node (K) requests observations, through CDM, from its strong-tied neighbors (A, B, and C) against node (M), which node (A) doubts as malicious due to observed packet dropping during a period (t_x). However, such packet dropping occurred due to the increased mobility of node (M) away from its neighbors. At the same time, node (B) and (C) did not catch enough data regarding dropped packets from node (M) to form a solid observation. The observations received from all nodes represent two possibilities against the suspected node: malicious and non-malicious. The following illustrates the intended calculations through DST in this case:

Let A(M), B(M), and C(M) represent the observations of A, B, and C respectively towards the suspicious node (M).

Let Ω be the frame of discernment which represents a set of mutually exclusive possibilities (Shafer, 1976). In our case, Ω consists of two possibilities regarding node (M): $\Omega = \{P, \overline{P}\}$, where P=non-malicious and $\overline{P}$=malicious. For this Ω, we have three focal elements: hypothesis S1={P} stating that node (M) is non-malicious, hypothesis S2=$\{\overline{P}\}$ stating that node (M) is malicious, and hypothesis U = Ω representing uncertainty by stating that node (M) is either malicious or non-malicious. A hypothesis in this context refers to any subset of Ω for which nodes (A), (B), and (C) can present evidence.

The following illustrates hypothetical observations from nodes (A), (B), and (C) with each of the three nodes having a probability assignment, as follows:

Observations from Node A	Observations from Node B	Observations from Node C
$A(M)_{S1} = 0.4$ $A(M)_{S2} = 0.1$ $A(M)_{S3} = 0.5$	$B(M)_{S1} = 0.1$ $B(M)_{S2} = 0.5$ $B(M)_{S3} = 0.4$	$C(M)_{S1} = 0.4$ $C(M)_{S2} = 0$ $C(M)_{S3} = 0.6$

Based on the collected observations from Nodes (A), (B), and (C), we apply Dempster's rule of combination (Shafer, 1976) to combine the observations of all three nodes. The result of this combination represents the final decision regarding the maliciousness of node (M).

First, we calculate the normalizing constant (K_{AB}) which represents the extent of conflict between the observations from node (A) and (B):

$$K_{AB} = A(M)_{S1}\ B(M)_{S1} + A(M)_{S1}\ B(M)_{S3} + A(M)_{S3}\ B(M)_{S1} + A(M)_{S2}\ B(M)_{S2} + A(M)_{S2}\ B(M)_{S3} + A(M)_{S3}\ B(M)_{S2} + A(M)_{S3}\ B(M)_{S3} = 0.79$$

Then, we combine the beliefs from node (A) and (B):

$$AB(M)_{S1} = A(M)_{S1} \oplus B(M)_{S1} = \frac{A(M)_{S1}\, B(M)_{S1} + A(M)_{S1}\, B(M)_{S3} + A(M)_{S3}\, B(M)_{S1}}{K_{AB}}$$

$$= 0.3164$$

$$AB(M)_{S2} = A(M)_{S2} \oplus B(M)_{S2} = \frac{A(M)_{S2}\, B(M)_{S2} + A(M)_{S2}\, B(M)_{S3} + A(M)_{S3}\, B(M)_{S2}}{K_{AB}}$$

$$= 0.4303$$

$$AB(M)_{S3} = A(M)_{S3} \oplus B(M)_{S3} = \frac{A(M)_{S3}\, B(M)_{S3}}{K_{AB}}$$

$$= 0.2531$$

After that, we calculate the normalizing constant (K_{ABC}) representing the conflict between the combined beliefs of nodes (A) and (B) with those of (C):

$$K_{ABC} = AB(M)_{S1}\ C(M)_{S1} + AB(M)_{S1}\ C(M)_{S3} + AB(M)_{S3}\ C(M)_{S1} + AB(M)_{S2}\ C(M)_{S2} + AB(M)_{S2}\ C(M)_{S3} + AB(M)_{S3}\ C(M)_{S2}$$

$$+ AB(M)_{S3}\ C(M)_{S3}$$

$$= 0.827$$

Finally, to get the final decision regarding the maliciousness of node (M), we combine AB(M)$_{S2}$ with C(M)$_{S2}$:

$$ABC(M)_{S2} = AB(M)_{S2} \oplus C(M)_{S2} =$$

$$\frac{AB(M)_{S2}\, C(M)_{S2} + AB(M)_{S2}\, C(M)_{S3} + AB(M)_{S3}\, C(M)_{S2}}{K_{ABC}}$$

$$= 0.311$$

Examining the results, we notice that even though node (B) had strong evidence supporting the suspicious activity from the suspected node (M). The insufficient evidence from nodes (A) and (C) saved (M) from falsely being marked as malicious since its packet dropping was accidental due to its mobility pattern. Compared to other methodologies, such as the Bayesian theorem, node (M) would've been falsely accused as malicious. Thus, it would have been isolated from the network and false alarm rates for similar situations would increase.

From the results of the example above, we notice how the application of DST can help reduce false accusations when partial or no evidence exists against a suspected node. This is useful considering such cases can very well happen when a node recently joined the network or when no evidence can be constructed due to the lack of sufficient collection intervals. On the other hand, the partial evidence scenario might occur in cases where the analyzed suspicious activity has not crossed the predefined threshold for the suspected attack. Since DST does not regard uncertainty as negative evidence, taking these scenarios in consideration has a significant impact on decreasing false alarms and increasing detection accuracy of the overall system in the long run.

The implementation of the CDM was as follows: upon receiving a collaborative investigation request message from the LDM, the CDM requests a list of IP addresses of strong-ties from the STBM. The STBM then queries its own STT to retrieve these addresses, and it passes them back to the CDM. When the CDM receives these addresses, it formulates a message containing a unique request identification (RID) number, the IP address of the suspicious node, and the type of the suspected attack extracted from the message received from the LDM (e.g., black hole). The CDM then sends the formulated message to each strong tie node in the list obtained from the STBM. The formulated message is then saved to the CDM's memory storage represented by a table of CDM requests and their corresponding received replies, called a collaborative information table (CIT). At this point the CDM investigation is still open, but no further processing is required until replies are received from the strong ties.

When the CDM receives a reply, it correlates it with the sent request through the RID number. The CDM reply message contains the following information: the replying node's IP address, the RID, the suspected attack type, the suspicious node's IP address, the probability of the suspect being malicious, the probability of the suspect being non-malicious, and the probability of the suspect being either malicious or non-malicious (uncertainty). The reply is then saved to the CIT. After that, the CDM checks if the number of replies to that particular request equals to 80% of the number of strong-ties that have received the request. The reason the researcher is considering 80% is to account for mobility and collisions in the network that might cause the CDM request to get lost in the network and not be received by some of the strong-ties. If the CDM has indeed received replies from 80% of the destination strong-ties, it combines the received observations through applying the DST calculations outlined above. Based on the result of these calculations, the CDM makes the final decision regarding the maliciousness of the suspicious node. If the node is found malicious, the CDM sends a message to the GRM containing the IP address of that node. The CDM then marks the request as complete in the CIT.

On the other hand, if the CDM receives a request for collaborative investigation, it first checks to see if the request originated from a strong tie node. This is done by sending a message containing the IP address of the request node to the STBM. The STBM, in turn, searches its STT to see if it can find that IP address and to determine if it's marked as a "strong-tie". The STBM then sends a message back to the CDM confirming whether the IP address belongs to a strong tie or not. If the STBM message states the originating node is not a strong tie, the CDM ignore the request. Otherwise, the CDM extracts the RID, the IP address of the suspect node, and the suspected attack and passes the message to the LDM. Based on the suspected attack, the LDM performs its calculations (outlined in the previous section) and returns the results to the CDM. After that, the CDM formulates and sends a reply message to the originating node containing the RID, the IP address of the suspect node, the probability of the suspect being malicious, the probability of the suspect being non-malicious, and the probability of the suspect being either malicious or non-malicious (uncertainty).

Global Response Module

The global response module (GRM) is responsible for taking responsive actions whenever an intrusion is detected. The responsive actions in this system work by sending an alarm message to a node's social circle and removing the detected malicious node from the routing table. When an attack is detected through the CDM, the GRM gets activated through a message containing the IP address of the malicious node. The first responsive action by this module is to remove the malicious node from the routing table to isolate it from all future communications. After that, the GRM sends an alarm message to all nodes in its social community, aka strongly-tied nodes. Each receiving node would then forward the alarm message to their social circle. That way the malicious node would be known even to remote nodes, and with time, information convergence would result in the elimination of the malicious nodes from the routing process. This assumption is backed by Granovetter's (1973) suggestion that diffusion of rumors between strongly-tied friends, utilizing weak ties as a crossing bridge, can reach a large number of entities in a larger social distance.

On the receiving side, to eliminate fake reports generated from malicious nodes against legitimate ones, when a node receives an intrusion alarm message, the GRM checks whether the source address is a strong-tie. If so, the reported node is removed from the routing table, and the alarm is forwarded to all nodes in the social circle of the receiving node. Otherwise, the alarm is discarded and no responsive actions would be taken.

The implementation of this module was as follows: whenever the GRM receives a message from the CDM, it contacts the STBM to retrieve a list of IP addresses of strong ties. The STBM then queries its own STT to retrieve these addresses and passes them back to the GRM. After that, the GRM creates a new message containing the IP address of the malicious node detected by the CDM and sends it to every IP address in the list returned from the STBM. On the other hand, when a node receives an alarm message, the GRM sends a message containing the IP address of the originating node to the STBM to check whether it belongs to a strong tie or not. The STBM, in turn, searches its STT to see if it can find that IP address and if it's marked as a "strong tie". The STBM sends a message back to the GRM confirming whether the IP address belongs to a strong tie or not. If the STBM message denies that the originating node is a strong tie, the GRM drops the alarm packet. However, if it is a strong tie, the GRM sends a request to the STBM to retrieve the IP addresses of strong ties. Once the GRM receives this list, it forwards the alarm message to all IP addresses in that list.

As a reactive action to the received alarm, the GRM deletes all entries of the malicious node in the routing table. Additionally, to avoid including the malicious node in future routes, when an IDS node receives a RREQ or RREP packets, as mentioned in the DIM section, it sends the source IP address to the STBM to check if the node is marked as malicious. The STBM then searches the STT for the source IP address and returns with a response that indicates whether it is marked as "malicious" or "non-malicious." The DIM then drops the packet if the response from the STBM confirms that the originating node is malicious. Otherwise, the DIM forwards the packet down the route. By dropping not only data packets from malicious nodes but also routing packets, the proposed IDS does not add any entries to the routing table for these nodes. As the alarm messages propagate among strongly-tied communities, malicious nodes eventually end up isolated and cannot communicate with these communities.

Data Collection Procedures

The implementation of the proposed IDS required extensive data collection operations throughout the testing phase until the collection of the final results of the experiment. The researcher performed the following three major phases for data collection:

- Normal network operations: throughout this phase, the researcher implemented a testing MANET comprised of a number of legitimate nodes communicating with each other over the AODV routing protocol. Details of the testing MANET are presented in the next sections. The researcher monitored the normal network operations with regards to routing, communications between nodes, and topological changes from nodes' mobility.

Continuous data collection operations were performed throughout the lifetime of the test network. The result of such operations was a collection of trace files containing various routing/data communication packets along with nodes' positions at each point of time during testing. The researcher then used these files to analyze and document communications and mobility patterns in an attack-free network. Additionally, this data was used to document the network performance without the existence of IDS nodes. This was then used to evaluate the performance impact of the proposed IDS on normal network operations, as discussed later in this section.

- Network under attacks: in this phase, the researcher deployed single and multiple attacker nodes in the same testing MANET. These attacker nodes targeted the disruption of the normal network operations of legitimate nodes. Attacker nodes launched variations of black hole, gray hole, modification, rushing, and flooding attacks. Each of these attacks had an adverse impact on communications between nodes and the survivability of the network in general. The researcher continuously collected data throughout the various attack scenarios in the form of trace files. These files were then used to analyze and determine the required attack thresholds defined for the detection components of the proposed IDS. In this context, the term attack threshold refers to a certain base criterion the proposed IDS will use to make intrusion detection decisions. Usage details of each attack threshold were previously explained in the "system design" section.
- Network with IDS nodes: after following the implementation steps described in the "system design" section, the proposed IDS was ready to start operating in a simulated network. Thus, the researcher used the same network configuration for collecting normal network operation parameters for this data collection phase. The testing MANET had an

IDS installed on each node. Nodes followed the same routing, data communications, and mobility patterns that would occur if the IDS module was not installed. The researcher monitored these activities along with any impact imposed by the installation of the proposed IDS. Continuous data collection operations took place in the form of trace files to collect routing/data communications and mobility parameters. These parameters were then used to evaluate the proposed IDS's impact on the network performance. This evaluation was achieved by examining this data with that collected data from the normal network operation phase as will be discussed in Chapter 4.

- Network with IDS and malicious nodes: this represented the final data collection phase in which both IDS and attacker nodes were installed on nodes in the network. The same testing MANET used in the normal network operations phase was used here. First, all nodes had the proposed IDS installed on them. Then, the researcher randomly-designated certain nodes to uninstall the IDS and install attack modules instead. Variations of single and multiple attackers were simulated in the network. The researcher continuously collected data from the network for evaluating the proposed IDS along with the adverse effects of the attacker nodes on the network. Evaluation parameters that were collected are described in the next section. Additionally, the impact of attacker nodes on routing/data communications and the lifetime of the network was collected. The collected data was then analyzed to evaluate the feasibility of the proposed IDS in achieving the research goals of attaining a high detection accuracy and minimizing communication overhead. The researcher followed exhaustive data analysis operations for such evaluations. Details of such evaluations are discussed in Chapter 4.

System Evaluation Metrics

To evaluate whether the proposed system achieves a high detection accuracy rate along with minimal communication overhead, the following evaluation metrics were collected throughout the life of the experiment. These metrics are commonly used by similar studies to evaluate the feasibility and performance of IDS implementations for MANETs (Manikopoulos & Ling, 2003; Kareem et al., 2006; Sen et al., 2008; Zhang et al., 2016).

- False Positives (FP): this represents the number of innocent nodes falsely identified by the system as malicious.
- False Negatives (FN): this denotes the number of malicious nodes incorrectly identified by the system as innocent.
- True Positives (TP): this denotes the number of malicious nodes correctly identified by the system as malicious.
- True Negatives (TN): This represents the number of innocent nodes correctly identified by the system as innocent.
- Detection Rate (DTR): this is calculated as follows:

$$DTR = \frac{TP}{TP + FN}$$

- False Positives Rate (FPR): this is calculated as follows:

$$FPR = \frac{FP}{TN + FP}$$

- Average End-to-End Delay (AED): this represents the average delay encountered to deliver packets from a source to a destination, and it is calculated as follows:

$$AED = \frac{TD}{PR}$$

Here, TD is defined as the time spent to deliver packets while PR represents the total number of packets received.

FPR and DTR were used to demonstrate the proposed system's achievement of the first goal of this research, which is "high detection accuracy." On the other hand, AED was measured to determine if the system met the second goal of this research, represented in minimizing communication overhead throughout the detection operations of the proposed IDS. The usage of AED targeted measuring the effect of eliminating regular dissemination of detection packets on delays and successful deliveries in the network. In current CIDS, detection-related packets are disseminated on a regular-basis. This, in turn, incurs extra delays in the network as such a dissemination could potentially cause collisions, and eventually packet loss. AED here measures the bandwidth consumption in a sense that lower AED values means less packet loss due to collisions caused by regular information dissemination, thus less delays.

Alongside, the proposed system was evaluated against a stand-alone installation of itself. To perform such an installation, the system was modified to remove the cooperative components. As such, the system relied solely on the LDM to make intrusion detection decisions. The LDM was modified to perform the final decision calculations. This comparison was intended to evaluate the efficiency of the cooperative detection nature of the proposed system as compared to a stand-alone intrusion detection.

Furthermore, the system was modified to have the DIM disseminate detection related information to all strong ties of the IDS node at each collection interval. This modification was intended to evaluate the efficiency of the proposed system in minimizing communication overhead as compared to regular dissemination of detection information found in the current

CIDSs. The researcher then compared the proposed system to the modified version with regards to bandwidth overhead. Results of the aforementioned comparisons are discussed in Chapter 4.

Test MANET Simulation Parameters

Table 4 shows the various parameters used for simulating the test MANET throughout the experiment. Previous studies have followed similar parameters in their simulation procedures (Kareem et al., 2006; Sen et al., 2008; Shao, Lin, & Lee, 2010; Tabari et al., 2012; Veeraiah & Krishna, 2018).

.

Parameter	Value
Topology area	1000 m x 1000 m
MAC Protocol	IEEE 802.11
Routing Protocol	AODV
Mobility model	Random Waypoint
Transmission range	250 m
Packet type	UDP
Packet size	512 bytes
Minimum node speed	2 m/s
Maximum node speed	10 m/s

Pause time	10 s
Number of nodes	100
Simulation time per iteration	500 s
Total iterations	40

Table 4 - Test MANET Simulation Parameters

The topology area parameter, shown in table 4, was used to define the simulation area boundaries that nodes in the test MANET are allowed to move in. This parameter is defined by Kurkowski, Camp and Colagrosso (2005) as "the square meter area of the topology." As such, 1000 m x 1000 m defines the height and width of the test MANET's area in meters respectively (Kurkowski et al., 2005). As previously outlined, AODV was used as the routing protocol while IEEE 802.11 was used as the MAC protocol commonly used in MANET studies due to its ability to minimize packet collisions (Akai, Martin, & Bagrodia, 2001).

Node speed parameter defines the speed that nodes travel around the test MANET. The mobility model parameter defines the movement pattern for nodes in the test MANET. Random waypoint is a mobility pattern implemented in ns-2 that allows each node to choose and move towards a random destination at each instant at the specified node speed. Nodes then pause for a period specified in the pause time parameter and then move towards another destination (Bai, Sadagopan, & Helmy, 2003). The transmission range parameter specifies the maximum distance that two nodes in the test MANET can be from one another to transmit data to each other (Gomez & Campbell, 2007). The total iterations parameter specifies the total iterations executed in the implementation of this research study targeting the evaluation of the proposed system against the research goals. Details of these iterations are discussed in Chapter 4.

Formats for Presenting Results

Data resulting from the experiment was presented in tables and line graphs. The tables were used to summarize the outlined evaluation metrics that were used to evaluate the system's feasibility to achieve the research goals. Line graphs were utilized to visualize the simulation results throughout the experiment and illustrate the performance of the proposed system.

Resource Requirements

The implementation of the proposed system was executed through ns-2. Simulation of the performance and efficiency of the proposed system was done through ns-2 as well, against different attacks scenarios. Since ns-2 implementations usually consume a fair amount of computing resources for compiling and running simulation packages, the proposed system was implemented in a researcher-prepared environment with the following technical specifications:

- Computer Hardware:
    - Processor: Intel 2.6 GHZ Coe i7
    - Memory: 16GB
    - Disk: 200GB
- Operating System: Mac OS 10.12 (Free BSD based)
- Implementation Platform: ns-2.35

All of the implementation steps and testing were executed by the researcher. No changes in the implementation environment were done during the implementation of the proposed system.

Summary

This research followed an experimental model to design and implement the proposed IDS. This chapter explained the approach that was followed to implement this research study. The novel approach this research took in applying the concept of social communities along with DST and how this approach enabled the system to achieve the research goals were also explained. Conceptual and technical details of the system implementation were provided as well, and descriptions of the various components of the proposed IDS were provided alongside a detailed explanation of their intended implementation. Attacks generation, data collection, system testing, and evaluation procedures were also explained. Additionally, formats of the results from the experiment along with the required resources were presented.

Chapter 4

Results

Introduction

The primary objective of this research was to develop a CIDS capable of achieving both high detection accuracy and minimized bandwidth consumption throughout the detection process. This chapter details the experiments conducted to evaluate the implementation of the proposed CIDS along with the results achieved. A detailed analysis of the results of each experiment is presented along with quantitative evaluation of results against the achievement of the presented research goals. The chapter concludes with research findings based on the collective analysis of the results obtained throughout the experiments.

Experiment Structure

The results collection phase of this research consisted of running four experiments. The first targeted the identification of attacks and the proposed system's thresholds previously discussed in Chapter 3. The second experiment involved the evaluation of the proposed IDS against a variety of attack scenarios. The third experiment targeted the evaluation of the proposed IDS against a stand-alone IDS implementation, specifically with regard to detection accuracy and false alarms rates. The last and fourth experiment assessed bandwidth consumption throughout the detection process of the proposed IDS against a CIDS implementation that continuously disseminated detection-related information through the network, referred to from here on as CCIDS (continuously-information-disseminating cooperative intrusion detection system). Details of each experiment are presented in the next sections.

Each of the above-mentioned experiments involved running multiple simulation iterations. Throughout the experiment, the researcher implemented the following sequential steps to execute each simulation:

- Attackers Generation: to measure the accuracy of the experiment results under different attack scenarios, the researcher included randomly-selected attack types for each simulation. This was intended to eliminate the possibility that the produced results might contain more inconsistency in the presence of certain attacks than others. Randomization of attackers in the test MANET throughout the various simulation iterations eliminated such potentiality and reduced the chances for the researcher to control the results by including certain attack types more than the others.
- Simulation Execution: in this step, the researcher modified the simulation file to include the randomly-selected attackers. The researcher then proceeded with executing the simulation on the host environment. Throughout the lifetime of the simulation, the researcher continuously observed the execution to ensure no interruptions or early

execution stoppage happened due to memory/CPU utilization problems on the host environment. This was done to eliminate potential partial or incorrect simulation results.

- Results Collection: Once a simulation iteration was done, the researcher moved towards collecting the generated data based on the following categorization:
    - Detection-related Results: The researcher wrote the underlying IDS code to output each final intrusion decision to an output text file. Each line of the contents of the output file included the decision-making node, suspect node, decision (malicious/innocent), and attack type.
    - Bandwidth-related Results: The researcher used the trace files generated by ns-2 to calculate the AED values for each iteration. The researcher did not write any custom code to output bandwidth-related data, as ns-2 does that without the need for any modifications. The trace files were then analyzed to extract the AED value based on the calculations outlined in Chapter 3.
- Evaluation Metrics Calculations: After each iteration, the researcher manually inspected the text file from the "Detection Results" step and calculated the evaluation metrics values based on the calculations outlined in Chapter 3. Those values were documented and are presented in the next sections.
- Results Documentation: the researcher used the following data presentation methods to document the results of the experiments:
    - Tables: these were used to document the results of the evaluation metrics calculations including TP, TN, FP, FN, DTR, and FPR.
    - Graphs: these were used to visually plot the DTR, FPR, and AED calculation results throughout the conducted experiments.

Experiments

Experiment 1 - Thresholds Identification

The first experiment following the research implementation involved identifying attacks and IDS-related thresholds. Prior to testing the actual implementation of the proposed IDS, the researcher ran multiple simulations on the test MANET to calculate attack thresholds. The researcher implemented the attack models as outlined in Chapter 3 then proceeded towards finding a consistent threshold for each attack model. This consisted of simulating five iterations of the test MANET with random nodes launching each type of attack (black hole, gray hole, modification, flooding, and rushing). After consistent thresholds were identified, the researcher ran three more simulations but with mixed attacker types. The goal was to validate the consistency of the identified thresholds in scenarios where a mixture of attacks exists while ensuring such scenarios would not affect the accuracy of the identified thresholds. The thresholds were adjusted manually as iterations went on until the researcher found threshold values that are consistent in all mixed attackers scenarios. The final attack thresholds along with other system thresholds (discussed next) are documented in table 5 below.

Threshold	Value
Recency of communications (r_{rec})	4 s
Reciprocity and shared knowledge (r_{rsh})	16

Time since first communication (r_t)	10 s
Black hole threshold (b_t)	0.9 x total packets received by suspect
Modification threshold (m_t)	0.5 x total packets received by suspect
Rushing threshold (r_t)	2.0 s
Flooding threshold (f_t)	800 bytes
Gray hole threshold (g_t)	0.4 x total packets received by suspect
STBM interval (t)	5 s
LDM interval (t_l)	6 s
Decision threshold (d_{ft})	0.7132
Minimum packets collected (p_c)	20
Minimum number of strong-ties (s_t)	5

Table 5 - Proposed System's Thresholds Values

Data Analysis

To identify the IDS threshold values, discussed in Chapter 3, the researcher simulated the test MANET with both IDS and attacker nodes. The first simulation produced inconsistent results from the different IDS nodes in making the final intrusion detection decisions. Following up by running multiple simulations while monitoring the execution output, the researcher noticed that the inconsistency resulted from the lack of a final detection decision threshold. The researcher has noticed that for IDS nodes to make consistent final intrusion detection decisions, a threshold was needed to define the minimum value for the final DST calculations. When the value stipulated in the threshold is crossed, the suspect node is declared as malicious for the specific attack type. This was referred to as the final decision threshold (d_{ft}).

Such a threshold was critical, specifically when the final intrusion decision indicated the suspect node as malicious. In such cases, before identifying d_{ft}, a node would be considered malicious if the final DST value for maliciousness was equal to or greater than 0.5. However, inspection of the decision output files showed that this resulted in rare cases where highly mobile nodes got mistaken as malicious. The researcher then ran four simulations with a mixture of attackers to identify the accurate value for d_{ft}. By running the final decision threshold identification simulation, and identifying it as 0.7132, the researcher noticed that the rare cases of inaccurate decisions no longer appeared in the results.

To identify the STBM thresholds, the researcher ran the test MANET with no attackers for three iterations. Using the resulting thresholds values, the researcher ran the proposed IDS in a MANET with a mixture of attackers for six iterations. During the implementation of the proposed IDS, the researcher coded the system to write the strong-ties of each IDS node to a text file for verifying the implementation of ties calculations. With each iteration, the researcher inspected the resulting ties file to examine the formulated ties throughout the simulation. The researcher adjusted the mobility of randomly chosen nodes to be higher than others to ensure the accuracy of these thresholds in situations where a node is legitimate but has abandoned its initial position in a social community. With each iteration, the researcher had to manually adjust the thresholds until reaching consistent results with each new iteration. The final threshold values obtained produced accurate and consistent identification of the social ties, based on the description in Chapter 3, regardless of the total number of attackers in the network. These values are documented in table 5 above.

While inspecting the detection decisions output files, the researcher noticed that some nodes were sending CDM replies to requests initiated by their social circles without enough collected information about the suspect node. Even though these cases did not affect the decision accuracy, thanks to the application of DST, their unnecessary replies added extra bandwidth utilization. By inspecting the intrusion decision log files, the researcher found that a detection decision can still retain accuracy while minimizing extra CDM replies by enforcing a minimum packets collected threshold (p_c). This threshold is defined as the minimum packets a strong-tie node has collected on a suspect node.

By using p_c, strong-ties that have very little or no collected packets on a suspect should not send out their replies as they do not affect detection accuracy but only increase bandwidth utilization. By manually adjusting the value of (p_c) over four simulations, the researcher found that assigning a value of 20 to p_c yielded no negative effects on the detection accuracy. This means a node can send out a CDM reply about a suspect node, if and only if it has collected at least 20 packets about the suspect. It's worth mentioning that enforcing such a threshold does not interfere with the lack of evidence calculations for DST.

Primarily, p_c ensures that an IDS node has sufficient data to suspect a node's maliciousness. That data might have strong, little or no evidence supporting the suspicion, which is what is needed for DST. Additionally, p_c ensures that strong-ties outside of the transmission range of a suspect do not weigh in their individual decisions without any observations. Experimental simulations have shown that excluding this threshold results in inaccurate decisions because nodes with no observations at all would respond with a high uncertainty value against the suspect. This, in turn, resulted in cases where malicious nodes were considered as innocent, as such values affect the final DST calculations. These occurrences greatly impacted detection results in cases where more than one node returned high uncertainty due to the lack of any observations against the suspect. On the other hand, after applying p_c, the trace files analysis indicated slight improvement in the AED values. Further analysis of AED values of the proposed IDS is presented in the next sections.

After running the initial simulation iterations, the researcher noticed some nodes had few strong-ties but still were making final intrusion detection decisions. By inspecting the ties output files from each testing iteration, the researcher noticed that results from decisions based on replies from five or more ties scored a higher certainty in the final detection decision value than those with less ties. Based on this observation, the researcher adjusted the CDM requests to check for a minimum number of strong-ties (s_t) before initiating the cooperative detection process. This also resulted in less unnecessary CDM requests in the network and, ultimately, less bandwidth utilization. The researcher then executed four simulations with the new condition applied and noticed results were consistently accurate with the certainty of the final detection decision values at 0.758 and higher.

The last part of the threshold identification phase involved collecting and documenting the AED values of the test MANET without the presence of the proposed system. The researcher ran ten simulations. These started with five attackers in the first iteration and then five more attackers were added for each consecutive iteration. The final iteration had a total of 50 attackers. The researcher then analyzed the resulting trace file from each iteration to extract and document the AED values of the test MANET. Figure 2 below shows the resulting AED values of the test MANET without the implementation of the proposed system, referred to as no-IDS MANET.

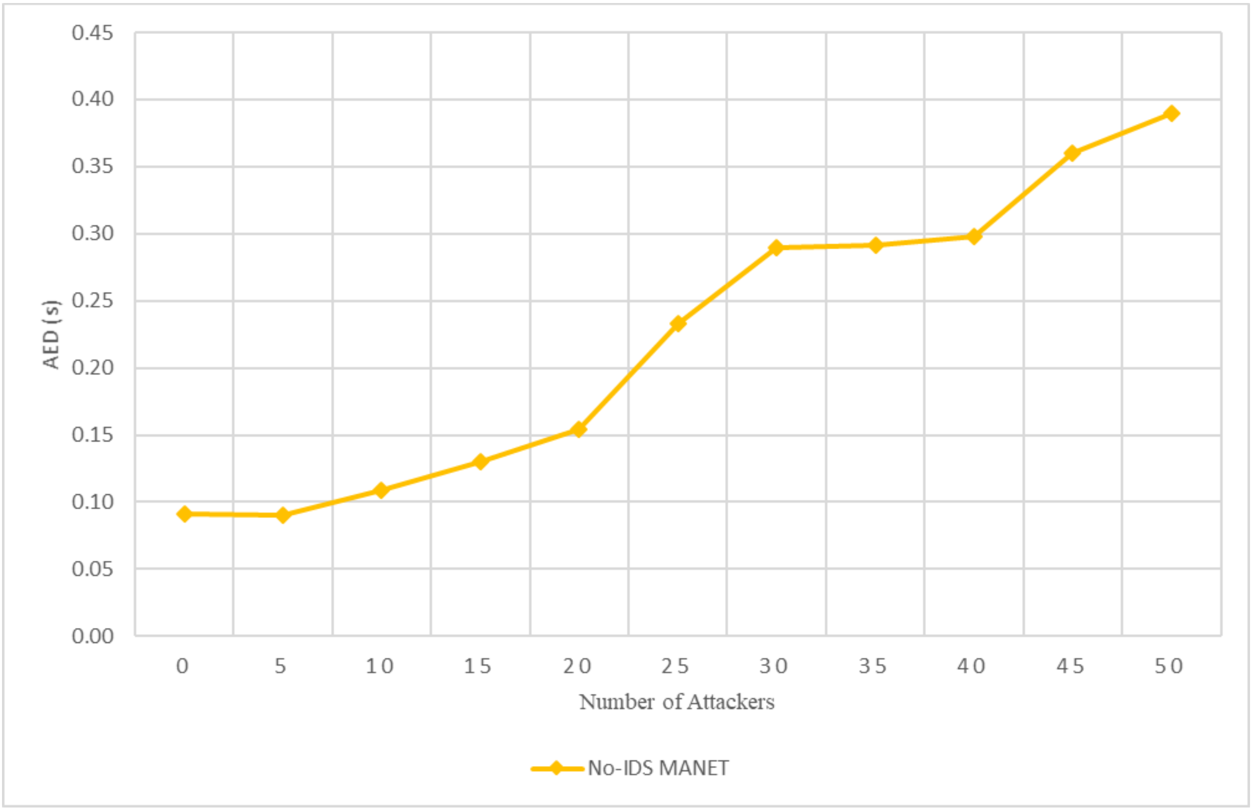

Figure 2 - AED Values of no-IDS MANET

Figure 2 above shows a continuous increase of AED values in the test MANET as more attackers occupy the network. These results were caused by more attackers dropping packets, delaying packet forwarding, and affecting other nodes survivability through exhausting their resources, causing less packets to be delivered successfully to their destination (Wazid, Katal, Sachan, Goudar, & Singh, 2013; Garg & Chand, 2014; Kumar, Vijay, & Suhas, 2016). Additionally, route discovery operations increase as nodes try to find alternative routes to deliver packets successfully to their destination. This adds extra delay in the delivery process until such routes are discovered, if any (Abdelshafy & King, 2013).

Experiment 2 - Performance Evaluation of the Proposed IDS

After establishing the required thresholds for the proposed IDS, the researcher installed the developed IDS module on all nodes in the test MANET. The researcher then ran ten iterations of the same simulation where no attackers exist in the network to measure the consistency of the IDS operations. The researcher then checked the consistency of the decision output files across all iterations. Inspection of such files revealed the expected consistency as the IDS did not issue any alarms in the attack-free network. Once that was established, the researcher included attackers and IDS nodes in the same network. This consisted of first installing the IDS module on all nodes in the test MANET. Then, the researcher picked random nodes and uninstalled the IDS module from each, and then installed a randomly-chosen attack module. The attack module was one of the five attacks identified in the previous chapter (black hole, gray hole, modification, rushing, and flooding).

As outlined in the previous sections, the attack-module type was chosen randomly in each simulation iteration. This was done to eliminate any biases towards inclusions of certain attacks that might be easier to detect than others. This experiment was divided into ten simulation iterations. Each iteration included a number of random attacker nodes, and the rest were IDS nodes. Iterations were identified based on the total number of attackers in the network starting from five attackers, adding five more at each consecutive iteration, and ending with 50 attackers. Each iteration ran for a period of 500 seconds, during which the researcher monitored the execution without intervention. Monitoring the execution of each iteration was necessary to ensure no errors occurred due to hosting environment issues. Ensuring an error-free run for each iteration preserved the accuracy of the results. This was important for the accuracy because such errors, without monitoring, could have potentially stopped the execution of the simulation and resulted in partial results, leading to inaccurate data analysis. After the end of each iteration, the researcher collected the intrusion detection decisions from the decision output files, mentioned in the previous sections. The researcher then calculated the detection evaluation metrics, previously outlined in Chapter 3. The results are documented in table 6 below.

Total attackers	TP	TN	FP	FN	DTR	FPR
5	7	30	2	0	1.00000	0.0625
10	14	56	2	1	0.933333	0.01754
15	19	66	1	2	0.90476	0.014925
20	23	58	4	2	0.92000	0.06451
25	27	57	3	1	0.96428	0.05000
30	51	26	1	3	0.94444	0.03703
35	54	40	1	2	0.96428	0.02439
40	50	30	2	4	0.92592	0.06250

45	70	24	1	3	0.95890	0.04000
50	109	33	2	5	0.95614	0.05714

Table 6 - Evaluation Metrics Values of the Proposed IDS

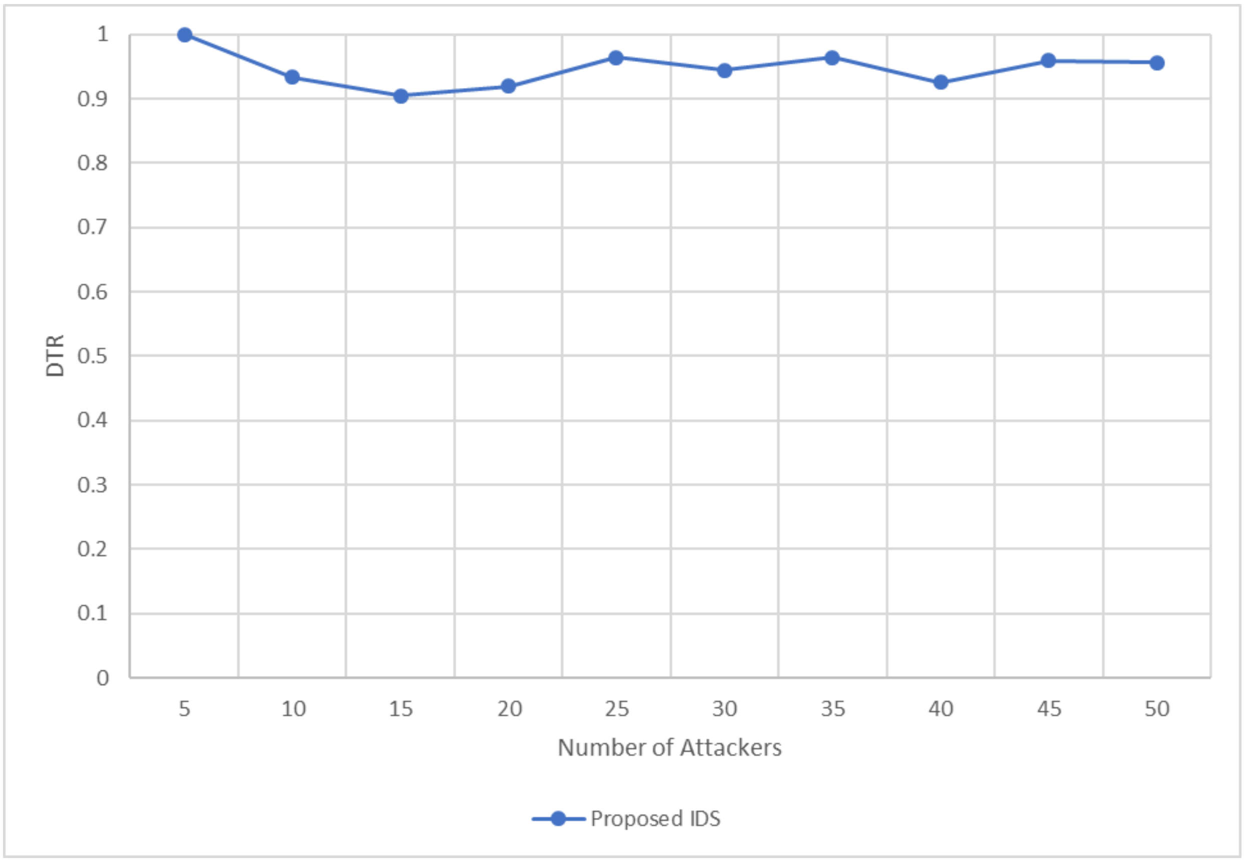

Figure 3 - DTR Values of the Proposed IDS

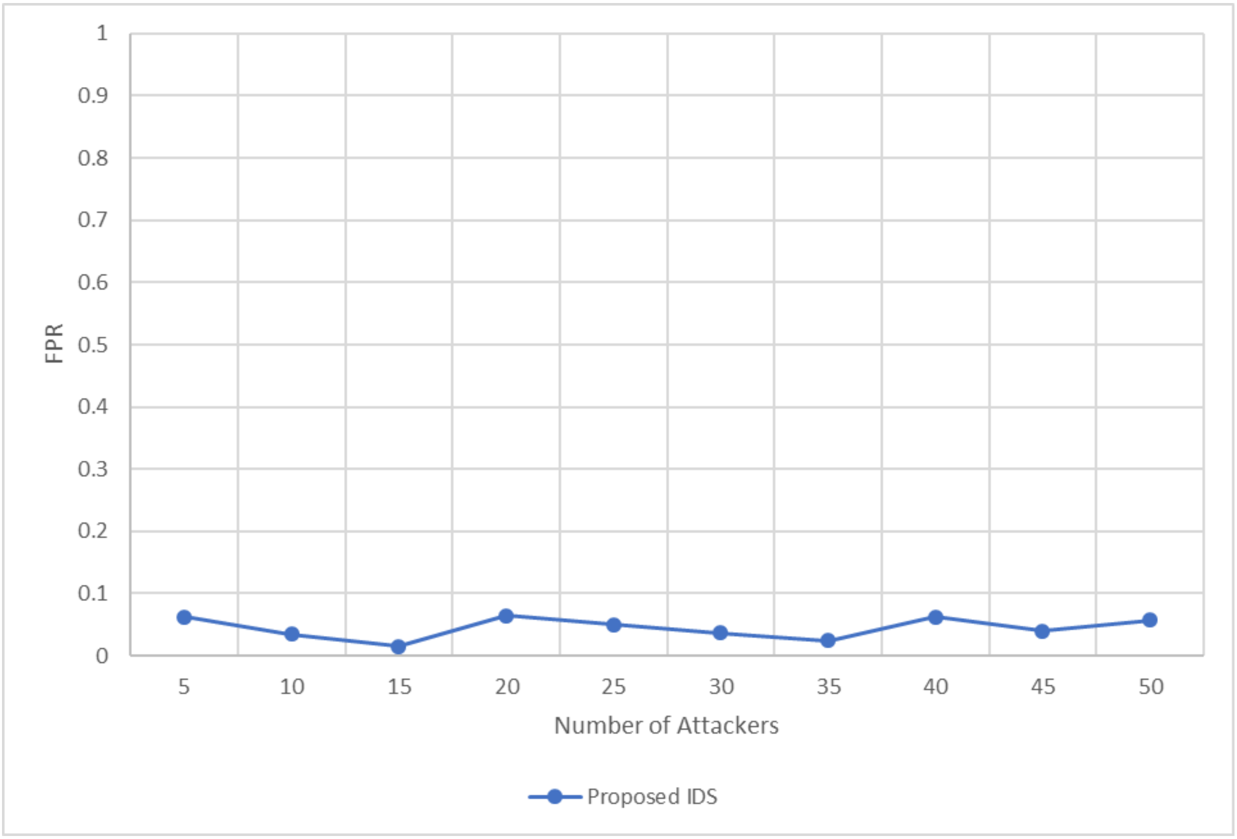

Figure 4 - FPR Values of the Proposed IDS

Data Analysis

Table 6 above shows the incremental increase of the total number of TPs as the number of attackers increase in each iteration. This happened because the first iteration had only five attackers that could potentially break the links between social communities. As such, IDS nodes were able to form a larger number of strong-ties and, eventually, social communities. This resulted in the following advantage: if one node makes the final detection decision towards a malicious attacker, the decision can reach a larger number of nodes in their communities. This meant that if any of the nodes in that certain community encounter the attacker as nodes move around the network, it would not have to go through the detection process again. This enables less bandwidth utilization by eliminating redundant detections. Additionally, it reduces the number of TPs as seen in table 6 above.

On the other hand, as the total number of attackers increase in the network, large social communities start decreasing. This leaves IDS nodes with less total strong-ties and eventually contributes to the formation of smaller social communities. This activity occurs because increased attackers break established connections within social communities. As IDS nodes detect malicious attackers, they remove them from their routing table, leaving the connection to distant nodes potentially broken. This, in turn, results in the STBM removing strong-ties from the STT since they might not satisfy the strong-tie features discussed in Chapter 3. Eventually, the process causes smaller communities to form with potentially no communications to distant communities. This then results in potentially triggering the detection process for the same attacker node(s) multiple times as each smaller community attempts to learn about the status of the malicious node(s).

Additionally, as attackers occupy large portions of the network, communities with less strong-ties might not be able to initiate the detection process. This is because of the established s_t restricting IDS nodes from initiating cooperative investigation when their total strong-ties do not exceed that threshold. Besides, less nodes participating in the detection process means weaker collective evidence established to support the final decision as opposed to that of larger communities. This resulted in the presence of FN values in the output decision files due to an inability of some nodes to collect strong-evidence against suspect nodes. However, the total number of FNs remained low across all iterations with a lowest value of zero and a highest of five when half the network was occupied by malicious nodes. Nonetheless, DTR remained high with a highest value of 1.00000 and a lowest of 0.90476, as shown in figure 3 above.

The above results show a presence of FP values throughout the experiment. However, the proposed system managed to keep the total number of FPs as low as one with a highest value of four. Even when the number of attackers increased, the total number of FPs remained low. This is mainly due to the combined application of social communities and DST. The application of social communities restricted IDS nodes from producing high numbers of FPs as more attackers joined the network. Such an application forced IDS nodes to only consider reports coming from strong-ties. Thus, eliminating situations where false reports coming from malicious nodes in aim to skew the final detection decision are considered.

Working in concert with social communities, the application of the DST contributed to the low total of FPs where IDS nodes incorporated lack/little evidence in the final decision making. This in turn, saved innocent nodes from being accused as malicious due to some of the strong-ties observing abnormal behaviors where as others did not have enough evidence supporting such accusations. Such behaviors might have resulted from high mobility or resource consumptions of the suspect nodes. In these cases, legitimate nodes managed to avoid false accusations, resulting in an overall low total number of FPs across all iterations.

The proposed IDS sustained a high total of TNs across throughout the experiment, with the lowest value being 24 and the highest being 66. The presence of TNs in this experiment resulted from situations where innocent nodes were saved from being falsely accused as malicious due to some IDS nodes observance of abnormal activities. This is due to the cooperative nature of the system solidified by considering observations only from strong-ties as well as enforcing lack of evidence in the final decision making. As opposed to IDS systems where a node gets accused for a certain attack as soon as it crosses a predefined threshold (Nadkarni & Mishra, 2004; Karim, Rajatheva, & Ahmed, 2006; Tangpon, Kesidis, Hsu, &

Hurson, 2009; Lauf, Peters, & Robinson, 2010), the proposed system does not rely solely on the attack thresholds to make such accusations.

Instead, the proposed IDS incorporates multiple restrictions before initiating any accusation against a node. These restrictions include attack thresholds, collected evidence over a period of time, minimum number of strong-ties, and minimum number of packets observed from the suspect node. All these restrictions must be met for an IDS node to initiate an accusation against a suspect node. Even so, the suspect remains innocent until all DST values come back from the initiating node's social circle in which a final detection decision is made. The high total of TNs accompanied by low totals of FPs across all iterations contributed to the low FPR values. The proposed system recorded FPR values as low as 0.01492 and a highest of 0.06451.

Figure 4 above shows how the FPR values for the different iterations do not follow a continuous increase corresponding to the total number of attackers in the network. In current CIDs, a common observation found in the literature is the proportional increase of FPR as more attackers join the network (Mustafa & Xiong, 2013; Ullah, Khan, Ahmed, Javaid & Khan, 2016; Alattar, Sailhan & Bourgeois, 2012; Mahmoud & Shen, 2010). The proposed IDS overcame this problem through the implementation of DST in decision making. As such, the reliance of DST on the collected evidence prevented IDS nodes from making accusations against other nodes without strong evidence. As attackers increase, large communities break into smaller communities leaving potentially less nodes participating in the final decision making. In cases where not enough or no strong evidence exists, these nodes avoid making false accusations against legitimate nodes. This minimizes the total number of FPs as more attackers join the network.

After documenting the detection evaluation metrics, the researcher calculated the AED values based on the corresponding calculation identified in Chapter 3. This involved analyzing each trace file of all simulation iterations to extract these values. Documenting these values was critical to demonstrating the proposed system's ability to fulfill the second goal of this research: minimizing bandwidth consumption throughout cooperative detection operations. Figure 5 below shows these values along with the corresponding AED values of no-IDS MANET from experiment 1.

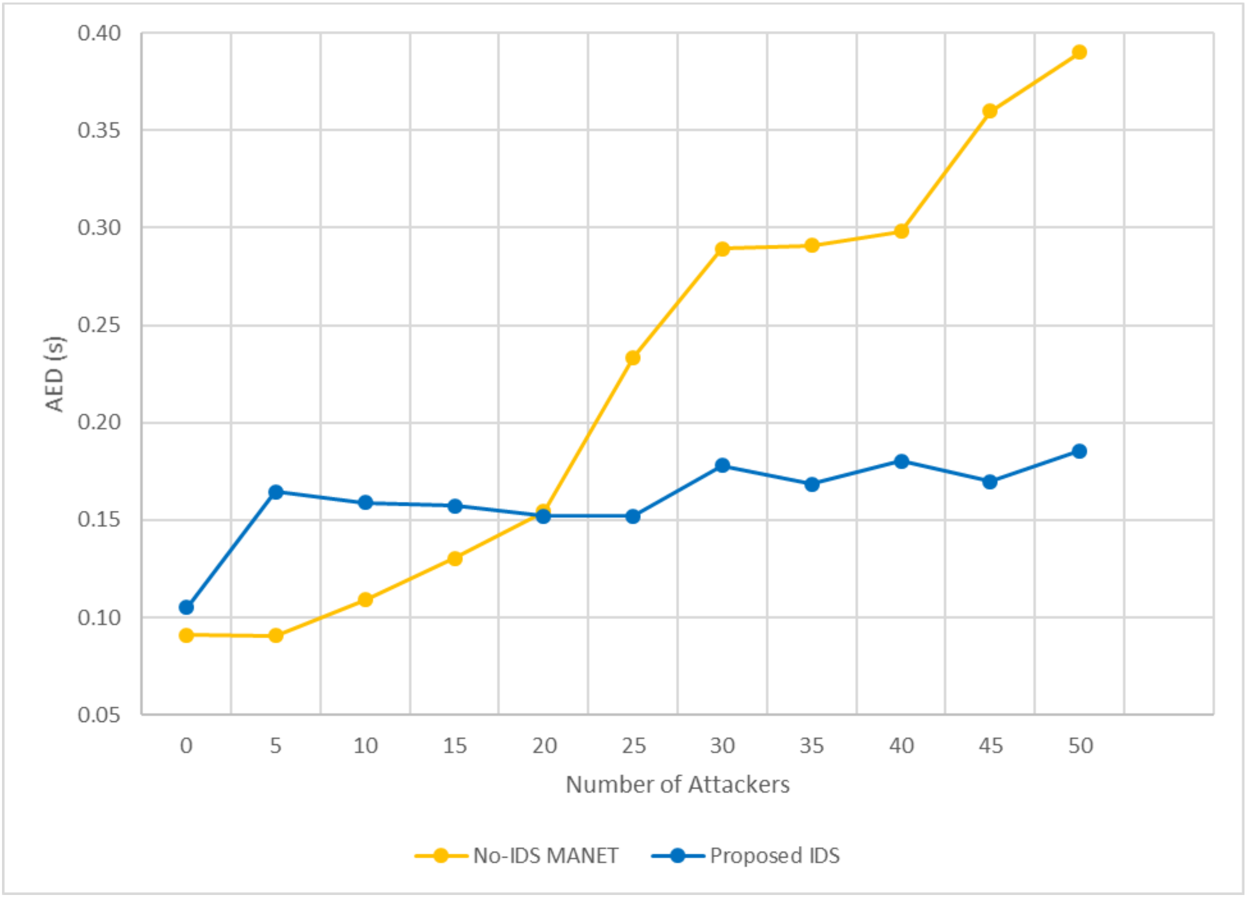

Figure 5 - AED of Proposed IDS vs. no-IDS MANET

Figure 5 above shows how the proposed IDS managed to maintain a consistently low rate of AED, as compared to the no-IDS MANET, despite the increase of the total number of attackers in the network. This happens because the network starts with a small number of attackers and nodes form large social communities with a higher total of strong-ties. This formation results in the successful delivery of data and detection-related packets between nodes in a single community as well as packets between different communities. As the number of attackers increase, the link between strong-ties may break, resulting in smaller social communities. However, due to the implementation of the proposed IDS, nodes re-route their packets through non-malicious nodes as soon as a malicious attacker is confirmed in the final DST calculations of the detection process. Additionally, smaller communities with no-legitimate weak-ties to other communities keep their detection process contained in their small community. This results in less packet loss as opposed to detection-related data flowing through malicious nodes between communities.

On the other hand, as malicious nodes occupy a larger portion of the network, delivery of detection-related packets gets affected. This is due to the packet loss encountered from malicious nodes dropping pass-through traffic. Additionally, even with IDS nodes re-routing packets around malicious nodes, smaller social communities continue to break down into smaller communities with less strong-ties. This results in less packets delivered around the network as strong-ties end up with fewer links to communicate with each other. Besides, some nodes might not be able to initiate the detection process since the total number of their strong-ties, as a result of a large number of attackers, does not suffice s_t identified previously. Additionally, as nodes get separated from each other by attackers breaking the corresponding links, more route-discovery queries are issued. This causes delays in packet delivery as nodes continue to look for a path with legitimate links to deliver such packets. The duration of establishing such paths, if even applicable as some nodes may potentially fail to do so as attackers occupy larger areas of the network, causes an increase in the AED values (Abdelshafy & King, 2013),

The comparison between the implementation of the proposed IDS versus no-IDS MANET shows improvements in the AED values as more attackers join the network. However, the goal of this study was to evaluate the proposed system's ability to minimize bandwidth consumption throughout the detection process. To examine the proposed system ability to fulfill that goal, a comparison of AED values between the proposed system and a CCIDS implementation is presented in the next sections.

Experiment 3 - Stand-alone IDS vs. Proposed IDS's Detection Accuracy Evaluation

This experiment was intended to compare the detection accuracy of the proposed system against a stand-alone version of itself. Before conducting this experiment, the researcher modified the original implementation of the proposed IDS. The modification involved removing the STBM, CDM, and GRM components along with changing the LDM to remove any calls to the CDM. This, in turn, removed all cooperative detection components from the IDS to match the stand-alone IDS definition outlined in Chapter 2. Additionally, the DST calculation code in the LDM was modified to make an intrusion decision after calculating the final DST value for each node. As such, all intrusion detection decisions were made individually by the IDS node without any cooperation or information sharing with the neighboring nodes. All STBM and CDM related thresholds were removed as they were irrelevant for this experiment.

The researcher then proceeded with a total of five test simulations to ensure that the modifications matched the description provided for the stand-alone IDS. In each iteration, the researcher selected a random node out of the total 100 nodes in the test MANET to install the stand-alone IDS module on. Each simulation ran for 100 seconds. After each iteration, the researcher inspected the decision output files and manually checked that all decisions were made solely by the randomly-selected IDS node. Manual modifications were made after iteration one and two as the final DST values were being printed out multiple times per each node. The researcher then corrected the LDM code to output an intrusion decision once per node per interval (t_l). By inspecting the output results from iterations three to five, the researcher observed consistency of the implementation in regard to the stand-alone description provided in Chapter 2.

Once accuracy of the implementation was established, the researcher proceeded to experiment simulations to extract detection evaluation metrics. The simulation procedures followed the same format as experiment 2. A total of ten simulations were executed. The first simulation started with five randomly-selected attackers, and five more attackers were added consecutively for each following iteration. Each duration spanned a total of 500 seconds. The researcher closely monitored each iteration to ensure no errors or sudden interruptions happened during simulations as a result of a sudden failure of the hosting environment. At the end of each iteration, the resulting detection decisions file was manually inspected to calculate the detection evaluation metrics as discussed in Chapter 3. The collected metrics are documented in table 7 below:

Total attackers	TP	TN	FP	FN	DTR	FPR
5	0	45	9	0	0.00000	0.16666
10	2	41	14	12	0.14285	0.25454
15	1	36	8	13	0.07142	0.18181
20	9	38	11	41	0.18000	0.22448
25	12	31	15	37	0.24489	0.32608
30	8	29	12	25	0.24242	0.29268
35	9	23	5	44	0.16981	0.17857
40	11	24	6	58	0.15942	0.20000
45	12	15	6	55	0.17910	0.28571
50	11	14	10	50	0.18032	0.41666

Table 7 - Evaluation Metrics Values of the Stand-alone IDS

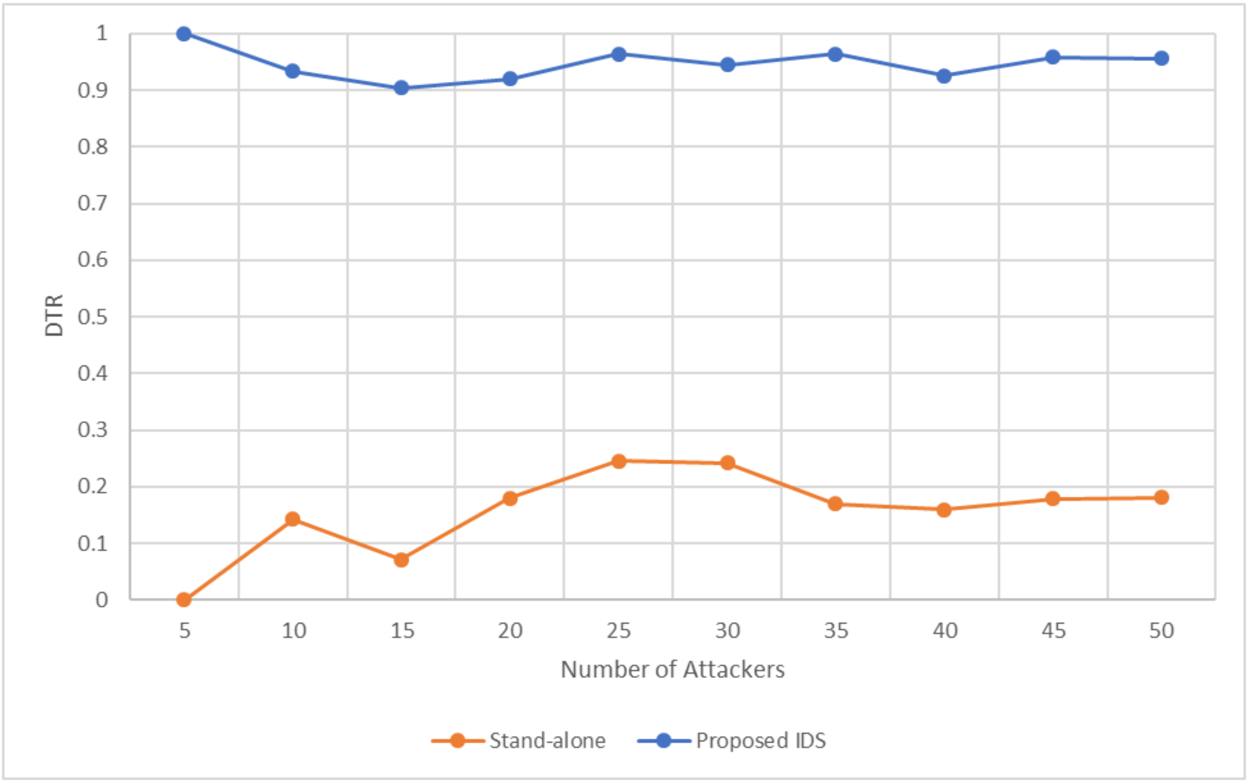

Figure 6 - DTRs of Proposed IDS vs. Stand-alone IDS

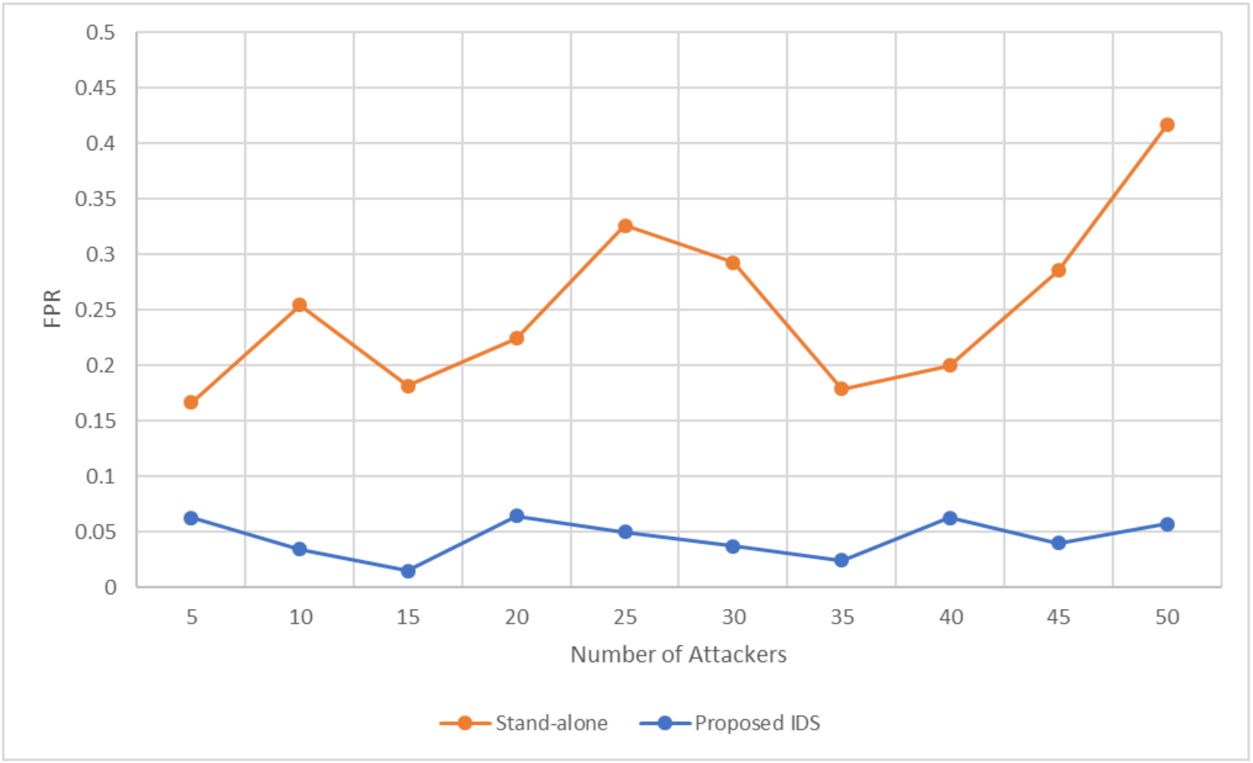

Figure 7 - FPRs of Proposed IDS vs. Stand-alone IDS

Data Analysis

Table 7 above shows the consistency of low TPs across all iterations, demonstrating the incapability of the stand-alone IDS in all cases to efficiently identify attackers in the network. This is based on the manual inspection of the detection decision output files, which revealed that the randomly-chosen IDS node was only able to detect attackers when they moved within its transmission range. As such, a lack of a wider view of the network rendered the stand-alone IDS incapable of observing and eventually detecting malicious activities happening in other parts of the network. This can be seen by the low overall values of TPs across all iterations.

In comparison with the proposed CIDS, it consistently maintained a high number of TPs across all iterations. Such results emphasize the significance of the application of social communities in keeping all IDS nodes informed of the security situation of the entire network. Additionally, the participation of trusted strong-ties in the final decision making significantly improved the accuracy of detecting malicious nodes, as seen in experiment 2 in the resulting TP values. As opposed to the stand-alone IDS where the IDS node made all final detection decisions on its own. This significantly impacted the detection accuracy, as observed across the resulting TP values.

On the other hand, table 7 shows how FPs of the stand-alone IDS remained high across all iterations. This is due to the fact that the stand-alone IDS node made all final DST calculations based on one-sided evidence, that is, its own. The lack of multiple DST calculations from social communities as provided by in the proposed CIDS, left the detecting node here with significant error-prone decision making. This resulted in a high number of FNs where malicious nodes were not detected as well as high FPs from falsely accusing innocent nodes. This reveals the criticality of multiple evidence to support the final detection decision and ensure the overall accuracy of the detection process.

Observing the TN values of the stand-alone IDS, table 7 above shows that the stand-alone IDS suffered from continuous degradation of the total TNs as more attackers joined the network. The stand-alone IDS ended up with a total TNs as low as 14 when half of the network was occupied with attackers. This was noteworthy when compared to those of the proposed IDS, which resulted in a high TNs of 33 in the same scenario. The proposed IDS managed to produce the high TNs due to the cooperative DST calculations that, as observed from the data, have saved a large number of innocent nodes from being marked as malicious. Such cooperative calculations were ultimately a result of the implementation of the concept of social communities in the proposed system.

The side-by-side comparison of DTRs and FPRs in figure 6 and 7 above, between the proposed IDS and a stand-alone IDS implementation shows two major observations. First, it shows the significance of implementing the concept of social-communities to provide a wider view on the network and utilize detection information coming only from trusted strong-ties. This is opposed to relying on a single observation from the IDS node to determine the final detection decision. Examining the DTR and FPR values of the two implementations reveals the obvious lack of detection accuracy in the stand-alone IDS as compared to the proposed system. This, in turn, shows the failure of the stand-alone IDS in fulfilling the goal of high detection accuracy with low false alarm rates. Table 7 above shows that the stand-alone IDS resulted in DTR values as low as 0.00000 with a highest of 0.24489. This was noteworthy when compared to the proposed IDS, as it maintained a high DTR rate with a lowest of 0.90476 and a highest of 1.00000 across all simulation iterations.

Additionally, the stand-alone IDS revealed the significance of applying DST in a cooperative manner when attempting to reduce and possibly eliminate false alarms in the network. A side-by-side comparison between the FPR values of the stand-alone versus the proposed IDS, as shown in figure 7 above, reveals a significant difference in the resulting values. The stand-alone IDS resulted in high FPR values, with the lowest being 0.16666 and the highest being 0.41666. As compared to the consistently low FPRs of the proposed system, which had a highest of 0.06451 and a lowest of 0.01492. The overall FPR values for the stand-alone IDS were significantly higher than those of the proposed IDS across all iterations. The stand-alone IDS suffered from such a high FPR because it lacked the significant implementations of DST and social communities and was incapable of accurately differentiating malicious from legitimate nodes. As opposed to the proposed IDS, which utilized the power of strong-ties along with DST in which lack/little evidence was taken in consideration in the final decision making. This in turn resulted in lower FPRs across all iterations as seen in figure 7 above.

The above detection accuracy comparison between the proposed system and the stand-alone IDS was conducted to evaluate the proposed system's fulfillment of the research goal of high detection accuracy. As outlined in Chapter 3, evaluating the detection accuracy in this research relied on two evaluation metrics: DTR and FPR. Across all simulations in this experiment, the proposed IDS managed to maintain high DTRs with values as high as 1.00000 and a lowest of 0.90476. This was noteworthy when compared to the stand-alone IDS, which consistently produced low DTR values, with a highest of 0.24489 and a lowest of 0.00000. On the other hand, the proposed system maintained low FPR values, even in situations where attackers occupied half of the network, with a lowest of 0.01492 and a highest of 0.06451. This was opposed to the stand-alone version, which suffered from high FPR values across all simulation iterations, producing a lowest FPR of 0.16666 with a significantly high FPR of 0.41666 when half of the test MANET was occupied with attackers.

From the results, it can be seen that the lack of the application of the concept of social communities combined with DST in the stand-alone IDS rendered the system inefficient even in cases where only few attackers existed in the network. The stand-alone IDS lacked the incorporation of lack/no evidence from strong-ties in the final detection decisions. As a result, the stand-alone IDS made all final detection decisions based solely on the IDS node's own observations. On the other hand, the lack of the application of the concept of social communities in the stand-alone IDS left the system incapable of incorporating trusted observations about suspect nodes in final decision making.

Additionally, the stand-alone IDS did not have any visibility of the security status of distant nodes, which left many attackers undetected, as can be seen from the consistently low TPs in this experiment. Results from this experiment that revealed consistently low DTRs and high FPRs show how the lack of a wider network visibility causes detection accuracy issues and inefficiencies. In MANETs, nodes can leave and rejoin, move to random locations inside the network or suffer from battery drainage affecting transmission activities. This leaves single observation-based detection susceptible to falsely detecting such cases as malicious, as can be seen from the high FPRs in this experiment.

Alternatively, the high DTR and low FPR values seen in the implementation of the proposed system are due to the combination of the concept of social communities and DST. First, the application of the concept of social communities forced the system to avoid accusing innocent nodes of being malicious without having enough strong-ties to support the claim. This can be seen in the consistently low FPRs found in experiment 2. Additionally, the application of DST restricted the detection process from accusing innocent nodes without providing strong supporting evidence from a node's social community. The combination of these two concepts allowed the IDS to minimize false accusations while making detection decisions based on strong evidence coming from trusted sources. The proposed system when compared to the stand-alone IDS has empirically shown the significance of these two concepts in fulfilling the research goal of high detection accuracy.

Experiment 4 – CCIDS vs. Proposed IDS's Bandwidth Consumption Evaluation

This experiment was intended to compare the proposed system's bandwidth consumption performance throughout the detection process against a CCIDS version of itself. Implementation the CCIDS involved modifying the proposed IDS to follow a common methodology in current CIDs where these systems disseminate detection-related information on a regular basis (Sterne et al., 2005; Razak et al., 2008; Shahnawaz, Joshi, & Gupta, 2012; Hernandez-Orallo, Serrat, Cano, Calafate, & Manzoni, 2012; Subramaniyan, Johnson, & Subramaniyan, 2014). The researcher modified the DIM component of the original IDS to send detection-related information to the IDS node's one-hop neighbors at each interval (t). That information consisted of a list of nodes along with their status (malicious/non-malicious) based on previous detection decisions made by the node or received in an alarm from other nodes. The original implementation already programmed the DIM to be aware of previous detection decisions as outlined in Chapter 3. Additionally, to test the implemented functionality, the DIM was modified to print out the total and destinations of detection-related packets sent at each interval (t). The purpose of that was to have a visual way to verify the accuracy of this modification.

The researcher then proceeded with a total of five test simulations to verify that the modifications matched the description outlined in Chapter 3. The researcher installed the modified IDS on all nodes in the test MANET. In each iteration, the researcher selected a total of ten random nodes out of the total 100 nodes in the test MANET to uninstall the IDS and install a random attacker module on. Each simulation ran for 100 seconds. After each iteration, the researcher inspected the DIM output file and manually checked that each IDS node was sending its malicious/non-malicious nodes list to its neighbors. After completing the manual inspection of the DIM output files for all simulations, no inconsistencies were found throughout all five simulations. For each iteration, the researcher observed the implementation conformed to the outlined description of the CCIDS implementation.

Once implementation accuracy was established, the researcher performed the experiment simulations. The simulation procedures followed the same format as experiments 2 and 3. A total of ten simulations were executed. These simulations started with five randomly-selected attackers, and five more attackers were added consecutively for each of the following iterations. Each simulation spanned a total of 500 seconds in duration. The researcher closely monitored each iteration to ensure no errors or sudden interruptions occurred during simulations as a result of a sudden failure of the hosting environment. After each iteration, the researcher analyzed the resulting trace file to calculate the AED value based on the calculation described in Chapter 3. Figure 8 below shows the resulting AED values across all iterations from the CCIDS as compared to those collected from the proposed IDS.

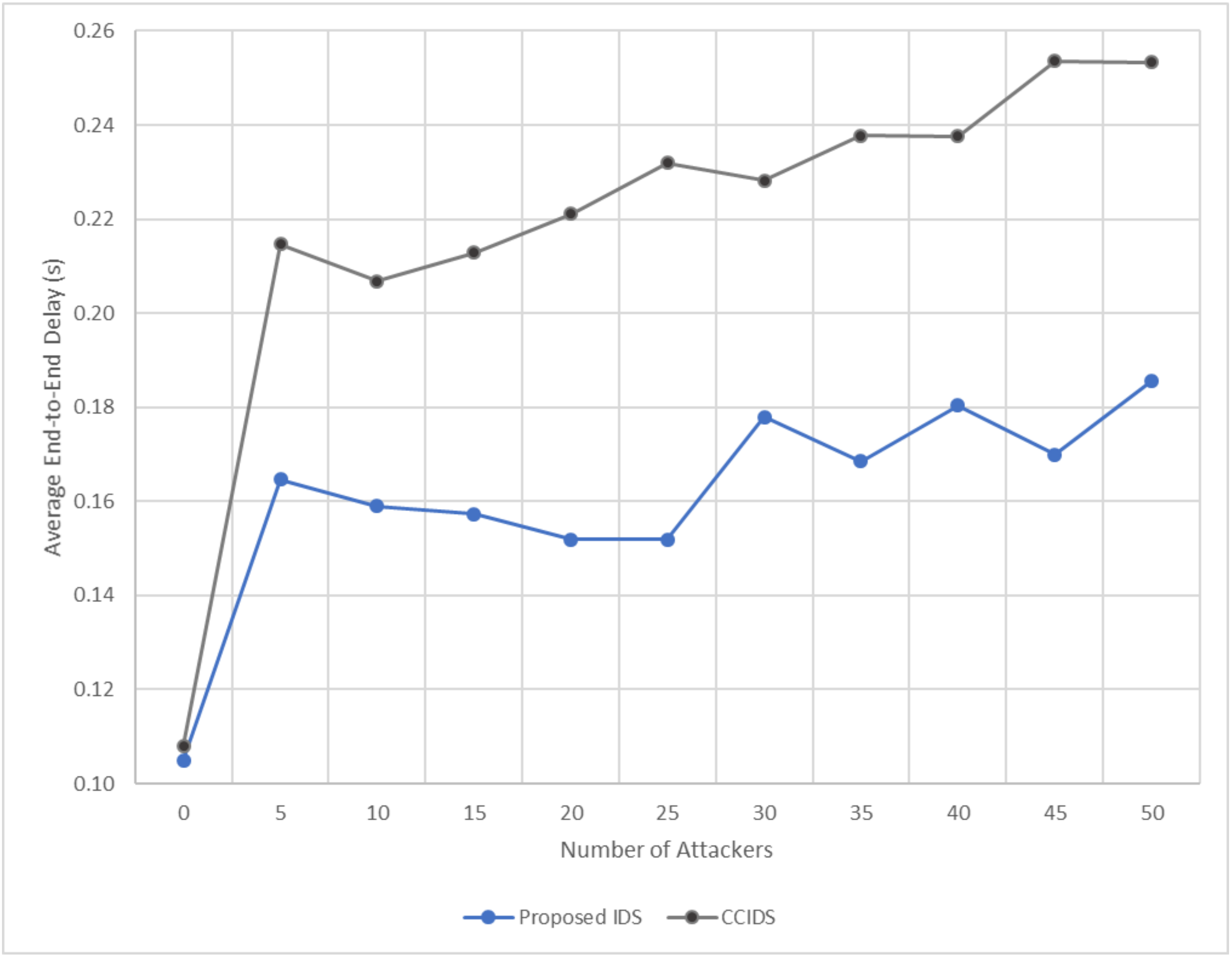

Figure 8 - AEDs of Proposed IDS vs. CCIDS

Data Analysis

Figure 8 above shows how the network suffered higher delays in the presence of CCIDS than those experienced with the proposed IDS. The CCIDS produced AED values as high as 0.25358 seconds with a lowest of 0.20674 seconds. The proposed IDS had consistently lower AED values, which were as low as 0.15192 seconds with a highest of 0.18554 seconds. The continuous dissemination of detection-related information contributed to the high delays produced by the CCIDS as they increase the potentiality of packet loss and collisions in the limited wireless medium of MANET (Lee & Gerla, 2000). Additionally, as more attackers join the network, the size of detection-related packets increase due to the corresponding increase in the size of the list of nodes sent by DIM. This caused the IDS nodes to consume extra bandwidth to send/receive these large packets, causing depletion in the available network bandwidth and delaying packet delivery (Vigna, Gwalani,Srinivasan, Belding-Royer, & Kemmerer, 2004). Moreover, the continuous dissemination of these packets added extra transmission delays since more routing discovery operations were needed to deliver these packets around malicious nodes as more attackers joined the network (Abdelshafy & King, 2013). The negative impact of the continuous dissemination of detection-related packets in CCIDS can be seen from the AED values in figure 8.

Figure 8 above presents a comparison between the resulting AED values from the CCIDS against the proposed system's. This comparison reveals an obvious improvement of AED values in the proposed system as opposed to CCIDS across all simulation iterations. In iterations where only five attackers existed in the network, the CCIDS produced an AED value of 0.2147 seconds, while the proposed system's AED value was 0.1647 seconds in the same scenario. On the other hand, the CCIDS resulted in an AED value of 0.25336 seconds when half of the network was occupied by attackers, while in the same scenario the proposed IDS produced a low AED value of 0.18554 seconds. The proposed system demonstrated lower AED values in both low and high attackers scenarios across all simulation iterations throughout the experiment.

Additionally, the CCIDS produced an overall average AED value of 0.2405 throughout the experiment while the proposed IDS maintained an overall average AED value of 0.177 seconds. Per these results, the CCIDS produced an overall average of 63.5 milliseconds delay over the proposed system's. Additionally, the CCIDS maintained AED values of over 0.2067 seconds in all iterations with attackers present in the network. On the other hand, the proposed system maintained AED values of less than 0.1856 seconds, even in situations where attackers occupied half of the network.

The side-by-side comparison between the proposed IDS and the CCIDS version revealed a major finding regarding the proposed IDS: the significance of applying the social communities concept in the proposed system and its observed impact on reducing bandwidth consumption throughout the detection process. This effect was consistent, even in scenarios where a large number of malicious nodes existed in the network. Nodes formulate their social communities utilizing the strong-ties features described in Chapter 3 without the need for continuous information dissemination. This showed a positive impact on the bandwidth consumption, as can be seen in the results. By enabling IDS nodes to be autonomous in their formation of social communities, the proposed approach allowed less bandwidth consumption as opposed to, for instance, relying on extra acknowledgement packets to establish such communities.

Additionally, the social communities restrictions that only alarm a node's social community instead of flooding the network reduce excessive packet transmission during the detection process, resulting in less bandwidth consumption. Lastly, combining DST and social communities to restrict the initiation of the detection process based on DST values, minimum strong-ties needed to initiate the detection, and strong-evidence support eliminates the need for excessive communications about a suspect node that might not be a suspect in the first place.

The primary purpose of this experiment was to evaluate the proposed system's ability to fulfill the research goal of minimizing bandwidth consumption throughout the detection process. As outlined in Chapter 3, AED was used as the evaluation metric for that goal. From the results above, it can be seen that the proposed system was better at conserving bandwidth than the CCIDS implementation. The experiment tested the two systems against scenarios with low numbers of attackers and those where attackers occupied large areas of the network. In all cases, the proposed IDS consistently maintained lower AED values than those in the CCIDS version. Additionally, the proposed implementation showed improved AED values in comparison with the no-IDS MANET thanks to the GRM and DIM roles in excluding malicious nodes from routing, resulting in more packets being delivered successfully. The application of the concept of social communities combined with the DST during the detection operations empirically demonstrated improvements in bandwidth consumption. This empirical proof can be seen in when comparing resulting AED values for the proposed system and the CCIDS.

Findings

The primary objective of this research was to implement a CIDS capable of producing high detection accuracy while minimizing bandwidth consumption throughout the detection process. To evaluate the proposed system's fulfillment of the stated goals, the implementation of the proposed IDS was compared to a stand-alone IDS and CCIDS implementations. DTR and FPR evaluation metrics were used to measure the system's ability to meet the high detection accuracy goal. Additionally, an AED evaluation metric was used to evaluate the system's ability to minimize bandwidth consumption throughout the detection process.

To evaluate the high detection accuracy goal of this study, a comparison of the intrusion detection decisions was made between the proposed system and a stand-alone IDS implementation, in the presence of a random mixture of attackers in the network. Research findings from that comparison suggest that the proposed system succeeded in fulfilling the high detection accuracy goal, with DTR values as high as 1.00000 with a lowest of 0.90476 and FPR values as low as 0.01492 with a highest of 0.06451. This can be compared to the stand-alone IDS, which resulted in a highest DTR of 0.24489 and a lowest of 0.00000 as well as a lowest FPR of 0.16666 and a highest of 0.41666. Additionally, the proposed system managed to maintain a low FPR of 0.05714 and a high DTR 0.95614, even when half of the network was occupied by malicious attackers. In comparison, the stand-alone version showed a high FPR of 0.41666 and a low DTR of 0.18032 in the very same situation.

On the other hand, to evaluate the proposed system's ability to fulfill the research goal of minimized bandwidth consumption throughout the detection process, the researcher compared the proposed IDS against the CCIDS implementation in scenarios where a mixture of random attackers existed in the network. AED was used as the evaluation metric for this purpose. Findings from the comparison showed the proposed system was able to achieve minimized bandwidth consumption goal with consistently low AEDs with values as low as 0.15192 seconds with a highest of 0.18554 seconds. Comparatively, the CCIDS recorded higher AED values with a highest of 0.25358 seconds and a lowest of 0.20674 seconds. The proposed IDS resulted in 63.5 milliseconds less overall average delay than the CCIDS. Additionally, findings show that the proposed system was successful in improving AED values as opposed to MANETs with no IDS installed. This is due to the application of the social community concept, which allowed IDS nodes to route traffic through legitimate nodes and, thus, eliminate delays/droppage caused by malicious nodes.

Results from this research empirically demonstrated the advantages of the application of social communities combined with DST in the proposed IDS. The autonomy provided by the concept of social communities allowed nodes to build their strong-ties without the need for excessive communications. This combined with the restrictions enforced by the application of the DST through eliminating initiation of cooperative detection without strong evidence, resulted in the system maintaining low AED values throughout the experiment. On the other hand, the enforcement of the concept of social communities on nodes to only consider detection reports from strong-ties eliminated potential false accusations coming from anonymous nodes. This, alongside the application of DST, in which a lack of or no evidence is considered in the final detection decision, resulted in the high DTRs and low FPRs previously discussed. The consistency of these results, throughout the lifetime of the experiment, showed the viability of applying social communities combined with DST in a CIDS implementation to achieve high detection accuracy and minimized bandwidth consumption in the detection process.

It should be noted that the research was conducted to evaluate the application of the concept of social communities combined with DST in a CIDS implementation towards the achievement of the research goals. As such, the research did not cover all attack types that a MANET could be exposed to. Only a number of these attacks were implemented to contain the scope of this study and evaluate the viability of the proposed system. Additionally, the research in this study did not cover situations where MANETs have a total number of nodes exceeding 100, to keep the scope of this study manageable. It was thought that 100 nodes in the test MANET is a sufficient number to simulate and evaluate the proposed system as followed by similar research studies in this area (Theresa & Sakthivel, 2017; Sangeetha & Kumar, 2018; Veeraiah & Krishna, 2018).

Summary of Findings

The study aimed to develop a CIDS capable of producing high detection accuracy while minimizing bandwidth consumption throughout the detection process. The research proposed the application of social communities in combination with DST in a CIDS implementation to achieve these goals. Several experiments were conducted to evaluate whether the proposed IDS fulfills the stated goals. Results of these experiments were examined and quantitatively evaluated. Outcomes of this study show the success of the proposed system in achieving both goals, even in situations where large number of attackers existed in the network. To evaluate the proposed system's fulfillment of the high detection accuracy goal, the study quantitatively compared the proposed system against a stand-alone IDS implementation.

The proposed system produced DTRs as high as 1.00000 with a lowest of 0.90476. Comparatively, the stand-alone IDS implementation achieved a highest DTR value of 0.24489 and a lowest of 0.00000. Similarly, the proposed system's results produced consistently low FPRs, with the lowest value at 0.01492 and the highest at 0.06451. This is opposed to a lowest FPR of 0.16666 and a highest of 0.41666 for the stand-alone version. The results of the proposed system's detection accuracy far exceeded those of the stand-alone IDS.

Additionally, to evaluate the proposed system's ability to fulfill the study's second goal of minimized bandwidth consumption throughout the detection process, the system was evaluated against a CCIDS implementation. Such an implementation followed an approach similar to the CIDS approaches found in the current literature, where detection-related information is disseminated over the network on a regular basis (Sterne et al., 2005; Razak et al., 2008; Shahnawaz, Joshi, & Gupta, 2012; Hernandez-Orallo, Serrat, Cano, Calafate, & Manzoni, 2012; Subramaniyan, Johnson, & Subramaniyan, 2014). Several simulations were conducted to extract the results of that comparison.

Quantitative comparison of these results revealed the proposed system's ability to achieve the aforementioned goal, with AED values as low as 0.15192 seconds with a highest of 0.18554 seconds. This was opposed to the CCIDS, which had a lowest value 0.20674 seconds and a highest value of 0.25358 seconds. The CCIDS resulted in an overall average of 63.5 milliseconds additional delay over the proposed IDS. Throughout the experiment, the proposed system resulted in lower AED values as compared to the CCIDS version, empirically demonstrating the proposed system's ability to minimize bandwidth consumption throughout the detection process.

The proposed system was implemented and evaluated through simulations in multiple experiments. These experiments involved randomized types of attackers with situations where attackers occupied half of the network. The developed system was able to successfully meet the research goals consistently across all experiments. Therefore, the results of this study demonstrate the success and efficacy of combining the social communities concept and DST in a CIDS implementation to improve detection accuracy while reducing bandwidth consumption throughout the detection process.

Chapter 5

Conclusions, Implications, Recommendations, and Summary

Conclusions

Current CIDS found in the literature suffer from high bandwidth overhead throughout the detection process as well as high false alarms rates. This research targeted the development of a CIDS capable of producing high detection accuracy while minimizing bandwidth consumption throughout the detection process. The study demonstrated the viability of combining social communities and DST in a CIDS implementation to achieve the presented research goals.

The study provided empirical evidence supporting the proposed solution through the implementation and quantitative evaluation of the proposed CIDS. Evaluation of the detection accuracy involved modifying the proposed system to implement a stand-alone IDS version and compare the results. Experimental evaluation demonstrated that applying the social communities concept along with DST in the proposed CIDS resulted in high detection accuracy, with consistently high DTRs as high as 1.00000 with a lowest of 0.90476. Comparatively, the stand-alone IDS implementation developed in this research resulted in consistently low DTRs, with a highest value of 0.24489 and a lowest of 0.00000. At the same time, the proposed CIDS maintained low FPRs with a lowest value of 0.01492 and a highest value of 0.06451 throughout the lifetime of the experiment. Comparatively, the stand-alone IDS resulted in overall high FPRs, with a highest value of 0.41666 and a lowest value of 0.16666. In fact, the proposed IDS sustained a low FPR of 0.05714 with a DTR of 0.95614, even when half of the network was occupied with malicious attackers. This occurred while the stand-alone IDS, in the same scenario, produced a high FPR of 0.41666 and a low DTR of 0.18032.

The study evaluated the bandwidth consumption of the proposed CIDS by modifying the proposed CIDS to implement a CCIDS and comparing the results. Experimental results demonstrated the system's ability to effectively minimize bandwidth consumption throughout the detection process. The system produced AED values as low as 0.15192 seconds with a highest of 0.18554 seconds as opposed to a CCIDS implementation, where the lowest AED value was 0.20674 seconds and the highest was 0.25358 seconds. On average, the proposed system produced reduced delays, represented in AEDs, of 63.5 milliseconds when compared to the average AEDs produced by the CCIDS.

Throughout all simulations, the proposed system sustained high DTRs, low FPRs, and low AED values. Additionally, the system did not suffer from an increase of FPRs proportional to the increase of the total number of attackers in the network, as found in IDS solutions in the current literature (Mustafa & Xiong, 2013; Ullah, Khan, Ahmed, Javaid & Khan, 2016; Alattar, Sailhan & Bourgeois, 2012; Mahmoud & Shen, 2010). In fact, the proposed system maintained low FPRs throughout the experiment, with values as low as 0.014925 with a highest of 0.06451. Given the consistency of high DTRs, low FPRs, and low AEDs throughout the entirety of the experiment, it can be stated that the proposed system succeeded in fulfilling the research goals of high detection accuracy and minimized bandwidth consumption.

As previously stated in this research, current CIDSs suffer from degraded detection accuracy due to the reliance on detection reports sourced from anonymous nodes. At the same time, these solutions tend to create high bandwidth consumption as a result of their continuous dissemination of detection-related information. This research aimed to address these issues by applying the social communities concept and DST in a CIDS implementation. Experimental results strongly demonstrated the proposed implementation was able to address the current issues outlined above. The study concluded the combination of the social communities concept and DST in a CIDS implementation was effective at increasing detection accuracy while minimizing bandwidth consumption throughout the detection operations.

Implications

For a CIDS to be efficient, based on Cannady's (2013) criterion, it needs to be able to identify attacks accurately in a timely manner. This criterion shows the significance of high detection accuracy and low bandwidth utilization in the effectiveness of an IDS. When it comes to detection accuracy, an IDS must be able to accurately identify attackers while avoiding false accusations against innocent nodes. On the other hand, an IDS should avoid over-consumption of the bandwidth-constrained nodes of MANETs. This in turn, reduces the effects of the IDS communications during normal network operations towards sustainable packet delivery. Thus, attacks identification packets can be delivered on time throughout the network. This research aimed to achieve high detection accuracy with minimized bandwidth consumption in a CIDS implementation.

As the applications of MANETs are gaining a widespread adoption in more commercial as well as military areas, so do their security threats and the significance of effective IDS solutions (Banerjee, Nandi, Dey, & Saha, 2015; Keyshap, 2015). The primary implication of this research is that the application of the social communities concept combined with DST can produce high detection accuracy while minimizing bandwidth consumption when applied to a CIDS implementation. The study has empirically shown through experimental evidence how the combination of these two applications eliminates reliance on detection reports coming from anonymous nodes in the intrusion detection process.

On the other hand, the consideration of a lack of or no evidence in the final intrusion detection decisions reduced false accusations, thus low false alarm rates. Such a consideration resulted in overall high detection accuracy in terms of accurately detecting malicious nodes across all the experimental testing. The proposed CIDS demonstrated high detection accuracy even in cases where half of the network was occupied by malicious attackers with DTRs as high as 1.00000 with a lowest of 0.90476. The application of the social communities concept allowed nodes to have the autonomy of forming strong-ties without the need for continuous communications. At the same time, the restrictions that the DST enforced on the system eliminated initiations of unnecessary cooperative communications without strong evidence. Experimental results demonstrated the efficiency of the CIDS implementation ability to sustain low bandwidth consumption, represented in consistently low AEDs, throughout the lifetime of the experiment.

Another implication obtained from the experimental results showed that the proportional increase of false alarms can be avoided by implementing the above-mentioned concepts. This is due to the restrictions the proposed system enforced on the IDS nodes to initiate or respond to cooperative detections. These include minimum strong-ties, minimum packets observed against a suspect, attack thresholds, and the applied DST calculations. These restrictions contributed to the hindrance of a proportional increase of false accusations against legitimate nodes, as discussed on Chapter 4. As a result, the proposed system maintained a steady low-rate of false alarms represented in the FPRs obtained throughout all the experiments, with values as low as 0.014925 with a highest of 0.06451. This contributed to the high detection accuracy goal and saved the bandwidth-constrained network from unnecessary detection packets, which in turn translates to less bandwidth consumption.

Recommendations

To contain the scope of this research to a manageable level, the researcher chose five of the common attack types against MANETs, as discussed in Chapter 3. However, as advancements in MANETs continue to move forward, so does the creativity of attackers to invent new attack types (Cannady, 2013). Wider implementation of attack types is recommended to evaluate the system's viability against new, potentially more complicated attack types.

Another recommendation for future research would be to extend the current implementation of the proposed system's LDM. Machine learning implementations for attack detection demonstrated undeniable efficiency in detecting both known and unknown attacks (Butun, Morgera, & Sankar, 2014). Future research may extend the LDM component to incorporate a machine learning-based engine for attack detection. This could increase the system's effectiveness in detecting both known and unknown attacks. MANET implementations are different depending on the deployment model. This difference introduces an increased complexity on manual calculation of attack thresholds. By incorporating machine learning-based detection in the LDM, the process of learning attack thresholds could potentially be reduced tremendously while, at the same time, the system would gain the advantage of adapting the detection process with network changes.

Lastly, this research implemented the LDM and STBM components as interval-based modules. This meant that the LDM triggered the attack detection process at specific intervals, as opposed to a continuous detection process. Similarly, the STBM calculated strong-ties features, discussed in Chapter 3, periodically at a predefined interval. This implementation was meant to contribute to lowering bandwidth consumption while conserving nodes' energy. It would be worthwhile for future research to investigate the variation of the LDM/STBM intervals and their impacts on the accuracy of the detection process as well as overall bandwidth consumption in the network.

Summary

The unique characteristics of MANETs have led to their wide adoptions in various military and commercial fields. The infrastructure-less nature of MANETs allows them to be deployed in various situations where no infrastructure exists, such as disaster relief sites and battlefields. This is due to their ability to dynamically form topologies as each node in the network acts as both host and router. On the other hand, nodes in MANETs have limited resources, constrained-bandwidth, and limited wireless range. All of these distinctive attributes made the mission of providing security solutions for MANETs a challenging task (Hubaux, Buttyán, & Capkun, 2001; Cannady, 2010; Sheik et al., 2010).

Due to their dynamic nature, security solutions designed for fixed networks cannot be applied to MANETs. Previous research proposed various types of solutions to the security vulnerabilities in MANETs while keeping in consideration the limitations accompanied with these networks (Kim & Jang, 2006; Mikki, 2009; Maleki, Dantu, & Pedram, 2002). The current body of knowledge contains a large number of preventive solutions that attempt to block attacks against MANETs before their occurrence. However, history has shown that these solutions cannot survive on their own as their exploitability increases along with the increased complexities of MANETs (Yang et al., 2004). As such, a second line of defense, represented in IDSs, has gained strong momentum from researchers. An extensive body of knowledge exists in the current literature pertaining to IDS solutions targeting intrusion-free MANETs. However, most of these solutions fail to deliver their promise due to the dynamic nature of MANETs.

For IDS solutions to operate efficiently in such infrastructure-less networks, the cooperative detection process is deemed mandatory (Mahmood, Amin, Amir, & Khan, 2009). Various CIDS were proposed in the current literature targeting the inclusion of cooperativeness in their implementations. However, all of the solutions suffer from two major problems: high communication overhead caused by continuous information exchange and reliance on intrusion reports originating from anonymous nodes. These result in high false alarms rates that cause degradation in the detection accuracy as well as increased bandwidth consumption that might disrupt the normal routing operations in such bandwidth-constrained networks.

The study developed a novel approach to implement an efficient CIDS by applying the concept of social communities with DST. That is a CIDS capable of achieving high detection accuracy while minimizing bandwidth consumption throughout the detection process. The concept of social communities, which has never been applied to MANETs security before, was implemented to improve detection accuracy. This was done by building strongly-tied communities that enable the exchange of reliable detection information solely among nodes' social circles. This addressed a major limitation that exists in current approaches: their reliance on intrusion reports from anonymous nodes, which can result in high false alarms rates. Additionally, the application of communities in this research allowed nodes the autonomy to build their social circle without the need for extensive back-and-forth communications with other nodes, thus minimizing bandwidth consumption.

Alongside the concept of social communities, this research proposed the application of the DST to improve detection accuracy and minimize false alarms rates. Researchers have applied DST in their security solutions in previous studies. However, the main issue with these applications is the determination of trustworthiness and untrustworthiness of nodes when weighing in nodes' votes (Chen & Venkataramanan, 2005; Li & Joshi, 2009). This requires a high bandwidth overhead imposed by trust-related information dissemination to establish accurate calculations. Otherwise, DST can combine observations from nodes without regard to their trustworthiness. However, this might yield inaccurate results in the presence of a large number of malicious nodes in the network (Chen & Venkataramanan, 2005).

Very little research has been done towards the application of DST for intrusion detection in MANETs. All of which relied on the calculated trustworthiness of nodes in the combined decision-making process. This can be problematic in the presence of a large number of malicious nodes because it can result in manipulated votes against legitimate nodes (Rajakumar et al., 2014). The application of DST in this research targeted the elimination of detection reports from anonymous sources, which usually result in a high rate of false alarms. Additionally, the research utilized the DST against observations obtained solely from strong-ties to handle cases where some of these nodes did not catch enough evidence against the suspicious activity.

The goal of this research was to develop, through an experimental approach, a CIDS that is capable of producing high detection accuracy while minimizing bandwidth consumption throughout the detection process. The combination of social communities and DST in the proposed CIDS aimed to address the study's goals. This research used DTR and FPR evaluation metrics to evaluate the proposed system's ability to meet the research goal of high detection accuracy. On the other hand, AED was used to evaluate whether the proposed system achieved minimized bandwidth consumption as stated in the research goals. Extensive experimental testing was conducted to evaluate the system in various scenarios where a mixture of random attackers existed in the network. Experimental results showed that the proposed system's achieved DTR values were as high as 1.00000 and the FPR values were as low as 0.014925. Additionally, the system managed to maintain low AED values throughout the experiment iterations, with a lowest value of 0.15192 seconds and a highest value of 0.18554 seconds.

To further evaluate the system's ability to meet the high detection accuracy goal, the researcher implemented a stand-alone IDS version of the proposed system. The stand-alone IDS underwent similar experimental evaluations as those for the proposed system. Results from all evaluations showed success and demonstrated that the proposed system was more effective at achieving high detection accuracy than the stand-alone IDS. Experimental evaluations revealed that the stand-alone IDS achieved lower DTR values, with a highest value of 0.24489 and a lowest value of 0.00000. As compared to the proposed CIDS, which achieved a DTR as high as 1.00000 with a lowest of 0.90476. Similarly, FPR values of the stand-alone IDS were as high as 0.41666 with a lowest value of 0.16666. Comparatively, the proposed CIDS maintained low FPR as low as 0.014925 with a highest value of 0.06451.

On the other hand, the researcher implemented a CCIDS version of the proposed CIDS, which disseminated detection-related information to other nodes on a regular basis. Same experimental evaluations applied for the proposed CIDS were applied to the CCIDS. That experiment's focus was on collecting and quantitatively comparing AED values of the CCIDS against those of the proposed IDS. The experimental results of the CCIDS revealed consistently higher AEDs, as compared to the proposed IDS, with values as high as 0.25358 seconds with a lowest of 0.20674 seconds. Comparatively, the proposed CIDS resulted in AED values as low as 0.15192 seconds with a highest of 0.18554 seconds. On average, the proposed IDS reduced delays, represented in AED, by 63.5 milliseconds when compared to the CCIDS. Quantitative evaluations empirically demonstrated the success of the proposed system's ability to minimize bandwidth consumption throughout the detection process, as compared to the CCIDS.

The study evaluated the viability of applying the concept of social communities along with DST towards a CIDS implementation. Results from the study suggest that such an application results in improved detection accuracy while maintaining minimal bandwidth consumption throughout the detection operations. Empirical evidence from experiments conducted in this research demonstrated consistent results from the proposed system. That is, the system maintained high DTR, low FPR, and low AED values across all experiments, even when half of the network was occupied with malicious attackers. Given the consistency of these values throughout the entirety of the experiment, the study concluded that the proposed system succeeded in fulfilling the research goals of achieving high detection accuracy and minimized bandwidth consumption throughout the detection process.

This research resulted in two primary implications. First, the application of the concept of social communities combined with DST can produce high detection accuracy while minimizing bandwidth consumption when applied to a CIDS implementation. The study has empirically proven, through evidence, how the combination of these two applications eliminate considerations of detection reports coming from anonymous nodes in the intrusion detection process. On the other hand, the consideration of a lack of or no evidence in the final intrusion detection decisions helped reduce false accusations against innocent nodes, thus lowering false alarm rates.

The second implication of this study suggests that the proportional increase of false alarms can be avoided through the implementation of the concept of social communities and DST. The restrictions that the proposed system enforced on the IDS nodes along with the DST calculations contributed to the hindrance of the proportional increase of false accusations against legitimate nodes. As a result, the proposed system maintained a steady low rate of false alarms all across the experiment. Thus, this contributed to the high detection accuracy goal, and it saved the network from unnecessary detection packets, which translated to less bandwidth consumption

Results of the study suggest three recommendations for future research. First, as the research implementation covered only five attack types to contain the scope of the study, research into more attack types is recommended. This would help evaluate the system's viability against new, potentially, more complicated attack types. Secondly, future research could variate the LDM and STBM intervals and examine their effects on the overall detection accuracy and bandwidth consumption. Lastly, previous studies demonstrated advancements in machine learning techniques towards attack detection and their demonstrated ability to adapt to behavioral changes in the network (Butun, Morgera, & Sankar, 2014). As such, another recommendation of this study is to extend the LDM to include a machine-learning-based attack detection and evaluate the impact of such an extension on the overall detection accuracy.

Appendix A

IDS Source Code

```
//IDS used in PhD research for Adam Solomon
//Dissertation titled "A Novel Cooperative Intrusion Detection System
//for Mobile Ad Hoc Networks"
//Nova Southeastern University


//**********************************************************************
// Global Objects
//**********************************************************************
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <list>

const double MAJORITY_THRESHOLD = 80;
const double FINAL_DECISION_THRESHOLD = 0.7132;
const double THRESHOLD_RECENCY_THRESHOLD = 4.0;
const double THRESHOLD_RECIPROCITY_SHARED_THRESHOLD = 16;
const double THRESHOLD_TIME_SINCE_FIRST_MESSAGE = 10.0;
const double BLACKHOLE_THRESHOLD = 0.9;
const double GRAYHOLE_THRESHOLD = 0.4;
const double MODIFICATION_THRESHOLD = 0.5;
const double GHOLE_COLLECTION_INTERVAL = 3;
const double BHOLE_COLLECTION_INTERVAL = 1;
const double LDM_DELAY_INTERVAL = 6.0;
const double STT_DELAY_INTERVAL = 5.0;
const double FLOODING_MAX_PACKET_SIZE = 800;
const double RUSHING_MAX_DELIVERY_TIME = 2.0;
const double MINIMUM_STRONG_TIES = 5;
const double MINIMUM_PACKETS_COLLECTED = 20;


struct Node
{
	int ip;
	int DATA_TOTAL;
	int RREQ_TOTAL;
	int RREP_TOTAL;
	double time_of_last_communication;
	double time_of_first_communcation;
	bool isMalicious;
	bool isStrongTie;
	list<Packet> packets;
	list<double> listOfPacketDeliveryTimes;
	list<double> listOfPacketSizes;
};

struct Packet
{
	int32_t ip_src;
	int32_t ip_dst;
	double time;
	char *data;
	int data_size;
	int protocol;
	const char *pk_type;
	int originPort;
	int destinationPort;
	bool isReceived;
	bool isForwarded;
	bool isModified;
	bool isToForward;
	double deliveryTime;
	PacketType type;
};

enum PacketType
{
	RREQ,
	RREP,
	_DATA,
	BROADCAST_IP_MESSAGE,
	OTHER
};

struct hdr_ids_alarm
{
	u_int8_t rq_type;
	u_int8_t reserved[2];
	nsaddr_t rq_dst;
	nsaddr_t rq_src;
	nsaddr_t malicious_node;
	double rq_timestamp; // when REQUEST sent;

	inline int size()
	{
		int sz = 0;
```

```
        sz = 8 * sizeof(u_int32_t);
        assert(sz >= 0);
        return sz;
    }
};

struct hdr_ids_cdm_req
{
    u_int8_t rq_type;
    int rq_id;
    u_int8_t reserved[2];
    nsaddr_t rq_dst;
    nsaddr_t rq_src;
    nsaddr_t suspect_node;
    int attack_type;
    double rq_timestamp;

    inline int size()
    {
        int sz = 0;
        sz = 8 * sizeof(u_int32_t);
        assert(sz >= 0);
        return sz;
    }
};

struct hdr_ids_cdm_rep
{
    u_int8_t rq_type;
    int rq_id;
    u_int8_t reserved[2];
    nsaddr_t rq_dst;
    nsaddr_t rq_src;
    nsaddr_t suspect_node;
    int attack_type;
    double rq_timestamp;

    double isMalicious;
    double isNonMalicious;
    double isUncertain;

    inline int size()
    {
        int sz = 0;
        sz = 10 * sizeof(u_int32_t);
        assert(sz >= 0);
```

```
                return sz;
        }
};

struct CDMReply
{
        int reqId;
        int repId;
        int replyingNodeIP;
        int suspectNodeIP;
        double rp_timestamp;
        AttackType suspectedAttack;
        double isMalicious;
        double isNonMalicious;
        double isUncertain;
};

struct CDMRequest
{
        int reqId;
        AttackType suspectedAttack;
        int initiatingNodeIP;
        int suspectNodeIP;
        double rq_timestamp;
        list<CDMReply *> replyQueue;
        bool isFullfilled;
        int totalRequestsSent;
};

struct AttackDSTValues
{
        AttackType attackType;
        double malicious;
        double nonMalicious;
        double uncertain;
};

struct GRMAlarm
{
        int maliciousNodeIP;
        bool isSentToAllTies;
        double timeAlarmSent;
};

enum AttackType
{
        Blackhole,
        Grayhole,
        Modification,
        Rushing,
        Flooding,
        None
};


//*********************************************************************
// DIM Component
//*********************************************************************
//DIM.h
#ifndef __DIM_H__
#define __DIM_H__
#include "global.h"
#include  "aodv/aodv_rtable.h"
#include "aodv.h"

class DIM
{
          list<Node> Nodes;
          list<double> listOfPacketDeliveryTimes;
          list<double> listOfPacketSizes;

          void HandleListenedPacket(const Packet *p);
          void HandleReceivedPacket(const Packet *p);
          void SavePacket(int32_t source_ip, int32_t destination_ip, int source_ip, int
        destination_ip, int sourcePort, int destinationPort, packet_t packet_type,
                double pktime, const char *rt_type, int original_packet_size, Packet *p);
          void SaveNode(node *nd);
          Node *IsExistingNode(int ip);
          list<Nodes> GetAllSavedNodes() void NodeReceivedPacketToForward(Packet *p);
          void NodeForwardedPacket(Packet *p);
}

//DIM.cpp

//function to read packets listened
void
DIM::HandleListenedPacket(const Packet *p)
{
          hdr_ip *iph = hdr_ip::access(p);
          hdr_cmn *pk = hdr_cmn::access(p);

          hdr_mac802_11 *mh;
```

```
            mh = HDR_MAC802_11(p);

            hdr_aodv *ah = HDR_AODV(p);
            aodv_rt_entry *rt = rtable.rt_lookup(iph->daddr());
            bool isSourceMalicious = IsExistingNode(iph->src().addr_)->isMalicious;
            if (!isSourceMalicious)
            {
                        SavePacket(iph->src().addr_, iph->dst().addr_, ETHER_ADDR(mh->dh_ta),
ip_dst, iph->sport(),
                                                iph->dport(), pk->ptype_, Scheduler::instance().clock(),
ah->ah_type, pk->size(), p);
            }
}

// function to handle saving packets
void SavePacket(int32_t source_ip, int32_t destination_ip, int source_ip, int destination_ip, int
sourcePort, int destinationPort, packet_t packet_type,
                                    double pktime, const char *rt_type, int
original_packet_size, Packet *p)
{
            {
                        Node *nd = IsExistingNode(source_ip);
                        if (source_ip != currentIP)
                        {
                                    if ((nd = IsExistingNode(source_ip)) == NULL)
                                    {
                                                nd = SaveNode(source_ip, pktime);
                                                nd->ip = source_ip;

                                                if (rt_type == "RREQ")
                                                {
                                                            nd->RREQ_TOTAL = nd->RREQ_TOTAL + 1;
                                                            nd->pkt_type = RREQ;
                                                            double rreq_delivery_time = pktime -
original_rreq_sent_time;
                                                            nd->totalRreqPackts = nd->totalRreqPackts + 1;
                                                            nd-
>listOfPacketDeliveryTimes.push_back(rreq_delivery_time);
                                                }
                                                else if (rt_type == "DATA")
                                                {
                                                            nd->DATA_TOTAL = nd->DATA_TOTAL + 1;
                                                            nd->pkt_type = DATA;
                                                            nd-
>listOfPacketSizes.push_back(original_packet_size);
                                                }
```

```
                                        else if (rt_type == "RREP")
                                        {

                                                nd->RREP_TOTAL = nd->RREP_TOTAL + 1;
                                                nd->pkt_type = RREP;
                                        }
                                        else if (rt_type == "BROADCAST_IP_MESSAGE")
                                        {
                                                nd->BROADCAST_TOTAL = nd-
>BROADCAST_TOTAL + 1;
                                                nd->pkt_type = BROADCAST_IP_MESSAGE;
                                        }
                                }
                                else
                                {
                                       nd = new Node();
                                        nd->listOfPacketSizes.push_back(original_packet_size);
                                        nd->ip = source_ip;
                                        nd->time = pktime;
                                        nd->time_of_last_communication = pktime;
                                        if (rt_type == "RREQ")
                                        {
                                                nd->RREQ_TOTAL = nd->RREQ_TOTAL + 1;
                                                nd->pkt_type = RREQ;

                                                double rreq_delivery_time = pktime -
original_rreq_sent_time;
                                                nd-
>listOfPacketDeliveryTimes.push_back(rreq_delivery_time);
                                                nd->totalRreqPackts = nd->totalRreqPackts + 1;
                                        }
                                        else if (rt_type == "DATA")
                                        {
                                                nd->DATA_TOTAL = nd->DATA_TOTAL + 1;
                                                nd->pkt_type = _DATA;
                                        }

                                        else if (rt_type == "RREP")
                                        {
                                                nd->RREP_TOTAL = nd->RREP_TOTAL + 1;
                                                nd->pkt_type = RREP;
                                        }
                                        else if (rt_type == "BROADCAST_IP_MESSAGE")
                                        {
                                                nd->pkt_type = BROADCAST_IP_MESSAGE;
                                                nd->BROADCAST_TOTAL = nd-
>BROADCAST_TOTAL + 1;
                                        }
                                }
                                nd->packets.push_back(p);
                                if ((destination_ip != nd->ip) && (destination_ip != currentIP))
                                {
                                        NodeReceivedPacketToForward(nd->ip, p);
                                }
                                if (source_ip != nd->ip)
                                {
                                        NodeForwardedPacket(nd->ip, p);
                                }
                        }
                }

                //function to save nodes of which packets listened tby he DIM
                void DIM::SaveNode(node * nd)
                {
                        Nodes.push_back(nd);
                }

                //function to check if DIM has records of a node
                Node *DIM::IsExistingNode(int ip)
                {
                        Node *exists = false;
                        for (list<Node>::iterator it = Nodes.begin(); it != Nodes.end(); ++it)
                        {
                                if (it->IP == ip)
                                {
                                        return it;
                                }
                        }
                        return NULL;
                }

              //function to handle received packets
              DIM::HandleReceivedPacket(const Packet *p)
              {
                hdr_ip *iph = hdr_ip::access(p);
                hdr_cmn *pk = hdr_cmn::access(p);

                hdr_mac802_11 *mh;
                mh = HDR_MAC802_11(p);

                hdr_aodv *ah = HDR_AODV(p);
```

```
        aodv_rt_entry *rt = rtable.rt_lookup(iph->daddr());
        bool isSourceMalicious = IsExistingNode(iph->src().addr_)->isMalicious;
        if (isSourceMalicious)
        {
                AODV->drop(p, DROP_RTR_NO_ROUTE);
        }
        else {
        if ( iph->src().addr_ == currentIP)
        {
              SavePacket(iph->src().addr_, iph->dst().addr_, ETHER_ADDR(mh->dh_ta),
              ip_dst, iph->sport(),  iph->dport(), pk->ptype_, Scheduler::instance().clock(), ah-
              >ah_type, pk->size(), p);
        }
        else {
                AODV->forward((rt*) 0, p, 0);
              }
        }

      }

        //function to return a list of all stored nodes
        list<Nodes> DIM::GetAllSavedNodes()
        {
                  return Nodes;
        }

        //function to find stored packet for a node
        Packet *DIM::FindPacket(int ip, Packet *p)
        {
                  Node *n = IsExistingNode(ip);
                  for (list<Packet>::iterator it = n->packets.begin(); it != n->packets.end();
++it)
                  {
                            if (memcmp(p, it, sizeof(p)))
                            {
                                      return it;
                            }
                  }
        }

        // to save packets that need forwarding by a node
        void DIM::NodeReceivedPacketToForward(int ip, Packet *p)
        {
                  Node *n = IsExistingNode(ip);
                  packet pk = (packet *) (malloc(sizeof (packet)))) < (void *)0;
                  pk->time = pktime;
```

```
			pk->data = (char *)malloc(sizeof(char) * data_size);
			pk->ip_src = ip_src;
			pk->ip_dst = ip_dst;
			pk->data_size = data_size;
			pk->time = pktime;
			pk->originPort = originPort;
			pk->destinationPort = destinationPort;
			pk->isReceived = true;
			pk->isForwarded = false;
			pk->isModified = false;
			pk->isToForward = true;
			n->packets.push_back(p);
		}

		// function to check if a node forwarded packets
		void DIM::NodeForwardedPacket(int ip, Packet *p)
		{
			packet foundPacket = FindPacket(ip, p);
			if (foundPacket != NULL)
			{
				foundPacket->isForwarded = true;
			}
			else if (strcmp(p->body, exists->body) != 0 || p->ip_src != exists->ip_src || p-
>ip_dst != exists->ip_dst)
			{
				foundPacket->isModified = true;
			}
		}

//*********************************************************************
// LDM Component
//*********************************************************************

//LDM.h
#ifndef __LDM_H__
#define __LDM_H__
#include "global.h"
#include "CDM.h"
#include "DSTHandler.h"

class LDM
{
	void PerformAttacksDetection(int currentIP, double current_time);
	AttackDSTValues *DetectAttacks(int currentIP, Node *n, double current_time, bool
isCDMRequest, AttackType attackType);
```

```
	AttackDSTValues *CalculateDSTValuesForSuspect(int currentIP, int suspectip, 
AttackType _attackType, double current_time);
}

//LDM.cpp

//function to iterate over observed nodes and calls attack detection method
void
LDM::PerformAttacksDetection(int currentIP, double current_time)
{
	for (list<Node>::iterator nd = Nodes.begin(); it != Nodes.end(); ++nd)
	{
		if (nd->isMalicious == false)
		{
			DetectAttacks(currentIP, nd, current_time, false, None); //dont worry 
about attack type, its not regard here
		}
	}

	Scheduler::instance().schedule(this, &intr, LDM_DELAY_INTERVAL);
}

//function to iterate over data collected against a node and calculate DST values
AttackDSTValues *LDM::DetectAttacks(int currentIP, Node *n, double current_time, bool 
isCDMRequest, AttackType attackType)
{
	int rushing_totalFastPacketsCounter = 0;
	int rushing_totalAveragePacketsCounter = 0;
	int flooding_largePacketsCounter = 0;
	int flooding_averagePacketsCounter = 0;
	double overallRreqDeliveryTime = 0;
	double overallAveragePacketSizeOfAll = 0;
	int bhole_totalDropped = 0;
	int bhole_totalForwarded = 0;
	int bhole_totalReceived = 0;
	int ghole_totalDropped = 0;
	int ghole_totalForwarded = 0;
	int ghole_totalReceived = 0;
	int totalModified = 0;
	int modifications_totalReceived = 0;
	double current_a_rreq = 0;
	double current_a_size =  0;
	double sum = 0;
	int totalNodes = 0;
	for (list<Node>::iterator nd = Nodes.begin(); it != Nodes.end(); ++nd)
	{
		if (nd->totalRreqPackts > 0)
		{
			sum=0;
			for (int n : nd->listOfPacketDeliveryTimes){
				sum += n;}
			current_a_rreq = sum / nd->listOfPacketDeliveryTimes.size();
		}
		if (nd->listOfPacketSizes.size() > 0)
		{
			sum = 0;
			for (int n : nd->listOfPacketSizes){
				sum += n;}
			current_a_size = sum / nd->listOfPacketSizes.size();
		}
		totalNodes++;
	}
	overallAveragePacketSizeOfAll = overallAveragePacketSizeOfAll + current_a_rreq /
totalNodes;
	overallRreqDeliveryTime = overallRreqDeliveryTime + current_a_size / totalNodes;

	if (n->listOfPacketDeliveryTimes.size() > 0)
	{

		for (list<double>::iterator it = n->listOfPacketDeliveryTimes.begin(); it != n-
>listOfPacketDeliveryTimes.end(); ++it)
		{
			if (*it < overallRreqDeliveryTime && *it <=
RUSHING_MAX_DELIVERY_TIME)
			{
				rushing_totalFastPacketsCounter =
rushing_totalFastPacketsCounter + 1;
			}
			else
			{
				rushing_totalAveragePacketsCounter =
rushing_totalAveragePacketsCounter + 1;
			}
		}
	}

	if (n->listOfPacketSizes.size() > 0)
	{
		for (list<double>::iterator it = n->listOfPacketSizes.begin(); it != n-
>listOfPacketSizes.end(); ++it)
		{
```

```
                        if (*it > overallAveragePacketSizeOfAll && *it >=
FLOODING_MAX_PACKET_SIZE)
                        {
                                flooding_largePacketsCounter = flooding_largePacketsCounter +
1;
                        }
                        else
                        {
                                flooding_averagePacketsCounter =
flooding_averagePacketsCounter + 1;
                        }
                }
        }

        for (list<Packet>::iterator p = n->packets.begin(); it != n->packets.end(); ++p)
        {
                if (p->isToForward)
                {
                        if (p->time < current_time && p->time > (current_time -
(BHOLE_COLLECTION_INTERVAL * LDM_DELAY_INTERVAL)))
                        {

                                bhole_totalReceived = bhole_totalReceived + 1;
                                if (p->isForwarded == false)
                                {
                                        bhole_totalDropped = bhole_totalDropped + 1;
                                }
                                else if (p->isForwarded == true)
                                {

                                        bhole_totalForwarded = bhole_totalForwarded + 1;
                                }
                        }

                        if (p->time < current_time && p->time > (current_time -
(GHOLE_COLLECTION_INTERVAL * LDM_DELAY_INTERVAL)))
                        {

                                ghole_totalReceived = ghole_totalReceived + 1;
                                if (p->isForwarded == false)
                                {

                                        ghole_totalDropped = ghole_totalDropped + 1;
                                }
                                else if (p->isForwarded == true)
                                {
                                        ghole_totalForwarded = ghole_totalForwarded + 1;
                                }
                        }

                        if (p->time < current_time && p->time > (current_time -
LDM_DELAY_INTERVAL))
                        {
                                modifications_totalReceived = modifications_totalReceived + 1;
                                if (p->isModified == true)
                                {
                                        totalModified = totalModified + 1;
                                }
                                else
                                {
                                        totalUnmodified = totalUnmodified + 1;
                                }
                        }
                }
        }

        double blackHoleThreshold = BLACKHOLE_THRESHOLD *
(double)bhole_totalReceived;
        AttackDSTValues *dst_blackhole;
        = DSTHandler->CalculateDSTValues(currentIP, n->ip, bhole_totalForwarded,
                                                                        bhole_totalReceived,
bhole_totalDropped, blackHoleThreshold, Blackhole, !isCDMRequest);

        double grayHoleThreshold = GRAYHOLE_THRESHOLD *
(double)ghole_totalReceived;
        AttackDSTValues *dst_grayhole = DSTHandler->CalculateDSTValues(currentIP, n->ip,
ghole_totalForwarded,

                          ghole_totalReceived, ghole_totalDropped, grayHoleThreshold,
Grayhole, !isCDMRequest);

        double modificationThreshold = MODIFICATION_THRESHOLD *
(double)modifications_totalReceived;
        AttackDSTValues *dst_modification = DSTHandler->CalculateDSTValues(currentIP, n-
>ip, totalUnmodified,

                                  modifications_totalReceived, totalModified,
modificationThreshold, Modification, !isCDMRequest);

        AttackDSTValues *dst_rushing = DSTHandler->CalculateDSTValues(currentIP, n->ip,
rushing_totalAveragePacketsCounter,
                                n->totalRreqPackts, rushing_totalFastPacketsCounter,
rushing_totalAveragePacketsCounter, Rushing, !isCDMRequest);

          AttackDSTValues *dst_flooding = CDSTHandler->CalculateDSTValues(currentIP, n-
>ip, flooding_averagePacketsCounter,

                                        n->listOfPacketSizes.size(), flooding_largePacketsCounter,
flooding_averagePacketsCounter, Flooding, !isCDMRequest);

          if (isCDMRequest == true)
          {
                    switch (attackType)
                    {
                    case Blackhole:
                              return dst_blackhole;
                              break;

                    case Grayhole:
                              return dst_grayhole;
                              break;

                    case Modification:
                              return dst_modification;
                              break;

                    case Rushing:
                              return dst_rushing;
                              break;

                    case Flooding:
                              return dst_flooding;
                              break;
                    }
          }
          return NULL;
}

//function to reply to a strong-tie requests with information about a suspect node
AttackDSTValues *LDM::CalculateDSTValuesForSuspect(int currentIP, int suspectip,
AttackType _attackType, double current_time)
{
          Node *n = DIM->IsExistingNode(suspectip);
          AttackDSTValues *_dstNewCalcs = new AttackDSTValues();
          if (n != NULL)
          {
                _dstNewCalcs = DetectAttacks(currentIP, n, current_time, true, _attackType);
        }
        return _dstNewCalcs;
}

//***********************************************************************
// STBM Component
//***********************************************************************
//STBM.h
#ifndef __LDM_H__
#define __LDM_H__
#include "global.h"
#include "DIM.h";

class STBM
{
        list<int> strongTiesAddresses;

        void CalculateTies(int currentIP);
        void removeTieAndMarkMalicious(int ip);
        void removeTieAndMarkMalicious(int ip);
        bool isSourceMalicious(int ip);

}

//STBM.cpp

//function to calculate strong-ties
void
STBM::CalculateTies(int currentIP)
{
        list<Nodes> Nodes = DIM->GetAllSavedNodes();
        for (list<Node>::iterator n = Nodes.begin(); it != Nodes.end(); ++n)
        {
                double R_RECENCY = Scheduler::instance().clock() - n-
>time_of_last_communication;
                double R_RSH = n->DATA_TOTAL + n->RREQ_TOTAL + n-
>RREP_TOTAL;
double comm_history = Scheduler::instance().clock() - n->time_of_first_communcation;

                if (R_RECENCY < THRESHOLD_RECENCY_THRESHOLD && R_RSH >
THRESHOLD_RECIPROCITY_SHARED_THRESHOLD && comm_history >
THRESHOLD_TIME_SINCE_FIRST_MESSAGE && n->isMalicious == false)
                {
                        //new strong-tie
                        n->isStrongTie = true;
```

```
                    strongTiesAddresses.push_back(n->ip);
               }
               else
               {
                    //remove an existing tie if it doesn't fullfill the condition
                    n->isStrongTie = false;
                    strongTiesAddresses.remove(n->ip);
               }
          }
          Scheduler::instance().schedule(this, &intr, STT_DELAY_INTERVAL);
}

//function to remove a strong-tie and mark it as malicious based on CDM/alarms
void STBM::removeTieAndMarkMalicious(int ip)
{
          Node *n = DIM->IsExistingNode(ip);
          if (n != NULL)
          {
               if (n->ip == ip)
               {
                    //remove tie;
                    n->isStrongTie = false;
                    n->isMalicious = true;
                    break;
               }
          }
}

//function to check if a node is already marked as malicious
bool isSourceMalicious(int ip)
{
          Node *n = DIM->IsExistingNode(ip);
          return n->isMalicious;
}

//function to return an IP list of my strong-ties
list<int> DIM::GetStrongTiesIPAddresses(int currentIP)
{
          strongTiesAddresses.sort();
          strongTiesAddresses.unique();
          return strongTiesAddresses;
}

//*********************************************************************
// DST-related Calculations
//*********************************************************************
```

```
//DSTHandler.h
#ifndef __DSTHandler_H__
#define __DSTHandler_H__
#include "global.h"
#include "STBM"
#include "CDM"
#include "DIM"

class DSTHandler
{
   DSTHandler();
   AttackDSTValues CalculateDSTValues(int currentIP, int suspectIP, double
nonMaliciousNominator,
                              double nonMaliciousDenominator, double maliciousNominator, double
threshold, AttackType attackType, bool doSendCDMRequest);
   CDMReply CombineAndReturnBelief(CDMReply A, CDMReply B);
}

//DSTHandler.cpp

//function to calcualte DST values for attack
AttackDSTValues *
DSTHandler::CalculateDSTValues(int currentIP, int suspectIP, double nonMaliciousNominator,
                        double nonMaliciousDenominator, double maliciousNominator, double
threshold, AttackType attackType, bool doSendCDMRequest)
{
   float nonMalicious = nonMaliciousNominator / nonMaliciousDenominator;
   float malicious = maliciousNominator / threshold;
   float uncertainty = 0;
   if (malicious < nonMalicious)
   {
      uncertainty = (malicious * 0.5) / nonMalicious;
   }
   else
   {
      uncertainty = (nonMalicious * 0.5) / malicious;
   }

   //calculate adjustment factor
   float adjuster = ((malicious + nonMalicious + uncertainty) - 1) / 3;

   //reclaculate DST by subtracting adjuster from each
   malicious = malicious - adjuster;
   nonMalicious = nonMalicious - adjuster;
   uncertainty = uncertainty - adjuster;
```

```
    AttackDSTValues *_dstNewCalcs = new AttackDSTValues();
    _dstNewCalcs->malicious = malicious;
    _dstNewCalcs->nonMalicious = nonMalicious;
    _dstNewCalcs->uncertain = uncertainty;
    _dstNewCalcs->attackType = attackType;

    int totalStrongTies = STBM->GetStrongTiesIPAddresses(currentIP).size();
    if (malicious > nonMalicious && doSendCDMRequest == true && maliciousNominator >
threshold && DIM->IsExistingNode(suspectIP).packets.size() >
MINIMUM_PACKETS_COLLECTED && totalStrongTies > MINIMUM_STRONG_TIES)
    {
        CDM->CreateNewCDMRequest(currentIP, suspectIP, attackType);
    }

    return _dstNewCalcs;
}

//function to perform DST combinatations
CDMReply DSTHandler::CombineAndReturnBelief(CDMReply A, CDMReply B)
{
    //Node A
    double mA_C = A.isNonMalicious;
    double mA_S = A.isMalicious;
    double mA_U = A.isUncertain;

    //node B
    double mB_C = B.isNonMalicious;
    double mB_S = B.isMalicious;
    double mB_U = B.isUncertain;

    //combine AB mA+mAB (orthogonal)

    double K_AB = mA_C * mB_C + mA_C * mB_U + mA_U * mB_C + mA_S * mB_S +
mA_S * mB_U + mA_U * mB_S + mA_U * mB_U;

    //calculate final belief
    double mAB_C = (mA_C * mB_C + mA_C * mB_U + mA_U * mB_C) / K_AB;
    double mAB_S = (mA_S * mB_S + mA_S * mB_U + mA_U * mB_S) / K_AB;
    double mAB_U = (mA_U * mB_U) / K_AB;

    CDMReply *_returnReply = new CDMReply();
    _returnReply->replyingNodeIP = A.replyingNodeIP;
    _returnReply->suspectNodeIP = B.replyingNodeIP; //this assignment is just for checking
    _returnReply->isNonMalicious = mAB_C;
    _returnReply->isMalicious = mAB_S;
    _returnReply->isUncertain = mAB_U;
```

```
   return *_returnReply;
}

//***********************************************************************
// CDM Component
//***********************************************************************

//CDM.h
#ifndef __CDM_H__
#define __CDM_H__
#include "global.h"
#include "STBM.h"

class CDM
{
   list<CDMRequest> CDMStorageDictionary;
   list<CDMRequest> CDMRequestsAlreadySent;
   list<int> CDMReuqestsAlreadyRepliedTo;
   list<CDMRequest> CDMFullfilledRequestsDictionary;
   list<CDMReply> CDMReplyDictionary;

   void ActivateCDM(int currentIP);
   void CreateNewCDMRequest(int currentIP, int maliciousNodeAddress, AttackType 
attackType);
   void AddNewCDMReply(int currentIP, int requestId, int initiaingNodeIP, int suspectNodeIP, 
CDMReply *reply);
   void recvCDMReq(Packet *p);
   void recvCDMRep(Packet *p);
   void sendCDMRequest(CDMRequest request, int dst);
   void sendCDMReply(CDMReply reply, int dst);
   bool IDSCheckIfCDMRequestAlreadySent(int reqId);
   bool isCDMRequestFullfilled(int reqId);
   bool checkIfAlreadyRepliedToCDMReq(int reqId);
}

//CDM.cpp

//this function is called from LDM to activate CDM
void
CDM::ActivateCDM(int currentIP)
{
   list<CDMRequest> cdmStorage = CDMStorageDictionary;

   if (cdmStorage.empty() == 0)
   {
        list<CDMRequest>::iterator req;
        for (req = cdmStorage.begin(); req != cdmStorage.end(); ++req)
        {
           bool isFullfilled = isCDMRequestFullfilled(req->reqId);
           bool isAlreadySentOut = IDSCheckIfCDMRequestAlreadySent(req->reqId);
           //check if request isFullfilled
           if (isFullfilled == false && isAlreadySentOut == false)
           {
              list<int> myStrongTies = STBM->GetStrongTiesAddresses(index);
              if (!myStrongTies.empty())
              {
                 int cnt = 0;
                 list<int>::iterator tie;
                 for (tie = myStrongTies.begin(); tie != myStrongTies.end(); ++tie)
                 {
                    if (*tie != req->suspectNodeIP)
                    {
                       sendCDMRequest(*req, *tie);
                    }
                    cnt++;
                 }
                 //mark it as sent so I dont keep sending the same request
                 CDMRequestsAlreadySent.push_back(req->reqId);
              }
           }
        }
    } //end if empty
}

//this function is used to create CDM requests
void CDM::CreateNewCDMRequest(int currentIP, int maliciousNodeAddress, AttackType
attackType)
{
    CDMRequest *req = new CDMRequest();
    req->reqId = rand() % 200000000; //   (int)time(NULL);// 11;
    req->suspectedAttack = attackType;
    req->initiatingNodeIP = currentIP;
    req->suspectNodeIP = maliciousNodeAddress;
    req->rq_timestamp = Scheduler::instance().clock();
    req->isFullfilled = false;
    req->totalRequestsSent = STBM->GetStrongTiesIPAddresses(currentIP).size();
    if (req->totalRequestsSent > 0)
    {
        CDMStorageDictionary.push_back(*req);
    }
    ActivateCDM(currentIP);
```

```
}

//function to handle receiving new CDM reply about a detection
void CDM::AddNewCDMReply(int currentIP, int requestId, int initiaingNodeIP, int
suspectNodeIP, CDMReply *reply)
{
   list<CDMRequest> cdmStorage = CDMStorageDictionary;
   //correlate to request
   if (cdmStorage.empty() == 0)
   {
      list<CDMRequest>::iterator req;
      for (req = cdmStorage.begin(); req != cdmStorage.end(); ++req)
      {
         bool isFullfilled = isCDMRequestFullfilled(req->reqId);

         if (isFullfilled == false)
         {
            float majorityFullfilled = (req->totalRequestsSent / STBM-
>GetStrongTiesIPAddresses(currentIP).size()) * 100;
            if (req->reqId == requestId && req->suspectNodeIP == suspectNodeIP)
            {
               CDMReplyDictionary.push_back(*reply);
               //check and see if we have enough replies to make a decision
               list<CDMReply> _foundReplies;
               list<CDMReply>::iterator _savedreply;
               int cct = 0;
               for (_savedreply = CDMReplyDictionary.begin(); _savedreply !=
CDMReplyDictionary.end(); ++_savedreply)
               {
                  if (_savedreply->reqId == requestId && _savedreply->suspectNodeIP ==
suspectNodeIP)
                  {
                     _foundReplies.push_back(*_savedreply);
                     cct++;
                  }
               }

               int totalTies = (STBM->GetStrongTiesIPAddresses(currentIP).size()) - 1;
               double repliesMajorityReceived = ((double)cct / (double)totalTies) * 100; // * 100;

               if (repliesMajorityReceived >= MAJORITY_THRESHOLD)

               {
                  list<CDMReply>::iterator _reply;

                  int cnt = 0;
                  CDMReply previousBelief;
                  CDMReply _previousReply;

                  list<int> processedRepliesIpAddressed;
                  for (_reply = _foundReplies.begin(); _reply != _foundReplies.end(); ++_reply)
                  {

                     if ((_reply->isMalicious == 0 && _reply->isNonMalicious == 0 && _reply-
>isUncertain == 0) || _reply->replyingNodeIP == suspectNodeIP)
                     {
                        continue;
                     }
                     else
                     {
                        int originatingNodeIP = _reply->replyingNodeIP;
                        bool exists = std::find(std::begin(processedRepliesIpAddressed),
                                    std::end(processedRepliesIpAddressed), originatingNodeIP)
!= std::end(processedRepliesIpAddressed);
                        if (exists == false)
                        {

                           if (cnt == 0)
                           {
                              _previousReply = *_reply;
                           }
                           else if (_previousReply.replyingNodeIP != _reply->replyingNodeIP)
                           {
                              previousBelief =
                                 DSTHandler::combineAndReturnBelief(_previousReply, *_reply);
                              _previousReply = previousBelief;
                              processedRepliesIpAddressed.push_back(_reply->replyingNodeIP);
                           }
                           cnt++;
                        }
                     }
                  }

                  double Bel_isMalicious = previousBelief.isMalicious;
                  double Bel_isNonMalicious = previousBelief.isNonMalicious;
                  double Bel_isUncertain = previousBelief.isUncertain;

                  //make a decision
                  Node *n = IsNodeExists(suspectNodeIP);
                  if (Bel_isMalicious > Bel_isNonMalicious && Bel_isMalicious >=
FINAL_DECISION_THRESHOLD)
                  {
                    if (n != NULL && n->isMalicious == false)
                    {
                        CreateNewGRMAlarmEntry(suspectNodeIP);
                        GRM->ActivateGRM(currentIP);
                printf("%d, %d, malicious, %d", currentIP, suspectNodeIP, req-
                >suspectedAttack);
                    }
                    //mark as malicious and remove from strong-ties
                    if (n != NULL)
                    {
                        n->isMalicious = true;
                        n->isStrongTie = false;
                    }
                }
                else
                {
                    //final decision is non-malicious
                    n->isMalicious = false;
                printf("%d, %d, innocent, %d", currentIP, suspectNodeIP, req->suspectedAttack);

                }
            }

            req->isFullfilled = true;
            CDMFullfilledRequestsDictionary.push_back(req->reqId);
        }
    }
  }
 }
}

//function to check if CDM already replied to a received request
bool CDM::checkIfAlreadyRepliedToCDMReq(int reqId)
{
    bool exists = std::find(std::begin(CDMRequestsAlreadyRepliedTo),
                    std::end(CDMRequestsAlreadyRepliedTo), reqId) !=
std::end(CDMRequestsAlreadyRepliedTo);

    return exists;
}

//function to receive CDM Request and calculate DST values, then send reply to originator
void CDM::recvCDMReq(Packet *p)
{
    //get suspectedNode and suspectedAttack
    struct hdr_ids_cdm_req *rq = HDR_IDS_CDM_REQ(p);
    //check if I already replied to that request
    bool alreadyReplied = checkIfAlreadyRepliedToCDMReq(rq->rq_id);

    if (alreadyReplied == false)
    {
        AttackDSTValues *_dstValues =
            LDM->CalculateDSTValuesForSuspect(index, rq->suspect_node, (AttackType)rq-
>attack_type, CURRENT_TIME);

        iDS->IDSAddCDMRequestRepliedTo(rq->rq_id);

        if (_dstValues != nullptr && (_dstValues->malicious + _dstValues->nonMalicious +
_dstValues->uncertain > 0))
        {
            CDMReply *_cdmReply = new CDMReply();
            _cdmReply->reqId = rq->rq_id;
            _cdmReply->repId = rand() % 99999 + 1;
            ;
            _cdmReply->replyingNodeIP = index;
            _cdmReply->suspectNodeIP = rq->suspect_node;
            _cdmReply->suspectedAttack = _dstValues->attackType;
            _cdmReply->rp_timestamp = CURRENT_TIME;
            _cdmReply->isMalicious = _dstValues->malicious;
            _cdmReply->isNonMalicious = _dstValues->nonMalicious;
            _cdmReply->isUncertain = _dstValues->uncertain;
            sendCDMReply(*_cdmReply, rq->rq_src);
        }
    }
}
}

//function to handle receiving CDM reply packets
void CDM::recvCDMRep(Packet *p)
{
    struct hdr_ids_cdm_rep *rep = HDR_IDS_CDM_REP(p);
    CDMReply *_cdmReply = new CDMReply();
    _cdmReply->reqId = rep->rq_id;
    _cdmReply->repId = std::rand();
    _cdmReply->replyingNodeIP = rep->rq_src;
    _cdmReply->suspectNodeIP = rep->suspect_node;
    _cdmReply->rp_timestamp = CURRENT_TIME;
    _cdmReply->isMalicious = rep->isMalicious;
    _cdmReply->isNonMalicious = rep->isNonMalicious;
    _cdmReply->isUncertain = rep->isUncertain;
    AddNewCDMReply(index, rep->rq_id, rep->rq_src, rep->suspect_node, _cdmReply);
}
```

```
//function to send CDM request to target node
void CDM::sendCDMRequest(CDMRequest request, int dst)
{
   Packet *p = Packet::alloc();
   struct hdr_cmn *ch = HDR_CMN(p);
   struct hdr_ip *ih = HDR_IP(p);
   struct hdr_ids_cdm_req *rq = HDR_IDS_CDM_REQ(p);

   ch->size() = IP_HDR_LEN + rq->size();
   ch->iface() = -2;
   ch->error() = 0;
   ch->addr_type() = NS_AF_NONE;
   ch->prev_hop_ = index;
   ch->next_hop_ = dst;

   ih->saddr() = index;
   ih->daddr() = dst;
   ih->sport() = RT_PORT;
   ih->dport() = RT_PORT;

   rq->rq_type = IDS_CDM_REQ;
   rq->suspect_node = request.suspectNodeIP;
   rq->attack_type = request.suspectedAttack;
   rq->rq_timestamp = CURRENT_TIME;
   rq->rq_dst = dst;
   rq->rq_src = index;
   rq->rq_id = request.reqId;

   Scheduler::instance().schedule(target_, p, 0.);
}

//function to send CDM reply to requesting node
void CDM::sendCDMReply(CDMReply reply, int dst)
{
   Packet *p = Packet::alloc();
   struct hdr_cmn *ch = HDR_CMN(p);
   struct hdr_ip *ih = HDR_IP(p);
   struct hdr_ids_cdm_rep *rq = HDR_IDS_CDM_REP(p);

   ch->size() = IP_HDR_LEN + rq->size();
   ch->iface() = -2;
   ch->error() = 0;
   ch->addr_type() = NS_AF_NONE;
   ch->prev_hop_ = index;
   ch->next_hop_ = dst;
```

```
    ih->saddr() = index;
    ih->daddr() = dst;
    ih->sport() = RT_PORT;
    ih->dport() = RT_PORT;

    rq->rq_type = IDS_CDM_REP;
    rq->suspect_node = reply.suspectNodeIP;
    rq->attack_type = reply.suspectedAttack;
    rq->rq_timestamp = CURRENT_TIME;
    rq->rq_dst = dst;
    rq->rq_src = index;
    rq->rq_id = reply.reqId;
    rq->isMalicious = reply.isMalicious;
    rq->isNonMalicious = reply.isNonMalicious;
    rq->isUncertain = reply.isUncertain;
    Scheduler::instance().schedule(target_, p, 0.);
}

//function to check if a CDM request was already sent out to strong-ties
bool CDM::IDSCheckIfCDMRequestAlreadySent(int reqId)
{
    bool exists = std::find(std::begin(CDMRequestsAlreadySent),
                            std::end(CDMRequestsAlreadySent), reqId) !=
std::end(CDMRequestsAlreadySent);

    return exists;
}

//functino to check if a CDM request is fullfilled from a node's strong-ties
bool CDM::isCDMRequestFullfilled(int reqId)
{
    bool isFullfilled = false;

    list<int> currentFullfilledRequests = CDMFullfilledRequestsDictionary;
    if (currentFullfilledRequests.empty() == 0)
    {
        list<int>::iterator _req;
        for (_req = currentFullfilledRequests.begin(); _req != currentFullfilledRequests.end();
++_req)
        {
            if (*_req == reqId)
            {
                isFullfilled = true;
                break;
            }
        }
    }
    return isFullfilled;
}

//**********************************************************************
// GRM Component
//**********************************************************************
//GRM.h
#ifndef __GRM_H__
#define __GRM_H__
#include "global.h"
#include "STBM.h"
#include "DIM.h"

class GRM
{
            list<GRMAlarm> GRMAlarmStorage;

            void ActivateGRM(int currentIP);
            void sendAlarm(nsaddr_t dst, nsaddr_t maliciousNode);
            void recvAlarm(Packet *p);

}

//GRM.cpp

//function to activate GRM based on CDM decisions
void
GRM::ActivateGRM(int currentIP)
{
            //check for pending alarms
            list<GRMAlarm> _allAlarms = GRMAlarmStorage;
            list<GRMAlarm>::iterator _savedAlarm;

            for (_savedAlarm = _allAlarms.begin(); _savedAlarm != _allAlarms.end();
++_savedAlarm)
            {
                        if (_savedAlarm->isSentToAllTies == false)
                        {
                                    //send alarms out and mark them as completed
                                    list<int> myStrongTies = iDS->GetStrongTiesAddresses(index);
                                    if (!myStrongTies.empty())
                                    {
                                                list<int>::iterator tie;
                                                for (tie = myStrongTies.begin(); tie != myStrongTies.end(); ++tie)
```

```
                                        {
                                                sendAlarm(*tie, _savedAlarm->maliciousNodeIP);
                                        }
                                        rtable.rt_delete((int)_savedAlarm->maliciousNodeIP);
                                        //mark as completed
                                        _savedAlarm->isSentToAllTies = true;
                                        _savedAlarm->timeAlarmSent = CURRENT_TIME;
                                }
                        }
                }
}

//function to send alarm to strong-ties
void GRM::sendAlarm(nsaddr_t dst, nsaddr_t maliciousNode)
{
        Packet *p = Packet::alloc();
        struct hdr_cmn *ch = HDR_CMN(p);
        struct hdr_ip *ih = HDR_IP(p);
        struct hdr_ids_alarm *rq = HDR_IDS_ALARM(p);

        ch->size() = IP_HDR_LEN + rq->size();
        ch->iface() = -2;
        ch->error() = 0;
        ch->addr_type() = NS_AF_NONE;
        ch->prev_hop_ = index;
        ch->next_hop_ = dst;

        ih->saddr() = index;
        ih->daddr() = dst;
        ih->sport() = RT_PORT;
        ih->dport() = RT_PORT;

        rq->rq_type = IDS_ALARM;
        rq->rq_dst = dst;
        rq->rq_src = index;
        rq->malicious_node = maliciousNode;
        rq->rq_timestamp = CURRENT_TIME;

        Scheduler::instance().schedule(target_, p, 0.);
}

//function to handle receiving alarm from strong-ties
void GRM::recvAlarm(Packet *p)
{
        struct hdr_ids_alarm *rq = HDR_IDS_ALARM(p);
        //check if alarm is from a strong tie
        Node *n = DIM->IsExistingNode(rq->rq_src);
        if (n != NULL && n->isStrongTie == true && n->isMalicious == false)
        {
                        //mark node as malicious
                        Node *n_malicious = DIM->IsExistingNode(rq->malicious_node);
                        if (n_malicious != NULL)
                        {
                                n_malicious->isMalicious = true;
                                n_malicious->isStrongTie = false;
                                rtable.rt_delete((int)n_malicious->ip);
                                list<int> myStrongTies = STBM->GetStrongTiesAddresses(index);

                                if (!myStrongTies.empty())
                                {
                                        list<int>::iterator tie;
                                        for (tie = myStrongTies.begin(); tie != myStrongTies.end(); ++tie)
                                        {
                                                sendAlarm(*tie, rq->malicious_node);
                                        }
                                }
                        }
        }
}
}
```

References

Abdelaziz, A.K., Nafaa, M., Salim, G. (2013). *Survey of routing attacks and countermeasures. Proceedings of the 15th International Conference on Computer Modelling and Simulation (UKSim)*, 693–698.

Abdelshafy, M. A., & King, P. J. (2013, December). Analysis of security attacks on AODV routing. *Proceedings of the 2013 8th International Conference for Internet Technology and Secured Transactions (ICITST)*, 290-295.

Abolhasan, M., Wysocki, T., & Dutkiewicz, E. (2004). A review of routing protocols for mobile ad hoc networks. *Ad hoc networks*, 2(1), 1-22.

Adhikari, S., & Setua, S. K. (2013, October). Cooperative network intrusion detection system (CNIDS) in mobile adhoc network based on DSR protocol. *Proceedings of the 2013 3rd International Conference on Computer Science and Network Technology* (ICCSNT), 929-935.

Alani, M. M. (2014, November). MANET security: A survey. *Proceedings of the 2014 IEEE International Conference on Computing and Engineering (ICCSCE)*, pp. 559-564.

Alattar, M., Sailhan, F., & Bourgeois, J. (2012, August). Log-based intrusion detection for MANET. *Proceedings of the 2012 8th International Conference on Wireless Communications and Mobile Computing (IWCMC)*, 697-702.

Amouri, A., Jaimes, L. G., Manthena, R., Morgera, S. D., & Vergara-Laurens, I. J. (2015, November). A simple scheme for pseudo clustering algorithm for cross layer intrusion detection in MANET. *Proceedings of the 2015 7th IEEE Latin-American Conference on Communications (LATINCOM)*, 1-6.

Anderson, J. P. (1972). *Computer Security Technology Planning Study*. Ft. Belvoir: Defense Technical Information Center.

Anderson, J. P. (1980). *Computer security threat monitoring and surveillance*. Technical report, Fort Washington, Pennsylvania: James P. Anderson Company.

Anjum, F., Subhadrabandhu, D., & Sarkar, S. (2003, October). Signature based intrusion detection for wireless ad-hoc networks: A comparative study of various routing protocols. *Proceedings of the Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th (3)*, 2152-2156.

Anantvalee, T., & Wu, J. (2007). A survey on intrusion detection in mobile ad hoc networks. *In Wireless Network Security*, 159-180. Springer US.

Aparicio-Navarro, F. J., Kyriakopoulos, K. G., & Parish, D. J. (2012, June). A multi-layer data fusion system for wi-fi attack detection using automatic belief assignment. *Proceedings of the 2012 World Congress on Internet Security (WorldCIS)*, 45-50.

Aziz, B., & Nourdine, E. (2008, April). A recent survey on key management schemes in manet. *Proceedings of ICTTA 2008, 3rd International Conference on Information and Communication Technologies: From Theory to Applications*, 1-6.

Bai, F., Sadagopan, N., & Helmy, A. (2003, March). IMPORTANT: A framework to systematically analyze the Impact of Mobility on Performance of Routing protocols for Adhoc Networks. *Proceedings of the Twenty-second annual joint conference of the IEEE computer and communications (INFOCOM 2003)*, 825-835.

Banerjee, S., Nandi, R., Dey, R., & Saha, H. N. (2015, October). A review on different Intrusion Detection Systems for MANET and its vulnerabilities. *Proceedings of the 2015*

International Conference and Workshop on Computing and Communication (IEMCON), 1-7.

Bansal, S., & Baker, M. (2003). Observation-based cooperation enforcement in ad hoc networks. *Tech. Rep., Stanford University, CA*.

Beijar, N. (2002). Zone routing protocol (ZRP). *Networking Laboratory, Helsinki University of Technology, Finland*, *9*, 1-12.

Berthier, R., & Sanders, W. H. (2011, December). Specification-based intrusion detection for advanced metering infrastructures. *Proceedings of the 2011 IEEE 17th Pacific Rim International Symposium on Dependable Computing (PRDC)*, 184-193.

Beyer, D. A. (1990, September). Accomplishments of the DARPA SURAN Program. Proceedings of the *Military Communications Conference, 1990. MILCOM'90, Conference Record, A New Era*. 855-862.

Biswas, S., Nag, T., & Neogy, S. (2014, February). Trust based energy efficient detection and avoidance of black hole attack to ensure secure routing in MANET. *Proceedings of the 2014 International Conference on Applications and Innovations in Mobile Computing (AIMoC)*, 157-164.

Bharati, T. S., & Kumar, R. (2015). Intrusion Detection System for MANET using Machine Learning and State Transition Analysis. *International Journal of Computer Engineering & Technology (IJCET)*, 6(12), 1-8.

Bose, S., Bharathimurugan, S., & Kannan, A. (2007, February). Multi-layer integrated anomaly intrusion detection system for mobile adhoc networks. *Proceedings of the International*

Conference on Signal Processing, Communications and Networking, 2007. ICSCN'07. 360-365.

Boston, J. R. (2000). A signal detection system based on Dempster-Shafer theory and comparison to fuzzy detection. *IEEE Systems, Man, and Cybernetics*, 30(1), 45-51.

Brutch, P., & Ko, C. (2003, January). Challenges in intrusion detection for wireless ad-hoc networks. *Proceedings of the 2003 Symposium on Applications and the Internet Workshops*, 368-373.

Butun, I., Morgera, S. D., & Sankar, R. (2014). A survey of intrusion detection systems in wireless sensor networks. *IEEE Communications Surveys & Tutorials*, 16(1), 266-282.

Cabrera, J. B., Gutiérrez, C., & Mehra, R. K. (2005, October). Infrastructures and algorithms for distributed anomaly-based intrusion detection in mobile ad-hoc networks. *Proceedings of the 2005 Military Communications Conference (MILCOM 2005)*, 1831-1837.

Cai, C., Ci, S., Guizani, S., & Al-Fuqaha, A. (2006, November). Constructing an efficient mobility profile of ad-Hoc node for mobility-pattern-based anomaly detection in MANET. *Proceedings of the 2006 Global Telecommunications Conference (GLOBECOM'06)*, 1-5.

Cannady, J. (2009). Distributed detection of attacks in mobile ad hoc networks using learning vector quantization. *Proceedings of Third International Conference on Network and System Security (NSS'09)*, 571-574.

Cannady, J. (2010). Dynamic neural networks in the detection of distributed attacks in mobile Ad-Hoc networks. *International Journal of Network Security & Its Application*, 2(1), 1-7.

Cannady, J. (2013). The detection of temporally distributed network attacks using an adaptive hierarchical neural network. *Proceedings of the 2013 World Congress on Nature and Biologically Inspired Computing (NaBIC),* 5-9.

Chlamtac, I., Conti, M., & Liu, J. J. N. (2003). Mobile ad hoc networking: imperatives and challenges. *Ad hoc networks*, 1(1), 13-64.

Conti, M., Gregori, E., & Maselli, G. (2004, March). Cooperation issues in mobile ad hoc networks. *Proceedings of the 24th International Conference on Distributed Computing Systems Workshops*, 803-808.

Corson, M. S., Macker, J. P., & Cirincione, G. H. (1999). Internet-based mobile ad hoc networking. *IEEE internet computing*, 3(4), 63-70.

Debar, H., Dacier, M., & Wespi, A. (1999). Towards a taxonomy of intrusion-detection systems. *Computer Networks*, *31*(8), 805-822.

Denning, D. E. (1987). An intrusion-detection model. *IEEE Transactions on software engineering*, (2), 222-232.

Denning, D. E., & Neumann, P. G. (1985). Requirements and model for IDES—a real-time intrusion detection expert system. *Document A005, SRI International*, 333.

Deng, H., Li, W., & Agrawal, D. P. (2002). Routing security in wireless ad hoc networks. *IEEE Communications magazine*, 40(10), 70-75.

Deng, H., Xu, R., Li, J., Zhang, F., Levy, R., & Lee, W. (2006). Agent-based cooperative anomaly detection for wireless ad hoc networks. *Proceedings of the 12th International Conference on Parallel and Distributed Systems (ICPADS)*, 613-620.

Ding, H., & Xu, X. (2006, November). Real-time cooperation intrusion detection system for MANets. *Proceedings of the 2006 IET International Conference on Wireless, Mobile and Multimedia Networks*, 1-4.

Djenouri, D., Khelladi, L., & Badache, N. (2005). A survey of security issues in mobile ad hoc networks. *IEEE communications surveys*, 7(4), 2-28.

Dorri, A., Kamel, S. R., & Kheirkhah, E. (2015). Security challenges in mobile ad hoc networks: A survey. International Journal on Computer Science and Engineering, 6(1), 15-29.

Ebinger, P., & Bibmeyer, N. (2009, May). TEREC: Trust evaluation and reputation exchange for cooperative intrusion detection in MANETs. *Proceedings of the Seventh Annual Communication Networks and Services Research Conference (CNSR'09)*, 378-385.

El Defrawy, K., & Tsudik, G. (2008, October). Prism: Privacy-friendly routing in suspicious manets (and vanets). *Proceedings of the IEEE International Conference on Network Protocols (ICNP 2008)*, 258-267.

Farhan, A. F., Zulkhairi, D., & Hatim, M. T. (2008, September). Mobile agent intrusion detection system for mobile ad hoc networks: A non-overlapping zone approach. *Proceedings of the 4th IEEE/IFIP International Conference on Internet (ICI 2008)*, 1-5.

Forrest, S., Hofmeyr, S. A., & Somayaji, A. (1997). Computer immunology. *Communications of the ACM*, 40(10), 88-96.

Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1), 18-28.

Garg, S., & Chand, S. (2014, September). Enhanced AODV protocol for defense against Jellyfish Attack on MANETs. *Proceedings of the 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2279-2284.

Garg, N., & Mahapatra, R. P. (2009). MANET security issues. International Journal of Computer Science and Network Security (*IJCSNS)*, *9*(8), 241.

Garvey, T. D., & Lunt, T. F. (1991, October). Model-based intrusion detection. In *Proceedings of the 14th National Computer Security Conference, 372-385.*

Goldsmith, A. J., & Wicker, S. B. (2002). Design challenges for energy-constrained ad hoc wireless networks. *IEEE wireless communications*, *9*(4), 8-27.

Gomez, J., & Campbell, A. T. (2007). Variable-range transmission power control in wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 6(1), 87-99.

Goyal, P., Parmar, V., & Rishi, R. (2011). Manet: vulnerabilities, challenges, attacks, application. *International Journal of Computational Engineering & Management (IJCEM)*, 11(2011), 32-37.

Hegland, A. M., Winjum, E., Mjolsnes, S. F., Rong, C., Kure, O., & Spilling, P. (2006). A survey of key management in ad hoc networks. *IEEE Communications Surveys & Tutorials*, 8(3), 48-66.

Hernandez-Orallo, E., Serrat, M. D., Cano, J. C., Calafate, C. T., & Manzoni, P. (2012). Improving selfish node detection in MANETs using a collaborative watchdog. *IEEE Communications letters*, 16(5), 642-645.

Hong, X., Xu, K., & Gerla, M. (2002). Scalable routing protocols for mobile ad hoc networks. *IEEE network*, 16(4), 11-21.

Hu, J., & Burmester, M. (2009). Cooperation in mobile ad hoc networks. *In Guide to Wireless Ad Hoc Networks*, 43-57. Springer London.

Hu, Y. C., Perrig, A., & Johnson, D. B. (2003, September). Rushing attacks and defense in wireless ad hoc network routing protocols. *Proceedings of the 2nd ACM workshop on Wireless security*, 30-40.

Huang, Y. A., & Lee, W. (2003, October). A cooperative intrusion detection system for ad hoc networks. *Proceedings of the 1st ACM workshop on Security of Ad Hoc and Sensor Networks,* 135-147.

Hubaux, J. P., Buttyán, L., & Capkun, S. (2001, October). The quest for security in mobile ad hoc networks. *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, 146-155.

Huhns, M. N. (1999). Networking embedded agents. *IEEE Internet Computing*, *3*(1), 91-93.

Husain, S., Gupta, S. C., Chand, M., & Mandoria, H. L. (2010, September). A proposed model for Intrusion Detection System for mobile adhoc network. *Proceedings of the 2010 International Conference on Computer and Communication Technology (ICCCT)*, 99-102).

Ilgun, K., Kemmerer, R. A., & Porras, P. A. (1995). State transition analysis: A rule-based intrusion detection approach. *IEEE transactions on software engineering*, *21*(3), 181-199.

Jacoby, G. A., & Davis, N. J. (2007). Mobile host-based intrusion detection and attack identification. *IEEE Wireless Communications*, 14(4).

Jawandhiya, P. M., Ghonge, M. M., Ali, M. S., & Deshpande, J. S. (2010). A survey of mobile ad hoc network attacks. *International Journal of Engineering Science and Technology*, 2(9), 4063-4071.

Johansson, P., Larsson, T., Hedman, N., Mielczarek, B., & Degermark, M. (1999, August). Scenario-based performance analysis of routing protocols for mobile ad-hoc networks. *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking,* 195-206.

Joseph, J. F. C., Lee, B. S., Das, A., & Seet, B. C. (2011). Cross-layer detection of sinking behavior in wireless ad hoc networks using SVM and FDA. *IEEE Transactions on Dependable and Secure Computing*, 8(2), 233-245.

Jubin, J., & Tornow, J. D. (1987). The DARPA packet radio network protocols. *Proceedings of the IEEE*, 75(1), 21-32.

Justin, V., Marathe, N., & Dongre, N. (2017, February). Hybrid IDS using SVM classifier for detecting DoS attack in MANET application. *Proceedings of the 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC),* 775-778.

Kannammal, A., & Roy, S. S. (2016, March). Survey on secure routing in mobile ad hoc networks. *Proceedings of the 2016 International Conference on Advances in Human Machine Interaction (HMI)*, 1-7.

Katal, A., Wazid, M., Sachan, R. S., Singh, D. P., & Goudar, R. H. (2013, December). Effective Clustering Technique for Selecting Cluster Heads and Super Cluster Head in MANET. *Proceedings of the 2013 International Conference on Machine Intelligence and Research Advancement (ICMIRA),* 1-6.

Karygiannis, A., Antonakakis, E., & Apostolopoulos, A. (2006, June). Detecting critical nodes for MANET intrusion detection systems. *Proceedings of the 2006 Second International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing (SecPerU 2006), 1*-9.

Kashyap, N. (2015, March). Smart intrusion detection system for MANET. *Proceedings of the 2015 International Conference on Advances in Computer Engineering and Applications (ICACEA)*, 252-177.

Kathiravelu, T., & Sivasuthan, S. (2011, August). A hybrid reactive routing protocol for Mobile Ad-hoc Networks. *Proceedings of the 2011 6th IEEE International Conference on Industrial and Information Systems (ICIIS)*, 222-227.

Kerrache, C. A., Lupia, A., De Rango, F., Calafate, C. T., Cano, J. C., & Manzoni, P. (2017, June). An energy-efficient technique for MANETs distributed monitoring. *Proceedings of the 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, 1274-1279.

Kheyri, D., & Karami, M. (2012). A comprehensive survey on anomaly-based intrusion detection in MANET. *Computer and Information science*, *5*(4), 132-137.

Kim, J. M., & Jang, J. W. (2006, February). AODV based energy efficient routing protocol for maximum lifetime in MANET. *Proceedings of the 2006 Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT-ICIW'06)*, 77.

Kim, H., Kim, D., & Kim, S. (2006). Lifetime-enhancing selection of monitoring nodes for intrusion detection in mobile ad hoc networks. *AEU-International Journal of Electronics and Communications*, *60*(3), 248-250.

Ko, C., Ruschitzka, M., & Levitt, K. (1997, May). Execution monitoring of security-critical programs in distributed systems: A specification-based approach. *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, 175-187.

Kumar, S. M., Vijay, K. A., & Suhas, N. S. (2016, May). A policy based preventive measure against flooding attack in MANETs. *Proceedings of the IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, 1612-1616.

Kumar, S., & Spafford, E. H. (1994). A pattern matching model for misuse intrusion detection. *Proceedings of the National Computer Security Conference*, 11–21.

Lee, S. J., & Gerla, M. (2000). AODV-BR: Backup routing in ad hoc networks. *Proceedings of the 2000 IEEE Wireless Communications and Networking Conference (WCNC)*, 1311-1316.

Li, W., & Joshi, A. (2009, May). Outlier detection in ad hoc networks using dempster-shafer theory. *Proceedings of the Tenth International Conference on Mobile Data Management: Systems, Services and Middleware (MDM'09)*, 112-121.

Lauf, A. P., Peters, R. A., & Robinson, W. H. (2010). A distributed intrusion detection system for resource-constrained devices in ad-hoc networks. *Ad Hoc Networks*, 8(3), 253-266.

Leiner, B. M., Ruther, R. J., & Sastry, A. R. (1996). Goals and challenges of the DARPA GloMo program [global mobile information systems]. *IEEE Personal Communications*, 3(6), 34-43.

Liu, K., Deng, J., Varshney, P. K., & Balakrishnan, K. (2007). An acknowledgment-based approach for the detection of routing misbehavior in MANETs. *IEEE transactions on mobile computing*, *6*(5), 536–550.

Lunt, T. F. (1993). A survey of intrusion detection techniques. *Computers & Security*, 12(4), 405-418.

Lunt, T. F., & Jagannathan, R. (1988, April). A prototype real-time intrusion-detection expert system. *Proceedings of the 1988 IEEE Symposium on Security and Privacy*, 59-66.

Lunt, T., Jagannathan, R., Lee, R., Listgarten, S., Edwards, D. L., Neumann, P. G., Havitz, H.S., & Valses, A. (1988). IDES: The enhanced prototype. *SRI International, SRI-CSL-88-12*.

Lunt, T. F., Jagannathan, R., Lee, R., Whitehurst, A., & Listgarten, S. (1989, March). Knowledge-based intrusion detection. *Proceedings of the Annual AI Systems in Government Conference*, 102-107.

Lupia, A., & Marano, S. (2016, July). A dynamic monitoring for energy consumption reduction of a trust-based intrusion detection system in mobile Ad-hoc networks. *Proceedings of the 2016 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, 1-5.

Ma, C. X., & Fang, Z. M. (2009, January). A novel intrusion detection architecture based on adaptive selection event triggering for mobile ad-hoc networks. *Proceedings of the Second International Symposium on Intelligent Information Technology and Security Informatics (IITSI'09)*, 198-201.

Mahmood, R. A., Amin, A. H. M., Amir, A., & Khan, A. I. (2009, December). Lightweight and distributed attack detection scheme in mobile ad hoc networks. *Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia*, 62-169.

Mahmoud, M. E., & Shen, X. (2010, March). Stimulating cooperation in multi-hop wireless networks using cheating detection system. *Proceedings of the 2010 IEEE INFOCOM, ,1-9.*

Malek, S. F., & Khorsandi, S. (2013, May). A cooperative intrusion detection algorithm based on trusted voting for mobile ad hoc network. *Proceedings of the 2013 21st Iranian Conference on Electrical Engineering (ICEE)*, 1-8.

Maleki, M., Dantu, K., & Pedram, M. (2002). Power-aware source routing protocol for mobile ad hoc networks. *Proceedings of the 2002 International Symposium on Low Power Electronics and Design (ISLPED'02)*, 72-75.

Mandalas, K., Flitzanis, D., Marias, G. F., & Georgiadis, P. (2005, December). A survey of several cooperation enforcement schemes for MANETs. *Proceedings of the Fifth IEEE International Symposium on Signal Processing and Information Technology*, 466-471.

Manikopoulos, C., & Ling, L. (2003, October). Architecture of the mobile ad-hoc network security (MANS) system. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 3122-3127.

Manousakis, K., Sterne, D., Ivanic, N., Lawler, G., & McAuley, A. (2008, November). A stochastic approximation approach for improving intrusion detection data fusion structures. *Proceedings of the Military Communications Conference (MILCOM 2008), IEEE*, 1-7.

Mapanga, I., Kumar, V., Makando, W., Kushboo, T., Kadebu, P., & Chanda, W. (2017). Design and Implementation of an Intrusion Detection System using MLP-NN for MANET. *Proceedings of the 2017 IST-Africa Week Conference (IST-Africa)*, 1-12.

Marias, G. F., Georgiadis, P., Flitzanis, D., & Mandalas, K. (2006). Cooperation enforcement schemes for MANETs: A survey. *Wireless Communications and Mobile Computing*, 6(3), 319-332.

McDonald, A. B., & Znati, T. F. (1999). A mobility-based framework for adaptive clustering in wireless ad hoc networks. *IEEE Journal on Selected Areas in communications*, *17*(8), 1466-1487.

Mikki, M. A. (2009). Energy efficient location aided routing protocol for wireless MANETs. *International Journal of Computer Science and Information Security, 4(1-2).*

Mishra, A., Nadkarni, K., & Patcha, A. (2004). Intrusion detection in wireless ad hoc networks. *IEEE wireless communications*, 11(1), 48-60.

Mitchell, R., & Chen, R. (2014). A survey of intrusion detection in wireless network applications. *Computer Communications*, *42*, 1-23.

Morais, A., & Cavalli, A. (2012, March). A distributed intrusion detection scheme for wireless ad hoc networks. *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, 556-562.

Mustafa, H., & Xiong, Y. (2013, May). Routing attacks detection and reaction scheme for mobile ad hoc networks using statistical methods. *Proceedings of the 22nd Wireless and Optical Communication Conference (WOCC)*, 659-664.

Nadeem, A., & Howarth, M. P. (2013). A survey of manet intrusion detection & prevention approaches for network layer attacks. *IEEE Communications surveys and tutorials*, 15(4), 2027-2045.

Nadkarni, K., & Mishra, A. (2004, March). A novel intrusion detection approach for wireless ad hoc networks. *Proceedings of the Wireless Communications and Networking Conference, (WCNC), IEEE*, 831-836.

Otrok, H., Debbabi, M., Assi, C., & Bhattacharya, P. (2007, June). A cooperative approach for analyzing intrusions in mobile ad hoc networks. *Proceedings of the 27th International Conference on Distributed Computing Systems Workshops (ICDCSW'07)*, 86-86.

Otrok, H., Mohammed, N., Wang, L., Debbabi, M., & Bhattacharya, P. (2007, October). An efficient and truthful leader IDS election mechanism for MANET. *Proceedings of the 3rd IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2007)*, 78-78.

Panos, C., Xenakis, C., & Stavrakakis, I. (2010, July). A novel intrusion detection system for MANETs. *Proceedings of the 2010 International Conference on Security and Cryptography (SECRYPT)*, 1-10.

Papadimitratos, P., & Haas, Z. J. (2004). Securing Mobile Ad Hoc Networks. *Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, 1-13.

Patel, D. N., Patel, S. B., Kothadiya, H. R., Jethwa, P. D., & Jhaveri, R. H. (2014, February). A survey of reactive routing protocols in MANET. *Proceedings of the 2014 International Conference on Information Communication and Embedded Systems (ICICES)*, 1-6.

Patcha, A., & Mishra, A. (2003, August). Collaborative security architecture for black hole attack prevention in mobile ad hoc networks. *Proceedings of the Radio and Wireless Conference (RAWCON'03)*, 75-78.

Prasannavenkatesan, T., Raja, R., & Ganeshkumar, P. (2014, April). PDA-misbehaving node detection & prevention for MANETs. Proceedings of the 2014 International Conference on Communications and Signal Processing (ICCSP), 1163-1167.

Rai, A. K., Tewari, R. R., & Upadhyay, S. K. (2010). Different types of attacks on integrated MANET-Internet communication. *International Journal of Computer Science and Security*, 4(3), 265-274.

Raj, N., Bharti, P., & Thakur, S. (2015, April). Vulnerabilities, Challenges and Threats in Securing Mobile Ad-Hoc Network. *Proceedings of the 2015 Fifth International Conference on Communication Systems and Network Technologies (CSNT)*, 771-775.

Rajakumar, P., Prasanna, V. T., & Pitchaikkannu, A. (2014, February). Security attacks and detection schemes in MANET. *Proceedings of the 2014 International Conference on Electronics and Communication Systems (ICECS)*, 1-6.

Razak, S. A., Samian, N., & Maarof, M. A. (2008, September). A friend mechanism for mobile ad hoc networks. Proceedings of the Fourth International Conference on Information Assurance and Security (ISIAS'08), 243-248.

Rezaul Karim, A. H. M., Rajatheva, R. M. A. P., & Ahmed, K. M. (2006, October). An efficient collaborative intrusion detection system for MANET using Bayesian Approach. *Proceedings of the 9th ACM international symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems*, 187-190.

Roy, D. B., Chaki, R., & Chaki, N. (2009). A new cluster-based wormhole intrusion detection algorithm for mobile ad-hoc networks. *International journal of network security and its application*,1(1), 44-52.

Royer, E. M., & Toh, C. K. (1999). A review of current routing protocols for ad hoc mobile wireless networks. *IEEE personal communications*, *6*(2), 46-55.

Sabat, S., & Kadam, S. (2014, April). Adaptive Energy aware reputation based leader election for IDS in MANET. *Proceedings of the 2014 International Conference on Communications and Signal Processing (ICCSP)*, 891-894.

Saeed, N. H., Abbod, M. F., & Al-Raweshidy, H. S. (2012, April). MANET routing protocols taxonomy. *Proceedings of the 2012 International Conference on Future Communication Networks (ICFCN)*, 123-128.

Sangeetha, V., & Kumar, S. S. (2018, January). Detection of malicious node in mobile ad-hoc network. *Proceedings of the 2018 International Conference on Power, Signals, Control and Computation (EPSCICON)*, 1-3.

Sarkar, M., & Roy, D. B. (2011, April). Prevention of sleep deprivation attacks using clustering. *Proceedings of the 2011 3rd International Conference on Electronics Computer Technology (ICECT)*, 391-394.

Schaumann, J. (2002). Analysis of the zone routing protocol. *Stevens Institute of Technology Hoboken, New Jersey, USA*.

Sen, J., Chandra, M. G., Harihara, S. G., Reddy, H., & Balamuralidhar, P. (2007, December). A mechanism for detection of gray hole attack in mobile Ad Hoc networks. *Proceedings of the 2007 6th International Conference on Information, Communications & Signal Processing*, 1-5.

Sen, P., Chaki, N., & Chaki, R. (2008, June). HIDS: Honesty-rate based collaborative intrusion detection system for mobile ad-hoc networks. Proceedings of the 7th Computer Information Systems and Industrial Management Applications (CISIM'08), 121-126.

Sen, J., Ukil, A., Bera, D., & Pal, A. (2008, December). A distributed intrusion detection system for wireless ad hoc networks. Proceedings of the 16th IEEE International Conference on Networks (ICON 2008), 1-6.

Shafer, G. (1976). *A Mathematical Theory of Evidence*. Princeton, New Jersey, USA: Princeton University Press.

Shahnawaz, H., Joshi, R. C., & Gupta, S. C. (2012). Design of detection engine for wormhole attack in adhoc network environment. *International Journal of Engineering and Technology*, 4(6), 381-395.

Shakshuki, E. M., Kang, N., & Sheltami, T. R. (2013). EAACK—a secure intrusion-detection system for MANETs. *IEEE transactions on industrial electronics*, *60*(3), 1089-1098.

Shao, M. H., Lin, J. B., & Lee, Y. P. (2010, June). Cluster-based cooperative back propagation network approach for intrusion detection in MANET. *Proceedings of the 2010 IEEE 10th International Conference on Computer and Information Technology (CIT)*, 1627-1632.

Sharma, B. (2015, August). A Distributed Cooperative Approach to Detect Gray Hole Attack in MANETs. *Proceedings of the Third International Symposium on Women in Computing and Informatics*, 560-563.

Sheikh, R., Chande, M. S., & Mishra, D. K. (2010, September). Security issues in MANET: A review. *Proceedings of the 2010 Seventh International Conference on Wireless and Optical Communications Networks (WOCN)*, 1-4.

Sheltami, T., Al-Roubaiey, A., Shakshuki, E., & Mahmoud, A. (2009). Video transmission enhancement in presence of misbehaving nodes in MANETs. *Multimedia systems*, *15*(5), 273-282.

Shenbagapriya, R., & Kumar, N. (2014, November). A survey on proactive routing protocols in MANETs. *Proceedings of the 2014 International Conference on Science Engineering and Management Research (ICSEMR)*, 1-7.

Shrestha, R., Han, K. H., Choi, D. Y., & Han, S. J. (2010, April). A novel cross layer intrusion detection system in MANET. *Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, 647-654.

Siaterlis, C., & Maglaris, B. (2004, March). Towards multisensor data fusion for DoS detection. *Proceedings of the 2004 ACM symposium on Applied computing*, 439-446.

Singh, S., Woo, M., & Raghavendra, C. S. (1998, October). Power-aware routing in mobile ad hoc networks. *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, 181-190.

Soni, M., Ahirwa, M., & Agrawal, S. (2015, December). A Survey on Intrusion Detection Techniques in MANET. *Proceedings of the 2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, 1027-1032.

Srivastava, A., Mishra, A., Upadhyay, B., & Kumar Yadav, A. (2014, August). Survey and overview of Mobile Ad-Hoc Network routing protocols. *Proceedings of the 2014 International Conference on Advances in Engineering and Technology Research (ICAETR)*, 1-6.

Stamouli, L., Argyroudis, P. G., & Tewari, H. (2005, June). Real-time intrusion detection for ad hoc networks. *Proceedings of the Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM)*, 374-380.

Sterne, D., Balasubramanyam, P., Carman, D., Wilson, B., Talpade, R., Ko, C., Balupari, R., Tseng, C. Y., & Bowen, T. (2005, March). A general cooperative intrusion detection architecture for MANETs. *Proceedings of the Third IEEE International Workshop on Information Assurance*, 57-70.

Subramaniyan, S., Johnson, W., & Subramaniyan, K. (2014). A distributed framework for detecting selfish nodes in MANET using Record-and Trust-Based Detection (RTBD) technique. *EURASIP Journal on Wireless Communications and Networking*, 2014(1), 205.

Sudarsan, D., & Jisha, G. (2012, August). A survey on various improvements of hybrid zone routing protocol in MANET. *Proceedings of the International Conference on Advances in Computing, Communications and Informatics,* 1261-1265.

Sun, B., Osborne, L., Xiao, Y., & Guizani, S. (2007). Intrusion detection techniques in mobile ad hoc and wireless sensor networks. *IEEE Wireless Communications*, 14(5), 56-63

Tabari, M. Y., Hassanpour, H., Pouyan, A., & Saleki, S. (2012, November). Proposing a light weight semi-distributed IDS for mobile ad-hoc network based on nodes' mode. *Proceedings of the 2012 Sixth International Symposium on Telecommunications (IST)*, 948-953.

Takai, M., Martin, J., & Bagrodia, R. (2001, October). Effects of wireless physical layer modeling in mobile ad hoc networks. *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, 87-94.

Tavallaee, M., Stakhanova, N., & Ghorbani, A. A. (2010). Toward credible evaluation of anomaly-based intrusion-detection methods. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *40*(5), 516-524.

Tan, H. X., & Seah, W. (2005). Dynamic topology control to reduce interference in MANETs. *Proceedings of 2nd International Conference on Mobile Computing and Ubiquitous Networking (ICMU 2005), 1-6.*

Tangpong, A., Kesidis, G., Hsu, H. Y., & Hurson, A. (2009, August). Robust Sybil Detection for MANETs. *Proceedings of the 18th International Conference on Computer Communications and Networks (ICCCN 2009)*, 1-6.

Theresa, W. G., & Sakthivel, S. (2017, August). Fuzzy based intrusion detection for cluster based battlefield MANET. *Proceedings of the 2017 IEEE International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*, 22-27).

Trang, C. M., Kong, H. Y., & Lee, H. H. (2006, August). A distributed intrusion detection system for AODV. Proceedings of the Asia-Pacific Conference on Communications (APCC'06), 1-4.

Tsang, C. H., & Kwong, S. (2005, December). Multi-agent intrusion detection system in industrial network using ant colony clustering approach and unsupervised feature extraction. *Proceedings of the IEEE International Conference on Industrial Technology (ICIT 2005)*, 51-56.

Ullah, Z., Khan, M. S., Ahmed, I., Javaid, N., & Khan, M. I. (2016, March). Fuzzy-Based Trust Model for Detection of Selfish Nodes in MANETs. *Proceedings of the 2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, 965-972.

Uppuluri, P., & Sekar, R. (2001). Experiences with specification-based intrusion detection. *Proceedings of the 4th International Symposium Recent Advances in Intrusion Detection*, 172-189.

Veeraiah, N., & Krishna, B. T. (2018, January). Selfish node detection IDSM based approach using individual master cluster node. *Proceedings of the 2018 2nd International Conference on Inventive Systems and Control (ICISC)*, 427-431.

Vigna, G., & Kemmerer, R. A. (1998, December). NetSTAT: A network-based intrusion detection approach. *Proceedings of the 14th Annual Computer Security Applications Conference*, 25-34.

Vij, A., & Sharma, V. (2016, April). Security issues in mobile adhoc network: A survey paper. *Proceedings of the 2016 International Conference on Computing, Communication and Automation (ICCCA)*, 561-566.

Wang, D., Hu, M., & Zhi, H. (2008, July). A survey of secure routing in ad hoc networks. *Proceedings of the Ninth International Conference on Web-Age Information Management (WAIM'08)*, 482-486.

Wang, F., Huang, C., Zhao, J., & Rong, C. (2008, March). IDMTM: A novel intrusion detection mechanism based on trust model for ad hoc networks. *Proceedings of the 22nd International Conference on Advanced Information Networking and Applications (AINA)*, 978-984.

Wang, K. H., & Li, B. (2002). Group mobility and partition prediction in wireless ad-hoc networks. Proceedings of the *IEEE International Conference on Communications (ICC 2002)*, 1017-1021.

Wazid, M., Katal, A., Sachan, R. S., Goudar, R. H., & Singh, D. P. (2013, April). Detection and prevention mechanism for blackhole attack in wireless sensor network. *Proceedings of the 2013 International Conference on Communications and Signal Processing (ICCSP)*, 576-581.

Whitehurst, R. A. (1987). *Expert systems in intrusion detection: A case study*. Computer Science Lab., SRIInternational, Menlo Park, CA.

Wu, B., Chen, J., Wu, J., & Cardei, M. (2007). A survey of attacks and countermeasures in mobile ad hoc networks. *In Wireless network security*, 103-135. Springer US.

Xiao, H., Seah, W. K., Lo, A., & Chua, K. C. (2000). A flexible quality of service model for mobile ad-hoc networks. *Proceedings of the 51st IEEE Conference on Vehicular Technology (VTC 2000)*, 445-449.

Yang, H., Luo, H., Ye, F., Lu, S., & Zhang, L. (2004). Security in mobile ad hoc networks: challenges and solutions. *IEEE wireless communications*, 11(1), 38-47.

Yi, S., Naldurg, P., & Kravets, R. (2001). Security-aware ad hoc routing for wireless networks. *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, 299-302).

Zhang, Y., & Lee, W. (2000, August). Intrusion detection in wireless ad-hoc networks. *Proceedings of the 6th annual international conference on Mobile computing and networking*, 275-283.

Zhang, Y., Lee, W., & Huang, Y. A. (2003). Intrusion detection techniques for mobile wireless networks. *Wireless Networks*, *9*(5), 545-556.

Zhang, Y., & Lee, W. (2005). Security in Mobile Ad-hoc networks. *In Ad hoc networks*, 249-268. Springer US.

Zeng, Y., Chen, Z., Qiao, C., & Xu, L. (2011, May). A cluster header election scheme based on auction mechanism for intrusion detection in MANET. *Proceedings of the 2011 International Conference on Network Computing and Information Security (NCIS)*, 433-437.

Zhou, L., & Haas, Z. J. (1999). Securing ad hoc networks. *IEEE network*, 13(6), 24-30.